# Outline: CoAd Lectures

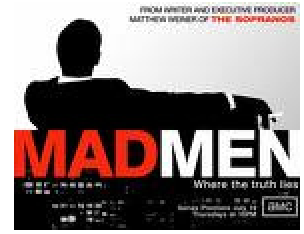
L1
- **Introduction**
- **Online advertising background**
- **Business models, Campaigns**

L2
- **Technology and Economics**
  - Forward Markets
    - Gradient Descent, Operations research, LP, QP

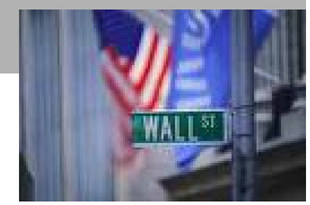
L3
  - Auction Theory and Game Theory
  - Spot Markets
    - ML, Ad quality, Ranking, Budgeting

L4
- **New Directions**
- **Challenges in online advertising**
- **Summary**

Business, Gold rush




Tech



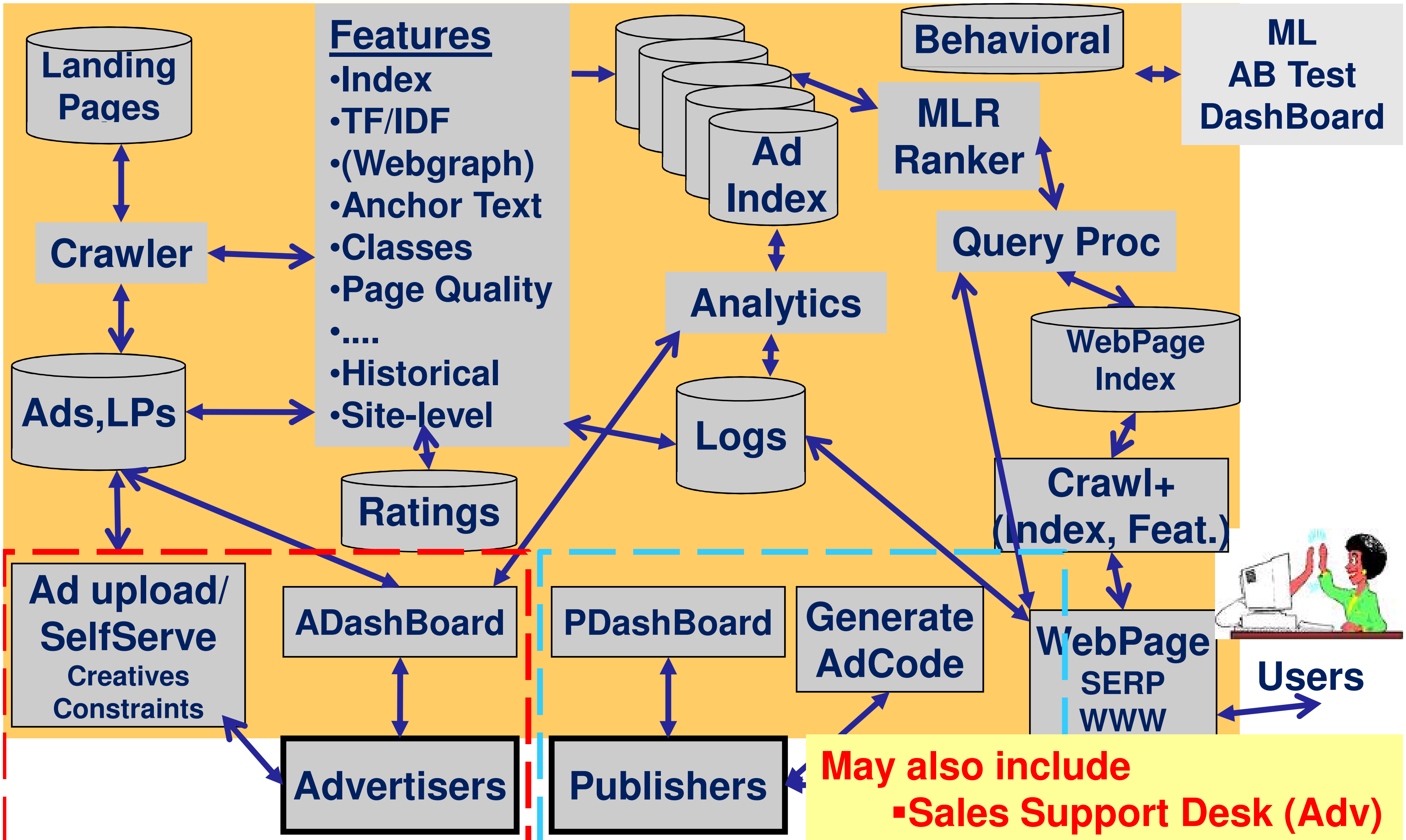

Hot Areas

| CoAd Lectures | | |
|---|---|---|
| Friday | 9/11/2009 | 10:30-12:00 |
| Saturday | 9/12/2009 | 8:30-10:00 |
| Sunday | 9/13/2009 | 8:30-10:00 |
| Monday | 9/14/2009 | 8:30-10:00 |

# Course philosophy

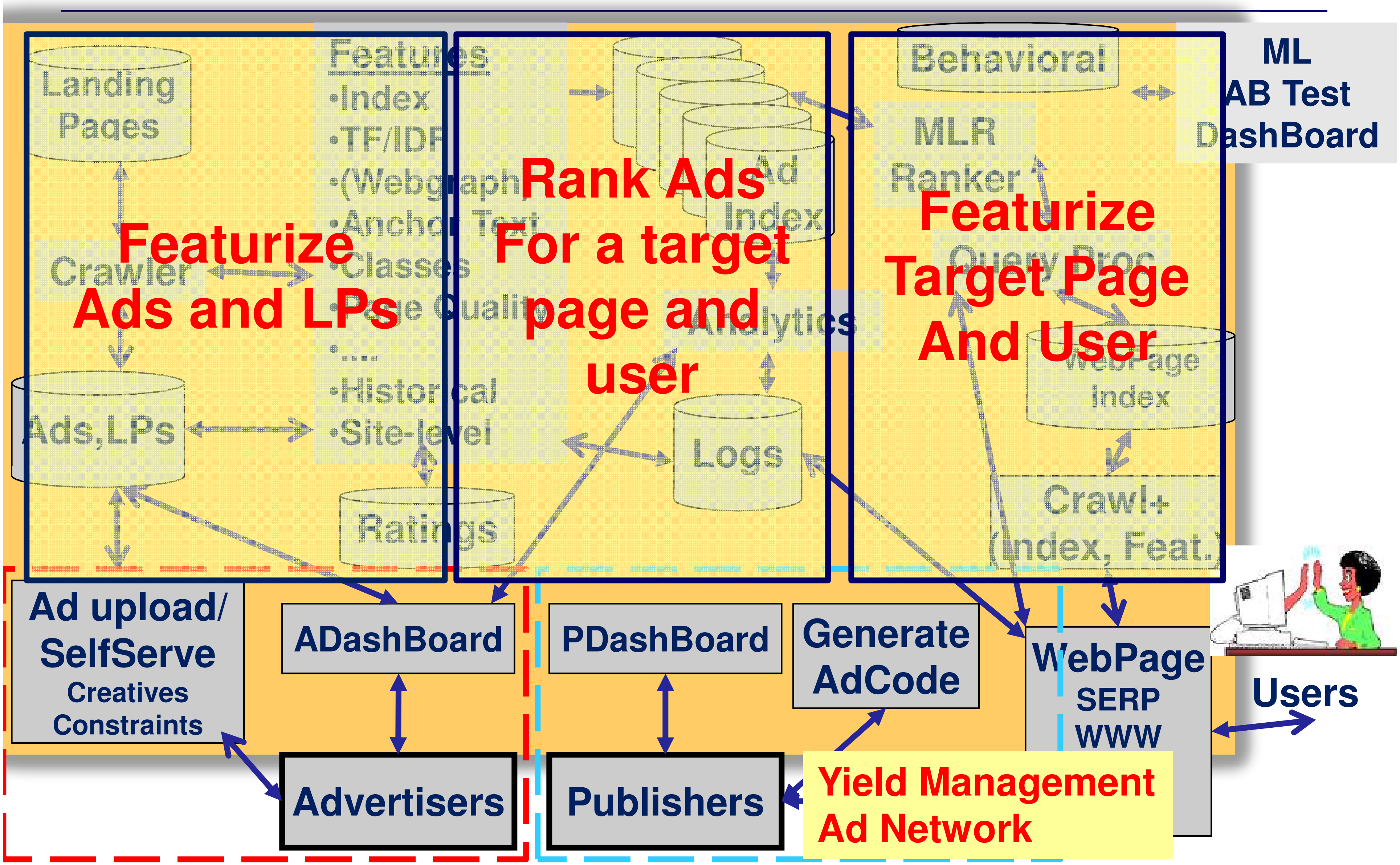
- **Socratic Method (more inspiration than information)**
  - participation strongly encouraged (please state your name and affiliation)
- **Highly interactive and adaptable**
  - Questions welcome!!
- **Lectures emphasize intuition, less rigor and detail**
  - Build on lectures from other faculty
  - Background reading will provide more rigor & detail
- **Action Items**
  - Read suggested books first (and then papers), read/**write** Wikipedia, watch/**make** YouTube videos, take courses, participate in competitions, do internships, network
  - Prototype, simulate , publish, participate
  - Classic (core) versus trendy (applications)

# Ad Network Architecture: Spot Market

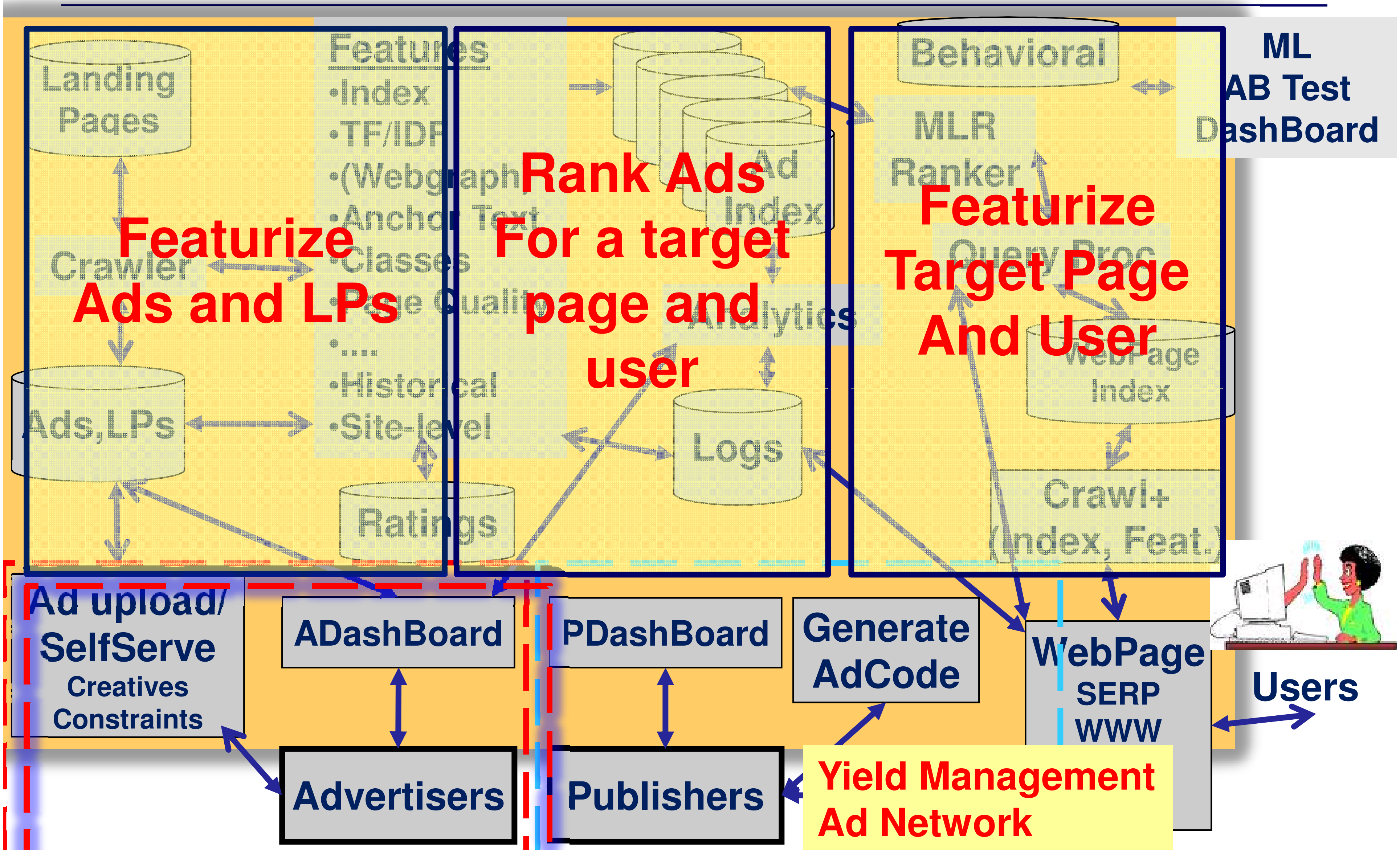



May also include
- Sales Support Desk (Adv)
- Media Buyer Desk (Pub)

# Ad Network Architecture: Spot Market

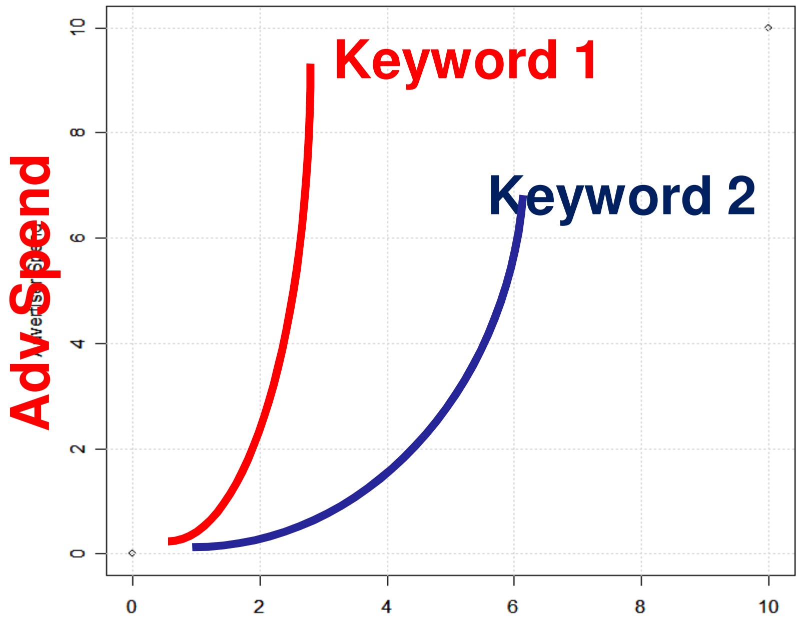


# Ad Network Architecture: Spot Market



# What is wrong with my ad?

| | OCT | NOV | DEC | JAN | FEB# |
|---|---|---|---|---|---|
| Cost | $1000 | $2000 | $2000 | $1500 | $1000 |
| Conversions | 20 | 15 | 10 | 15 | 20 |
| CPA ($/conv) | $50/conv | $133/conv | $200/conv | $100/conv | $50/conv |




# [Kourosh Gharachorloo, Google, 2007]

**Consumer behavior**
**– seasonality, time-of-day**
**– demographics: geo, age, income, etc.**
**Advertiser Side**
**-Ad Creative**
**-Landing page experience**
**-Pricing**
**Publisher Side**
**-KW, TP, Position on Page**

# Ad placement

- **Which Ads: Which ad creatives, landing page should the advertiser use?**
- **Real Estate:**
  - Which pages should the advertiser put ads on?
    - Website, Category, Keyword
  - Book pages based on
    - A forward schedule
    - A non-guaranteed fashion (specify bid, budget and schedule)
- **Advertiser can do all this …**
  - By themselves
  - Or through an ad network / ad agency
- **Approaches**
  - Guess, Hire Experts, AB Testing, Fractional Factorial Design
  - Take a portfolio approach (see next section)

# Optimizing Ads: SEMs

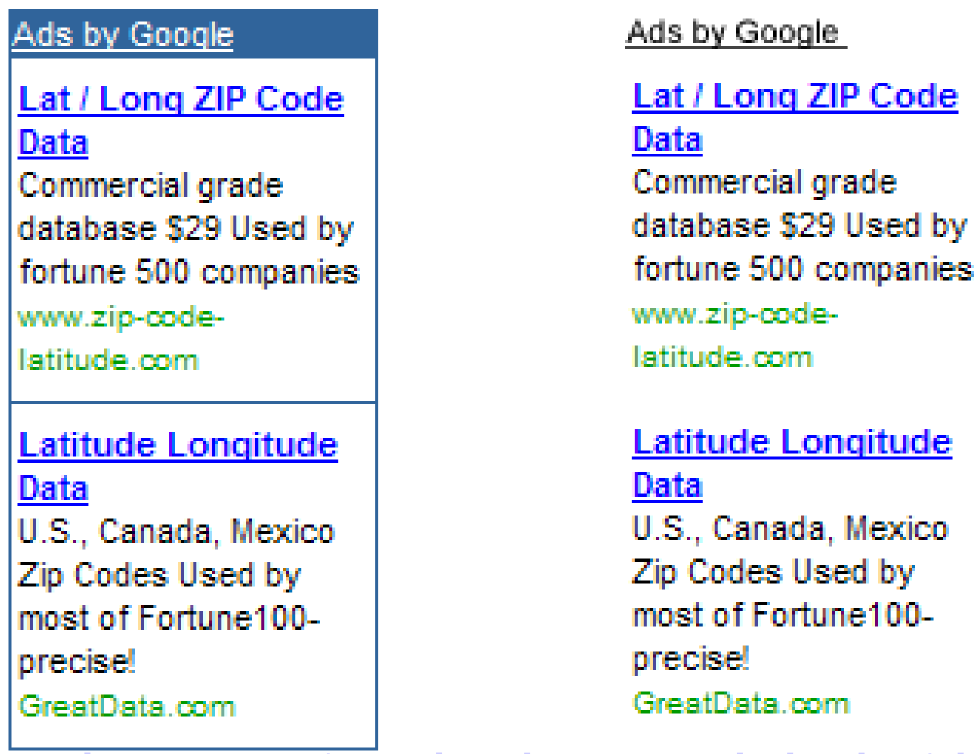
- **Search engine marketing (SEM) refers to services that determine optimal ad placement.**
  - Many SEMs leverage AB Testing and DOE
  - SEMs optimize ad creatives, landing page, keywords
    - Efficient Frontier(Keyword Mgt.)
    - Offermatica (Ad creative, landing page)
    - Optimost (Ad creative, landing page)
    - TaguchiNow (Ad creative, landing page)
    - …..
- **SEOs (Search engine optimization) refers to the process of tailoring a web site to optimize its (unpaid, or "left side", or "organic") ranking for a given set of keywords or phrases.**
  - For more details see SEO Literature

# A/B Test: Border or not to Border?

- **The ad unit has a border around it at present and you want to know if removing the border would have any positive effect on the performance of the ad. This is where A/B testing comes in.**

Advert 'A' Advert 'B'

**(From 10^3 to 10^6 in online advertising. see Statistical power)**




**[For ads see: http://www.sitetoolcenter.com/google-adsense-optimization/ab-testing.php]**

# Which Ad Creative? Landing page?

- **Design of experiments (DOE) (versus AB Testing)**
  - Which ads are working?
  - Is the ad creative working well?
  - Is the landing page experience working well?
  - What features of creatives/landing pages work?
    - Colour? Location? Text Style? Navigation? Action words?
- **Fractional factorial designs are experimental designs consisting of a carefully chosen subset (fraction) of the experimental runs of a full factorial design.**
  - The subset is chosen so as to exploit the sparsity-of-effects principle to expose information about the most important features of the problem studied, while using a fraction of the effort of a full factorial design in terms of experimental runs and resources

# Dell DOE Study [TaguchiNow.com]

- **Target business employees with computers for personal use**
  - Dell selected the Employee Purchase Program (EPP) e-mail campaigns as the initial implementation of the Taguchi-based ad optimization methodology
  - EPP e-mail advertising campaigns are targeted to 450,000 individuals: 250,000 corporate employees, 150,000 government employees, and 50,000 professors at schools or universities, all of them users of Dell computers at work.
  - The aim of Dell's EPP e-mail campaigns is to sell computers, software and peripherals to these individuals for their personal use leveraging the fact that they are already familiar with the brand.
  - As an enticing benefit, Dell's EPP members enjoy discounts of up to 12% and special promotions like free shipping, product bundles, and others.

# DOE: Taguchi Testing Array

Fig. 4.1, right, shows a Taguchi testing array that was selected to analyze 7 factors with 2 options and 4 factors with 3 options in only 18 test e-mails. This allowed to test 10,368 campaigns with only 18 tests – a small fraction of all possible combinations (only 0.2%!).

*Fig. 4.1. Factors and options in the Taguchi testing array.*

11 factors considered

| Factor | Option 1 | Option 2 | Option 3 |
|---|---|---|---|
| Promotion | Single | several | - |
| Teaser | yes | no | - |
| Financing | yes | no | - |
| Price | high-end | low-end | - |
| S&P* Promotion | yes | no | - |
| Discount | 5% | 10% | - |
| Image | product | people | - |
| Subject Line | creative | promo | dated |
| Headline | creative | promo | seasonal |
| Configurations | two | one | none |
| Product Mix | both | notebook | desktop |

(*) Software & Peripherals

**7 factors(2 options); 4 factors with 3 options**
**18 ads out of 10,368 ads are tested**

# 18 out of 10,368 Ads Tested

## Design Matrix

| | FACTORS | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Test # | Promo | Teaser | Finance | Price | S&P | Discount | Image | Subject | Headline | Configs | Product |
| 1 | single | no | no | high-end | yes | 5% | product | creative | creative | two | both |
| 2 | single | no | no | high-end | no | 5% | people | promo | promo | one | notebook |
| 3 | single | no | no | low-end | yes | 10% | product | dated | seasonal | none | desktop |
| 4 | single | no | yes | high-end | yes | 5% | product | promo | promo | none | desktop |
| 5 | single | no | yes | high-end | no | 5% | people | dated | seasonal | two | both |
| 6 | single | no | yes | low-end | yes | 10% | product | creative | creative | one | notebook |
| **7** | **single** | **yes** | **no** | **high-end** | **yes** | **5%** | **people** | **creative** | **seasonal** | **one** | **desktop** |
| 8 | single | yes | no | high-end | no | 10% | product | promo | creative | none | both |
| 9 | single | yes | no | low-end | yes | 5% | product | dated | promo | two | notebook |
| 10 | several | no | no | high-end | yes | 10% | product | dated | promo | one | both |
| 11 | several | no | no | high-end | no | 5% | product | creative | seasonal | none | notebook |
| 12 | several | no | no | low-end | yes | 5% | people | promo | creative | two | desktop |
| 13 | several | no | yes | high-end | yes | 5% | people | dated | creative | none | notebook |
| 14 | several | no | yes | high-end | no | 10% | product | creative | promo | two | desktop |
| 15 | several | no | yes | low-end | yes | 5% | product | promo | seasonal | one | both |
| 16 | several | yes | no | high-end | yes | 10% | product | promo | seasonal | two | notebook |
| 17 | several | yes | no | high-end | no | 5% | product | dated | creative | one | desktop |
| 18 | several | yes | no | low-end | yes | 5% | people | creative | promo | none | both |

**18 test email ads were sent to 2,000 people each**

# Send each email ad to multiple groups

| Test # | RESPONSE DATA | | | |
|---|---|---|---|---|
| | Open Rate | | Sales | |
| 1 | 4.8% | 5.7% | $ - | $ - |
| 2 | 5.2% | 6.1% | $ - | $ - |
| 3 | 7.2% | 8.4% | $1,638 | $1,530 |
| 4 | 10.5% | 11.6% | $1,913 | $2,215 |
| 5 | 6.0% | 7.3% | $1,234 | $1,755 |
| 6 | 5.0% | 5.8% | $ - | $ - |
| 7 | 12.7% | 13.8% | $4,919 | $4,522 |
| 8 | 7.9% | 8.8% | $2,890 | $2,933 |
| 9 | 7.2% | 8.8% | $1,296 | $1,104 |
| 10 | 5.5% | 6.4% | $ - | $ - |
| 11 | 4.9% | 5.8% | $ - | $ - |
| 12 | 4.2% | 5.0% | $ - | $ - |
| 13 | 5.5% | 6.4% | $ - | $ - |
| 14 | 5.7% | 6.1% | $ - | $ - |
| 15 | 5.2% | 5.8% | $ - | $ - |
| 16 | 7.4% | 8.3% | $1,212 | $896 |
| 17 | 6.3% | 7.0% | $1,076 | $1,555 |
| 18 | 9.9% | 10.9% | $2,448 | $1,998 |

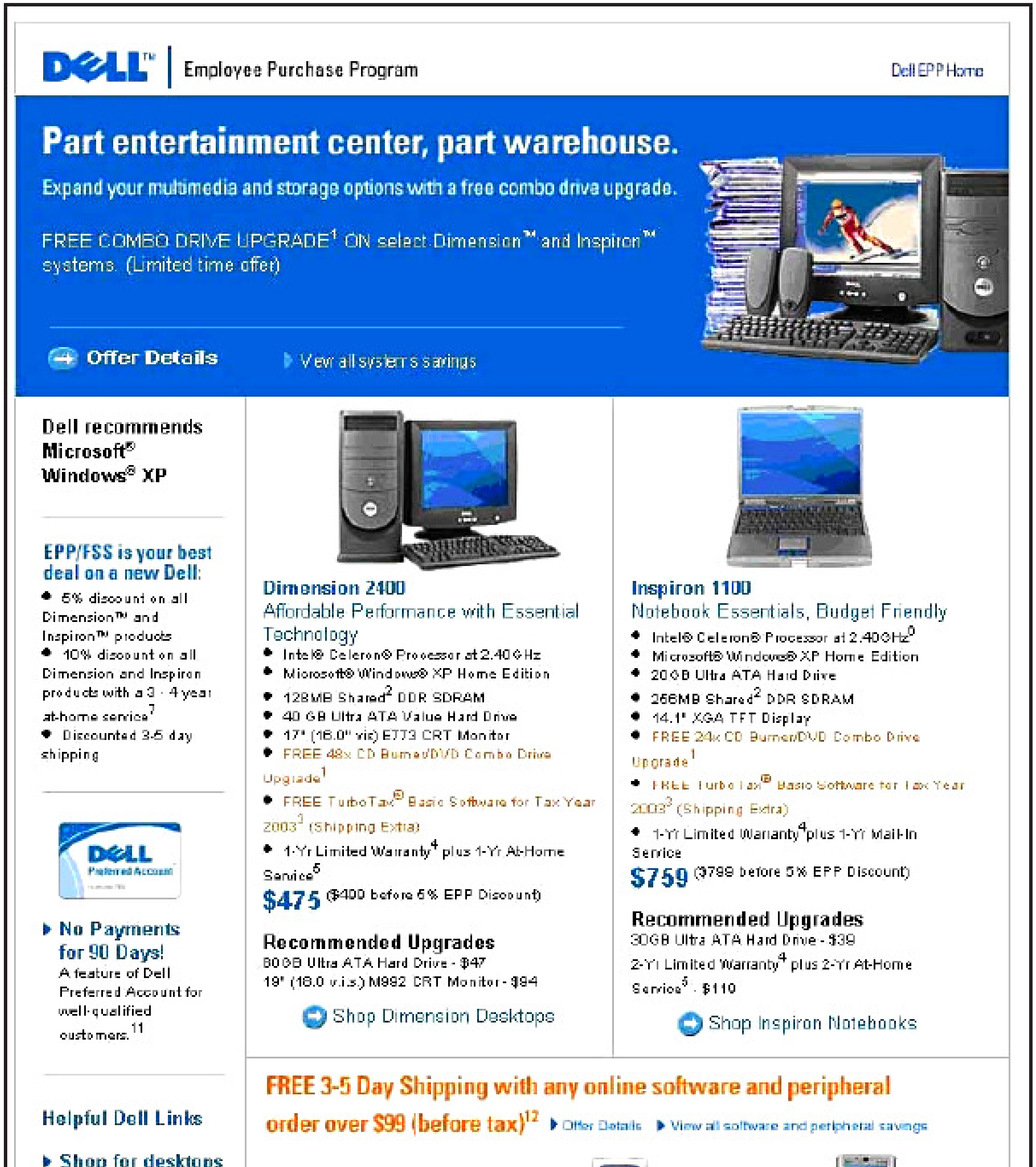
**Each group of people and its response (CTR or Sales) becomes an example. E.g., 10 groups leads to 180 examples**

**Perform regression on data**

# Most Influential Factors

| FACTOR | OPTIMUM OPTION | INFLUENCE |
|---|---|---|
| **Teaser** | **yes** | **34%** |
| **Product Mix** | **desktop** | **17%** |
| **Promotion** | **primary** | **16%** |
| **Headline** | **seasonal** | **13%** |
| **Configurations** | **none** | **13%** |
| **Subject Line** | **dated** | **7%** |
| Financing | yes or no | 0 % |
| Price | high-end of low-end | 0 % |
| S & P Promotion | yes or no | 0 % |
| Discount | 5% or 10% | 0 % |
| Photo | Product or Lifestyle | 0 % |

- **Before optimization**



# Features that worked well


**Select multiple ads**

Fig. 6.2.
April 28, 2006
Taguchi 3:
No Seasonality,
One Shopping Button,
Two Coupons.

# DOE Works

- **Click Through Rate increase: 5.2 times**
- **7.1 times more sales per e-mail**
- **Annual sales *before optimization: $8,900,000***
- **Annual sales *after optimization: $63,100,000***
- ***Data-based (as opposed to intuitions)!!***
- ***Crowd-sourcing at its most efficient!!***
- ***[TaguchiNow.com]***

| CAMPAIGN | Type | Audience Size | Total Clicks | Click Thru | Total Sales | $sales/ e-mail |
|---|---|---|---|---|---|---|
| June 17, 2004 EPP email | Control email | 268,610 | 8,058 | 3.00% | $90,678 | $0.34 |
| June 17, 2004 EPP email | Optimized email | 142,633 | 22,379 | 15.69% | $345,095 | $2.42 |

## Full Factorials

| Number Factors | Main Effects | Order of Interactions 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2 | 1 | | | | | | | | |
| 3 | 3 | 3 | 1 | | | | | | | |
| 4 | 4 | 6 | 4 | 1 | | | | | | |
| 5 | 5 | 10 | 10 | 5 | 1 | | | | | |
| 6 | 6 | 15 | 20 | 15 | 6 | 1 | | | | |
| 7 | 7 | 21 | 35 | 35 | 21 | 7 | 1 | | | |
| 8 | 8 | 28 | 56 | 70 | 56 | 28 | 8 | 1 | | |
| 9 | 9 | 36 | 84 | 126 | 126 | 84 | 36 | 9 | 1 | |
| 10 | 10 | 45 | 120 | 210 | 252 | 210 | 120 | 45 | 10 | 1 |

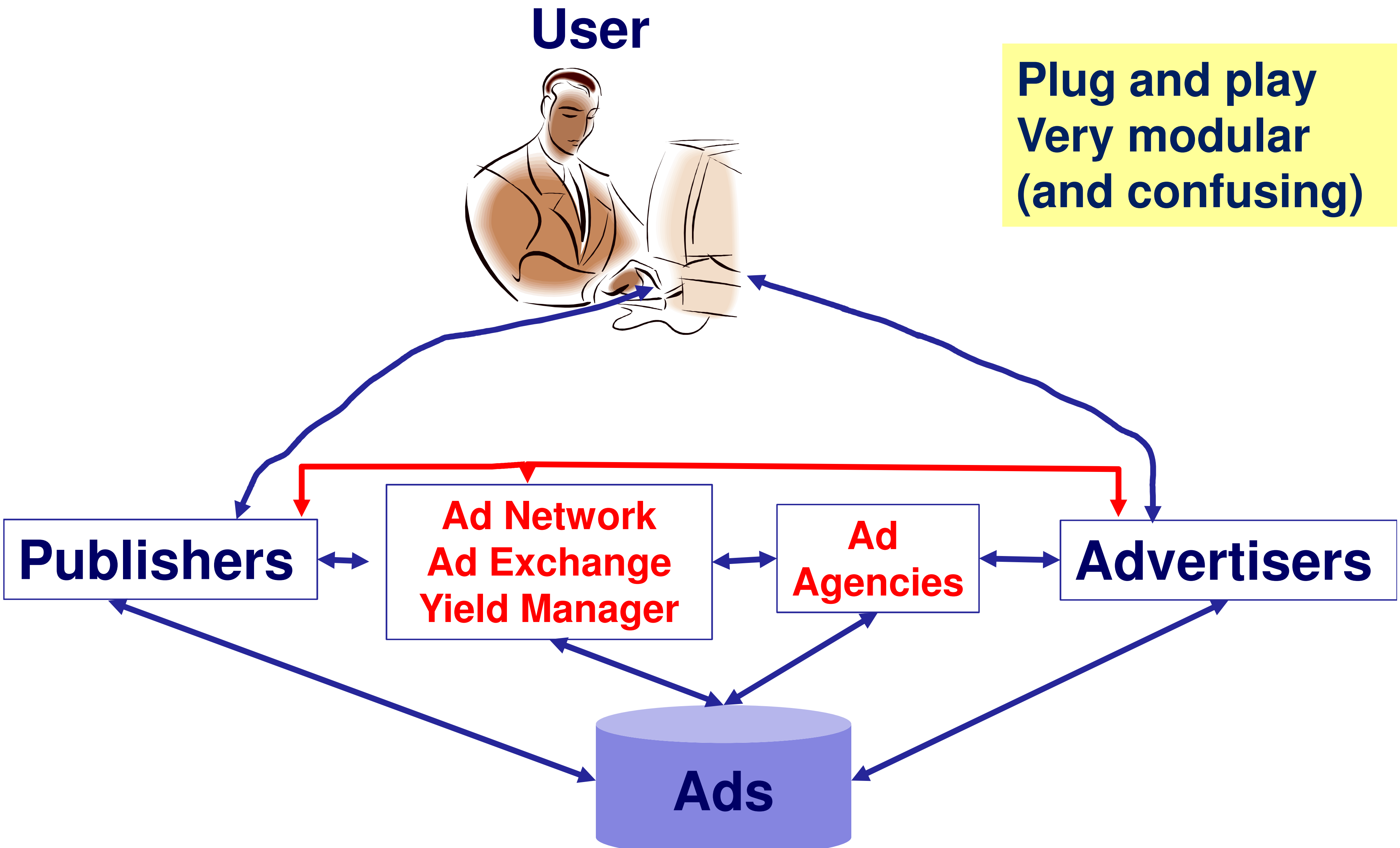
*Box et al. (1978) "There tends to be a redundancy in [full factorial designs] – redundancy in terms of an excess number of interactions that can be estimated …*
*Fractional factorial designs exploit this redundancy …"* → **philosophy**

# Fractional Factorial Design

- **Multiple factors impact the performance of an ad/landing page**
- **DOE provides a means to quantify the impact of each factor in an efficient manner**
- **In the full factorial design, as the number of factors increases, the required number of groups increases exponentially.**
- **The fractional factorial design reduces the number of groups (ads/LPs in the case of advertising) that need to be evaluated**
  - FFD based on orthogonalization of features (use prescribed recipes: read feature combinations and data requirements from tables)
  - Used in automobile manufacturing industry (Developed 1960s)
  - Linear Regression of CTR variable using the 18 input variables
- **Used by Optimost, Offermatica for Ad/LP optimization**

# Online Advertising

Plug and play
Very modular
(and confusing)



# The Advertiser's View

- **Some Tools and Pointers:**
  - Google's information for advertisers and keyword tool.
  - Yahoo!'s search marketing resources , including the View Bids Tool.
  - Ask's sponsored listing basics.
  - Third-party optimization and management tools and services such as Efficient Frontier, Did-It, Atlas Search, Bloofusion, SearchRev, and Hitwise.
  - Some keyword bidding robots: Atlas Search, Did-It's Maestro Client, BidRank, Dynamic BidMaximizer, Apex Pacific, PPC Management, and Search Marketing Tools PPC BidTracker.
  - The Search Engine Marketing Professional Organization.
  - "An Adaptive Algorithm for Selecting Profitable Keywords for Search-Based Advertising Services. Rusmevichientong, Williamson.
  - Optimal Bidding on Keyword Auctions. Kitts, Leblanc.

# Challenges on Advertiser Side

- **Ad Network needs to provide services**
  – Keywords suggestions
  – Exact Match vs. Broad match **(techniques??)**
  – Keyword disambiguation (R the statistical package vs. the letter R; what does the advertiser mean?)
  – *Commercial intent of keywords (contextual advertising)*
  – *When to pass on an adcall? Sentiment*
  – Geo targeting
  – Categorization (organize ads by category, limit publishers by category; e.g., porn, gambling, religious, sports, etc.)
  – Bundling Paradox: More segmentation implies expensive CPM but smaller less competitive marketplace?

# Keyword Suggester

Enter one keyword or phrase per line:

data mining

☑ Use synonyms

Get More Keywords

Choose data to display: Possible Negative Keywords

Use the Possible Negatives column below to add a negative keyword for any keyword phrase that doesn't specifically reflect your business or service. For example, if you advertise on the keyword books, and you don't sell used books, you can add the negative keyword -used. This means your ad won't appear for the keyword used books. Learn More about using and choosing negative keywords.

**More specific keywords** - sorted by relevance

| Keywords | August Search Volume | Possible Negatives |
|---|---|---|
| data mining | | No Negative |
| data mining software | | Add negative: -software » |
| data mining tools | | Add negative: -tools » |
| web data mining | | Add negative: -web » |
| data mining tool | | Add negative: -tool » |
| data mining techniques | | Add negative: -techniques » |
| data mining jobs | | Add negative: -jobs » |
| what is data mining | | Add negative: -what is » |
| data mining tutorial | | Add negative: -tutorial » |
| data mining algorithms | | Add negative: -algorithms » |
| data mining solutions | | Add negative: -solutions » |
| introduction to data mining | | Add negative: -introduction to » |
| data mining solution | | Add negative: -solution » |
| data mining applications | | Add negative: -applications » |
| data mining definition | | Add negative: -definition » |

# Outline

- **Introduction**
- **Online advertising background**
- **Business models**
- **Creating an online ad campaign**
- **Technology and Economics**
  - Advertisers (optimizing ROI thru ads and ad placement)
  - Publishers (optimizing revenue and consumer satisfaction)
    - Forward Markets
    - Spot Markets (Auction Systems, Ad Quality, Budgeting)
- **New Directions**
- **Challenges in online advertising**
- **Summary**

$$Bid_{Ad}$$

$$Bid_{Ad} * CTR_{Ad}$$

$$Bid_{Ad} * CTR_{Ad} * ThrottleFa\,ct$$

# Traditional Sales/Forward Markets

**ONLINE ADVERTISING RATE SHEET**

**Chicagoreader.com**
**More than 100,000 unique users and 1,000,000 pageviews every week**

Chicagoreader.com focuses on function, popular features, and daily updates. Our homepage is an essential portal into local arts, entertainment, and issues. *Chicago Reader On Film* archives more than 10,000 capsule movie reviews. The *Reader Restaurant Finder* is an online guide to more than 3,000 area restaurants. *Reader Online Classifieds* are a complete online marketplace for apartment rentals, houses and condos, jobs, personals, services, and more.

**Online Ad Rates**

| | |
|---|---|
| **50,000 - 199,000 ad impressions** | **$12 per 1,000** |
| **200,000 - 499,000 ad impressions** | **$10 per 1,000** |
| **500,000 + ad impressions** | **$8 per 1,000** |

Online Ad Sizes

| | | |
|---|---|---|
| Leaderboard | Top of page | 728 pixels x 90 pixels |
| Skyscraper | Right hand column | 160 pixels x 600 pixels |
| Rectangle | Within text | 300 pixels x 250 pixels |

**Hybrid Advertising: Print + Online**
**50% of our print readers use chicagoreader.com. (2006 MRI Survey)**

Advertisers can increase the reach and frequency of their print advertising with simultaneous ad impressions on chicagoreader.com. Reach our total audience with the combination of the Chicago Reader and chicagoreader.com.

**High human touch**
**-Use rate cards**
**-Auction/negotiate**

# Online Advertising Marketplaces

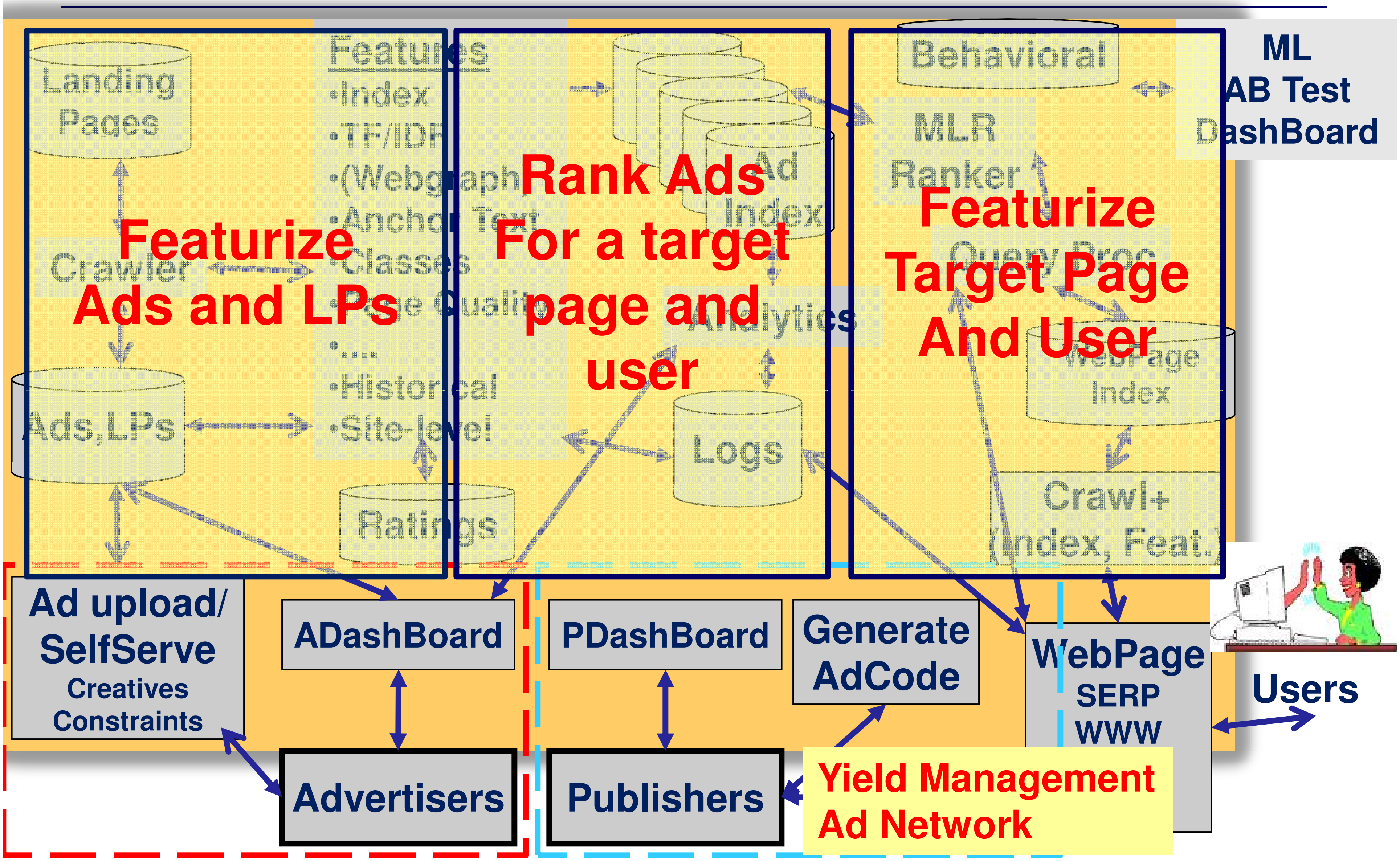
Static

- **Manual sale in large batches (1000s); Charge advertiser on a CPM basis**
  - Price negotiated up front ; can be human-intensive
  - ~1994 onwards
  - Forward Markets/Guaranteed delivery

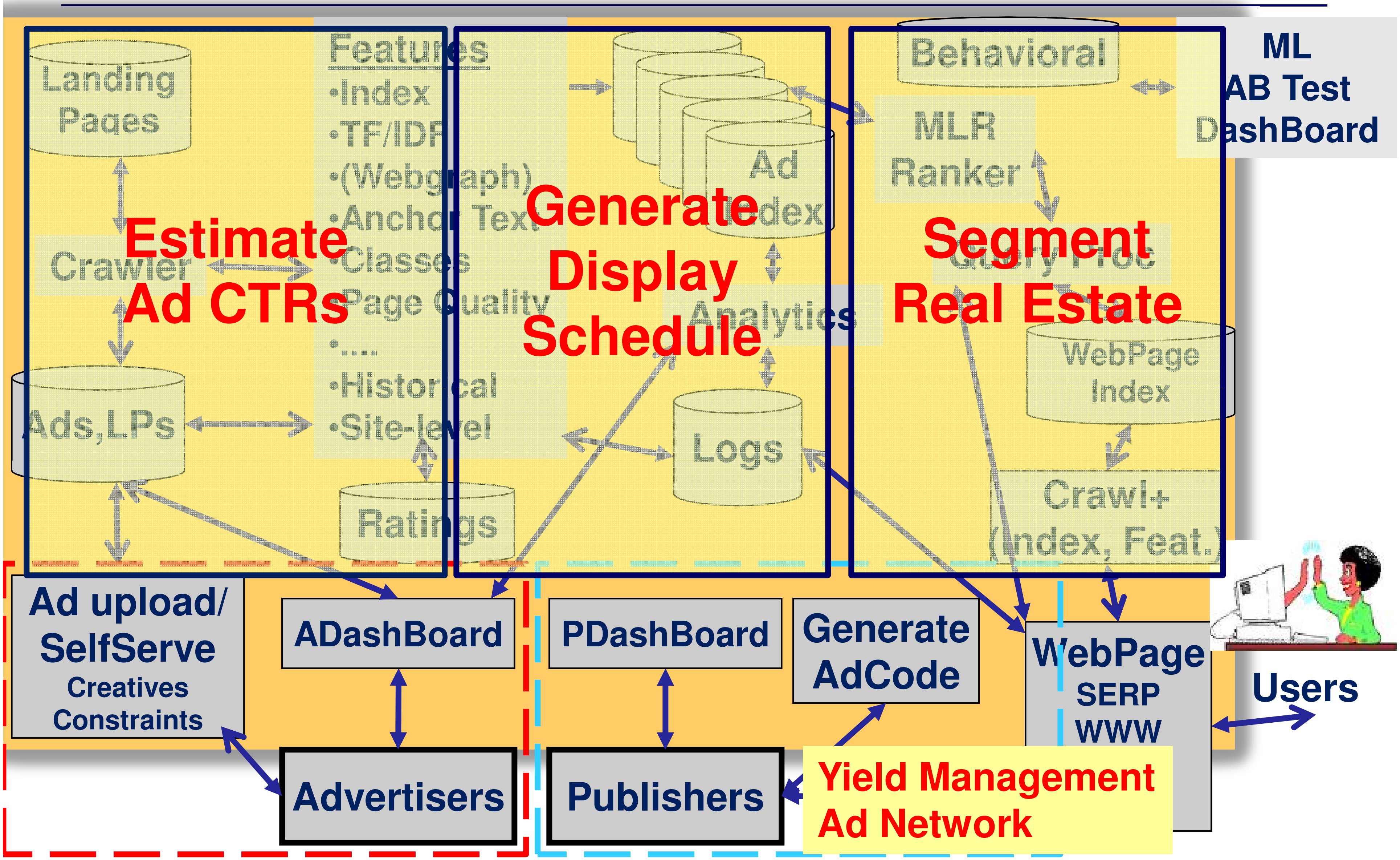
Dynamic

- **Self-serve; Charge advertiser on a CPC basis (1997)**
- **Auction on a per impression basis**
  - First-price auction, a la Goto/Overture (1997)
  - Second-price auction (GSP); Google (2002) and Yahoo
  - VCG auction (not adapted in practice)
  - Spot market

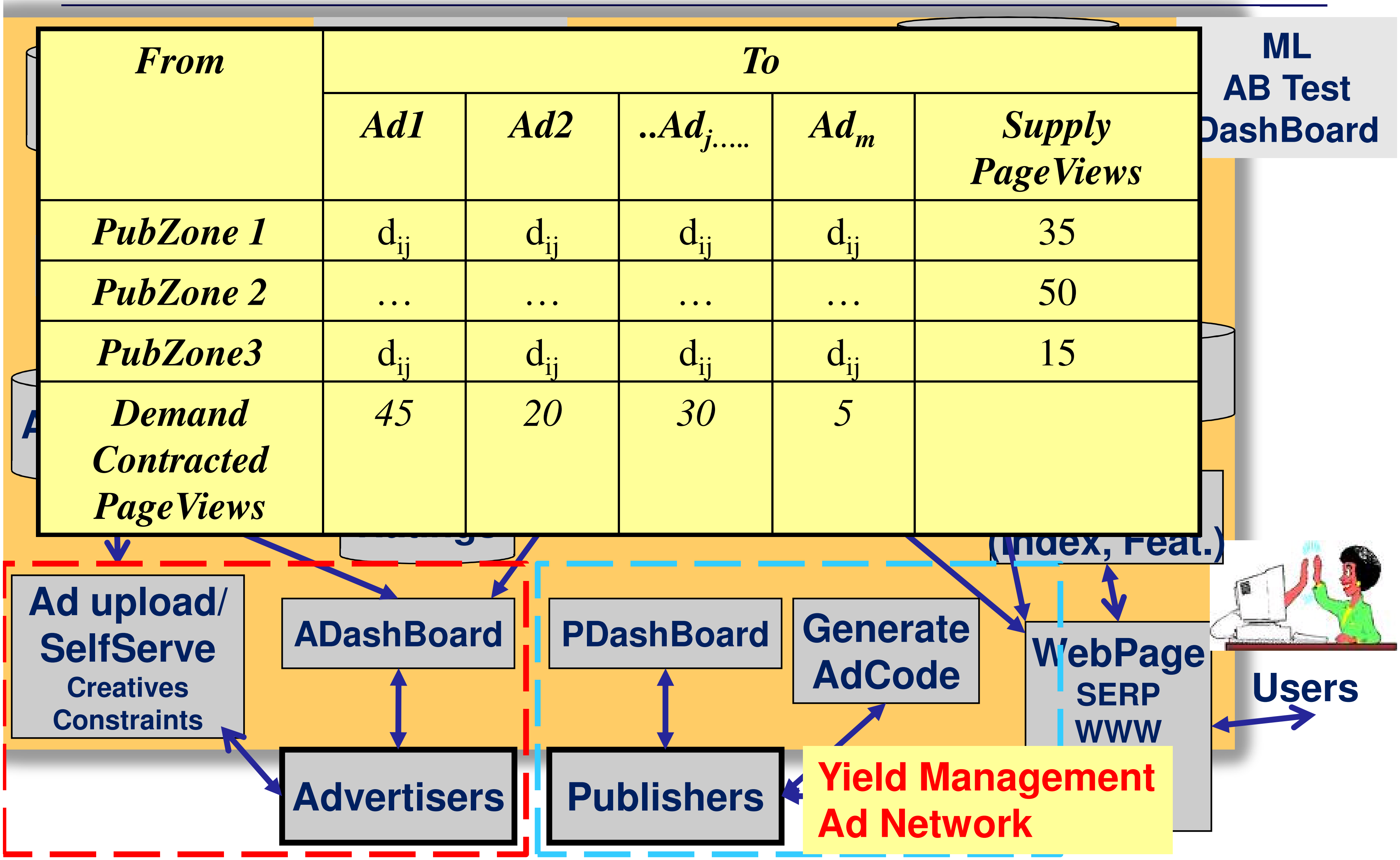
# Ad Network Architecture: Spot Market



# Ad Network Architecture: Forward Market



# Ad Network Architecture: Forward Market




| *From* | *To* | | | | |
|---|---|---|---|---|---|
| | *Ad1* | *Ad2* | *..Ad$_j$.....* | *Ad$_m$* | *Supply PageViews* |
| *PubZone 1* | $d_{ij}$ | $d_{ij}$ | $d_{ij}$ | $d_{ij}$ | 35 |
| *PubZone 2* | … | … | … | … | 50 |
| *PubZone3* | $d_{ij}$ | $d_{ij}$ | $d_{ij}$ | $d_{ij}$ | 15 |
| *Demand Contracted PageViews* | *45* | *20* | *30* | *5* | |

# Maximize Revenue: Ad Allocation Example

| *From* | *To* | | | | |
|---|---|---|---|---|---|
| | *Ad1* | *Ad2* | *..Ad$_j$.....* | *Ad$_m$* | *Supply PageViews* |
| *PubZone 1* | $d_{ij}$ | $d_{ij}$ | $d_{ij}$ | $d_{ij}$ | 35 |
| *PubZone 2* | … | … | … | … | 50 |
| *PubZone3* | $d_{ij}$ | $d_{ij}$ | $d_{ij}$ | $d_{ij}$ | 15 |
| *Demand Contracted PageViews* | *45* | *20* | *30* | *5* | |

**Use LP to generate the Ad display schedule to maximize my revenue (or rev proxy, .i.e., CTR)**

# Ad Networks and Optimisation

- **Allocation of Ads to Publisher real estate**
  - Give ads play in network
    - Optimize *revenue* subject to ….
- **Inventory Management**
  - Contract as many impressions as possible but don't oversell
- **Media Buyer (Arbitrage) (NLP-problem)**
  - Talks to publisher
  - Determine publisher mix for network
    - Optimize *publisher mix* subject to constraints

# Technology

- ***Infrastructure (not going to discuss here)***
  - Commodity components such as Distributed systems, Logging Systems, DBMS, OLAP, Reporting, Load balancers Firewalls, server farms, data-centers, Hadoop, GridSQL, etc.
- **Targeting, Analysis, Yield management**
  - This is where the money ("^$d+,d+[BbMm]illion") is at!

# Outline

- **Introduction**
- **Online advertising background**
- **Business models**
- **Creating an online ad campaign**
- **Technology and Economics**
  - Advertisers (optimizing ROI thru ads and ad placement)
  - Publishers (optimizing revenue and consumer satisfaction)
    - Forward Markets (Operations Research, segmentation)
    - Spot Markets (Auctions, Game Theory, Ad Quality, Budgeting)
- **New Directions**
- **Challenges in online advertising**
- **Summary**

# Forward Markets

- **Gradient Descent**
- **Linear Programming**
- **Quadratic Programming**
- **Allocation of Ads to Publisher real estate**
  - Give ads play in network
    - Optimize *revenue* subject to ….
- **Inventory Management**
  - Contract as many impressions as possible but don't oversell
- **Media Buyer (Arbitrage)**
  - Frame as a non-linear programming (NLP) problem
  - Talks to publisher
  - Determine publisher mix for network
    - Optimize *publisher mix* subject to constraints

# Gradient Descent

- **Common tool in optimisation, machine learning**
  - Perceptron learning, logistic regression, SVMs, LP, QP, NN, etc.
- **Gradient descent is a first-order optimization algorithm. To find a local minimum of a function using gradient descent, one takes steps proportional to the *negative* of the gradient (or the approximate gradient) of the function at the current point.**
  - If instead one takes steps proportional to the gradient (i.e., not negative), one approaches a local maximum of that function; the procedure is then known as gradient ascent.
- **Basic gradient descent (and other variations) works well…..**

**[Wikipedia]**

# A real-valued function decreases fast..

Gradient descent is based on the observation that if the real-valued function $F(\mathbf{x})$ is defined and differentiable in a neighborhood of a point $\mathbf{a}$, then $F(\mathbf{x})$ decreases *fastest* if one goes from $\mathbf{a}$ in the direction of the negative gradient of $F$ at $\mathbf{a}$, $-\nabla F(\mathbf{a})$. It follows that, if

$$\mathbf{b} = \mathbf{a} - \gamma \nabla F(\mathbf{a})$$

for $\gamma > 0$ a small enough number, then $F(\mathbf{a}) \geq F(\mathbf{b})$. With this observation in mind, one starts with a guess $\mathbf{x}_0$ for a local minimum of $F$, and considers the sequence $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots$ such that

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n),\ n \geq 0.$$

We have

$$F(\mathbf{x}_0) \geq F(\mathbf{x}_1) \geq F(\mathbf{x}_2) \geq \cdots,$$

so hopefully the sequence $(\mathbf{x}_n)$ converges to the desired local minimum. Note that the value of the *step size* $\gamma$ is allowed to change at every iteration.

[Wikipedia]

# Gradient Descent Example

Gradient descent is based on the observation that if the real-valued function $F(\mathbf{x})$ is defined and differentiable in a neighborhood of a point $\mathbf{a}$, then $F(\mathbf{x})$ decreases *fastest* if one goes from $\mathbf{a}$ in the direction of the negative gradient of $F$ at $\mathbf{a}$, $-\nabla F(\mathbf{a})$. It follows that, if

$$\mathbf{b} = \mathbf{a} - \gamma \nabla F(\mathbf{a})$$

for $\gamma > 0$ a small enough number, then $F(\mathbf{a}) \geq F(\mathbf{b})$. With this observation in mind, one starts with a guess $\mathbf{x}_0$ for a local minimum of $F$, and considers the sequence $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots$ such that

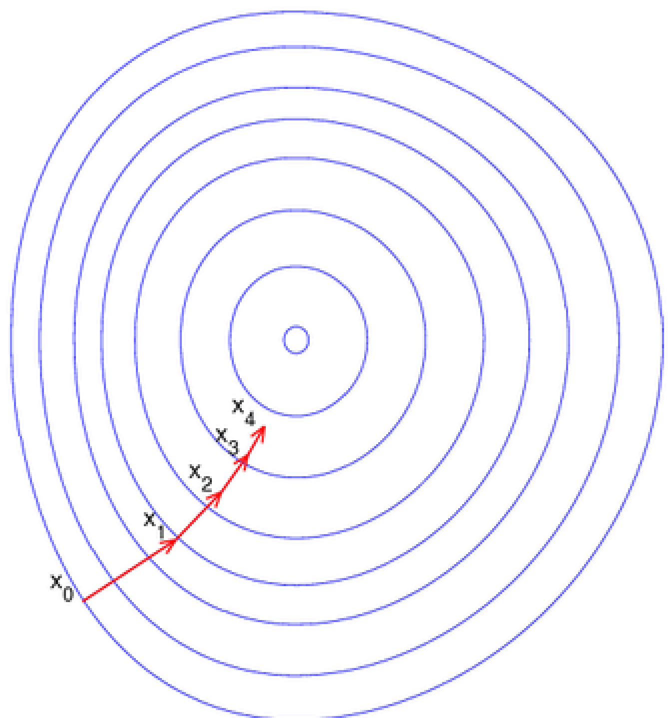
$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n), \; n \geq 0.$$

We have

$$F(\mathbf{x}_0) \geq F(\mathbf{x}_1) \geq F(\mathbf{x}_2) \geq \cdots,$$

so hopefully the sequence $(\mathbf{x}_n)$ converges to the desired local minimum. N
*step size* $\gamma$ is allowed to change at every iteration.

This process is illustrated in the picture to the right. Here $F$ is assumed to be
and that its graph has a bowl shape. The blue curves are the contour lines, th
which the value of $F$ is constant. A red arrow originating at a point shows the
gradient at that point. Note that the (negative) gradient at a point is orthogonal
going through that point. We see that gradient *descent* leads us to the bottom
the point where the value of the function $F$ is minimal.




**[Read Linear and Nonlinear Programming by David G. Luenberger, Yinyu Ye]**

**[WikiPedia]**

# Gradient Descent Algorithm

***#python code***

# find a local minimum of the function $f(x)=x^4-3x^3+2$ , with derivative $f'(x)=4x^3-9x^2$.

```
# From calculation, we expect that the local minimum occurs at x=9/4
 xOld = 0
xNew = 6
# The algorithm starts at
x=6
eps = 0.01    # step size
precision = 0.00001

def    f_prime(x):
     return 4 * x**3 - 9 * x**2

while abs(xNew - xOld) > precision:
   xOld = xNew
  xNew = xNew - eps * f_prime(xNew)

print "Local minimum occurs at", xNew
```

**Homework:** find a local minimum of the function $f(x)=6x^5-8x^2+6$ using your favourite programming language! Plot the function and comment on boundedness.
Be careful about initial value? Why?
Prove that the candidate optimum, x*, is a maximum or minimum using f''(x*); recall if f''(x*)< 0 then **local max, else** f''(x*)> 0 then **local min**
**Optional Homework:** Is the function $f(x)=6x^5-8x^2+6$ a convex or concave function? Recall that if $f''(x) < 0$ forall $x$ then $f$ is **concave**; and if $f''(x) > 0$ then $f(x)$ is **convex**. Note: $f''(x)$ is the second derivative of $f$

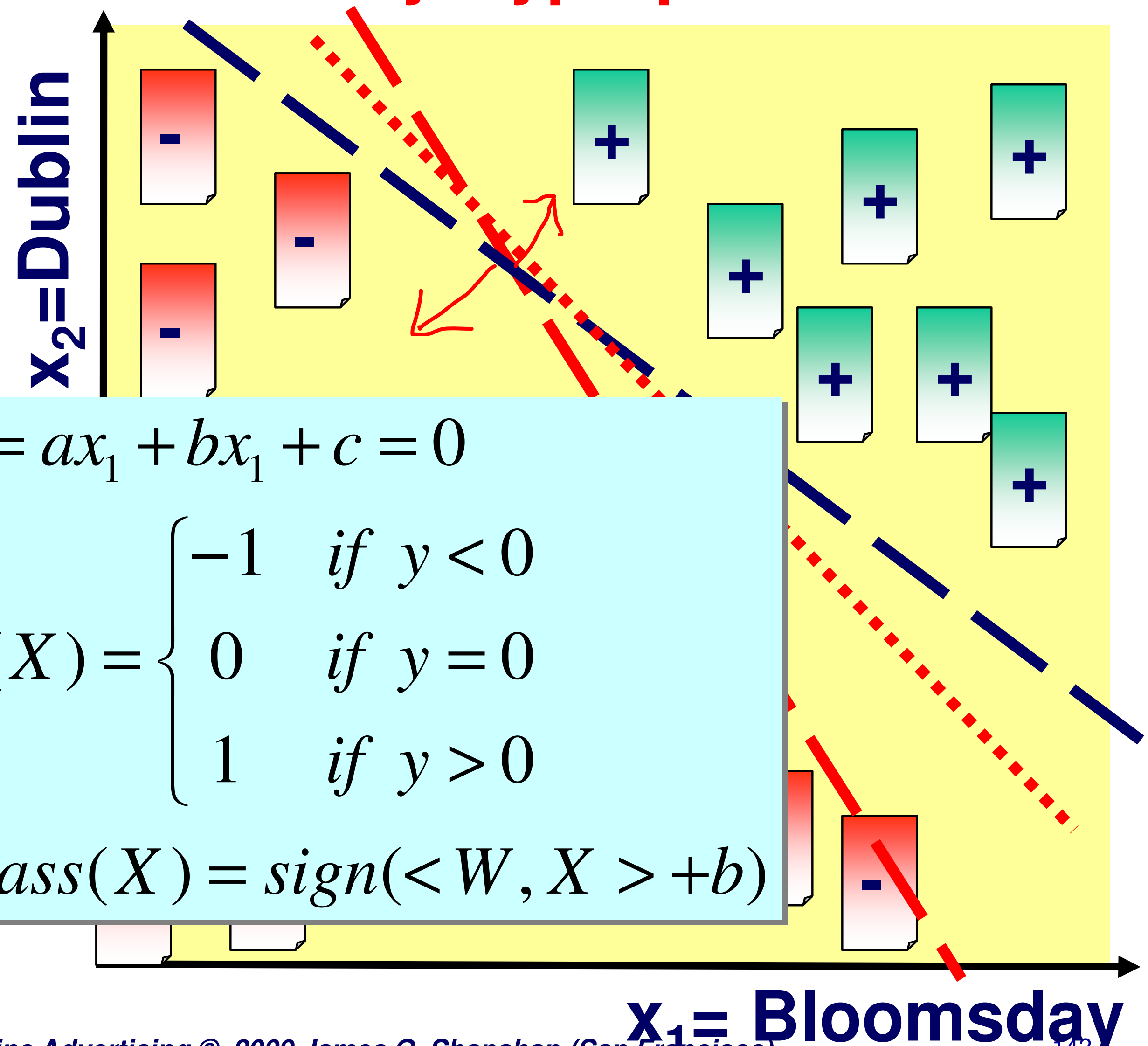
With this precision, the algorithm converges to a local minimum of 2.24996 in 70 iterations. A more robust implementation of the algorithm would also check whether the function value indeed decreases at every iteration and would make the step size smaller otherwise. One can also use an adaptive step size which may make the algorithm converge faster.
[http://en.wikipedia.org/wiki/Gradient_descent]

# Linear Discriminant Model

A linear discriminant function is linear in the components of X

E.g., $y = ax_1 + bx_2 + c$

## Many Hyperplanes Exist




$$y = ax_1 + bx_1 + c = 0$$

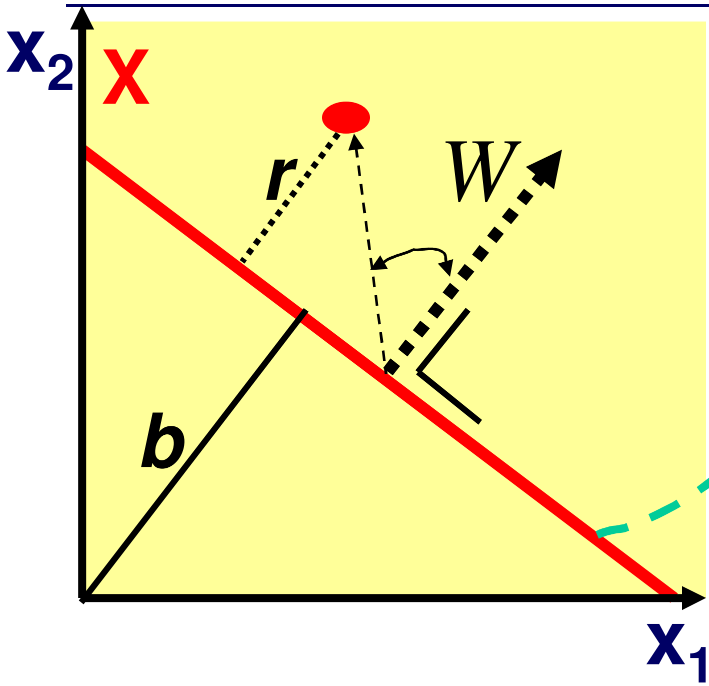
$$f(X) = \begin{cases} -1 & if \;\; y < 0 \\ 0 & if \;\; y = 0 \\ 1 & if \;\; y > 0 \end{cases}$$

$$Class(X) = sign(<W, X> + b)$$

## Training Data

| E.g. | $x_1$ | $x_2$ | y |
|---|---|---|---|
| 1 | 3 | 0 | -1 |
| 2 | | | +1 |
| … | … | … | … |
| L | 0 | 4 | -1 |

# Geometry: Linear Separators




$$w_1 x_1 + w_2 x_2 + b = 0$$ Basic hyperplane

$$\left(\sum_{i=1}^{n} w_i x_i\right) + b = 0$$ More general

$$\langle W, X \rangle + b = 0$$ Dot Product

For a plane given by the equation $ax + by + cz = d$, the vector $(a,b,c)$ is a normal.

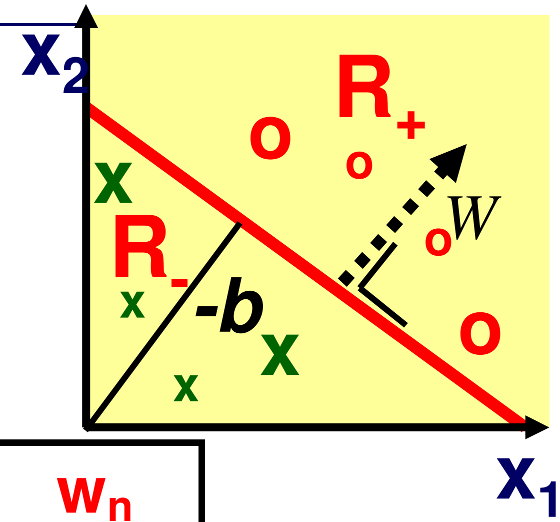
**Represent a hyperplane, H, in terms of vector W, and scalar b**

***W*** **determines the orientation of the hyperplane/discriminant plane**

***b*** **denotes the offset (Perpendicular distance) from the plane to the origin**

$$r = W^T X + b$$ **Perpendicular distance from point X to a hyperplane**

# Learning Linear Discriminants




**Primal learning (e.g., perceptron) involves learning weight values associated with term/feature.**

| Wgt Vector | $w_0$ | $w_1$ | … | …. | $w_n$ |
|---|---|---|---|---|---|

| Instance\Attr | $x_0$ | $x_1$ | $x_2$ | … | $x_n$ | y |
|---|---|---|---|---|---|---|
| 1 | 1 | 3 | 0 | .. | 7 | -1 |
| 2 | 1 | | | | | +1 |
| … | … | … | … | … | … | … |
| L | 1 | 0 | 4 | … | 8 | -1 |

# Augmented Representations

**Hyperplane as an Augmented weight vector**

**Augmented Data vector**

**Drop bias term**

$$W = \begin{bmatrix} w_0 \\ w_1 \\ .. \\ w_n \end{bmatrix} = \begin{bmatrix} b \\ w_1 \\ .. \\ w_n \end{bmatrix} \qquad X = \begin{bmatrix} \\ x_1 \\ .. \\ x_n \end{bmatrix} = \begin{bmatrix} 1 \\ x_1 \\ .. \\ x_n \end{bmatrix}$$

$$\boxed{Class\,(X) = sign(\langle W, X \rangle + b)}$$

**Classification rule simplifies**

$$\boxed{Class\,(X) = sign(\langle W, X \rangle)}$$

# Learning Linear Separators

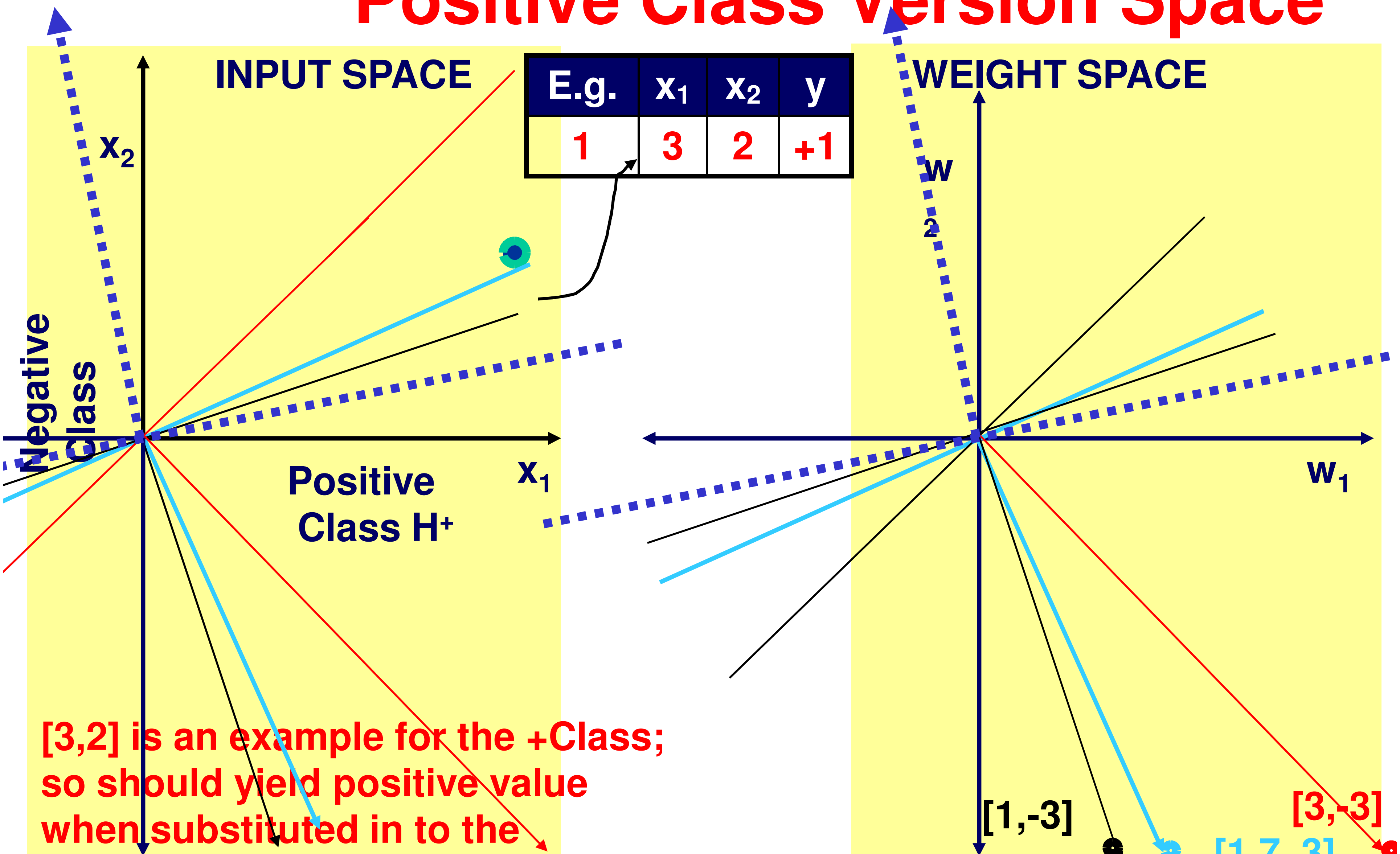
- **Linear discriminant functions have a variety of pleasant analytical and pedagogical properties!!**
- **Formulate the learning of a linear discriminant function as a problem of minimizing a criterion function**
  - E.g., training error
- **Learning corresponds to finding a weight vector**
  - A weight vector is can be thought of as a point in weight space (version space).
  - Each training example places a constraint on the possible location of a solution vector (feasible region)

# Version Space

- A version space in concept learning or induction is the subset of all hypotheses that are consistent with the observed training examples [Mitchell 1997].
- This set contains all hypotheses that have not been eliminated as a result of being in conflict with observed data.

# Positive Class Version Space

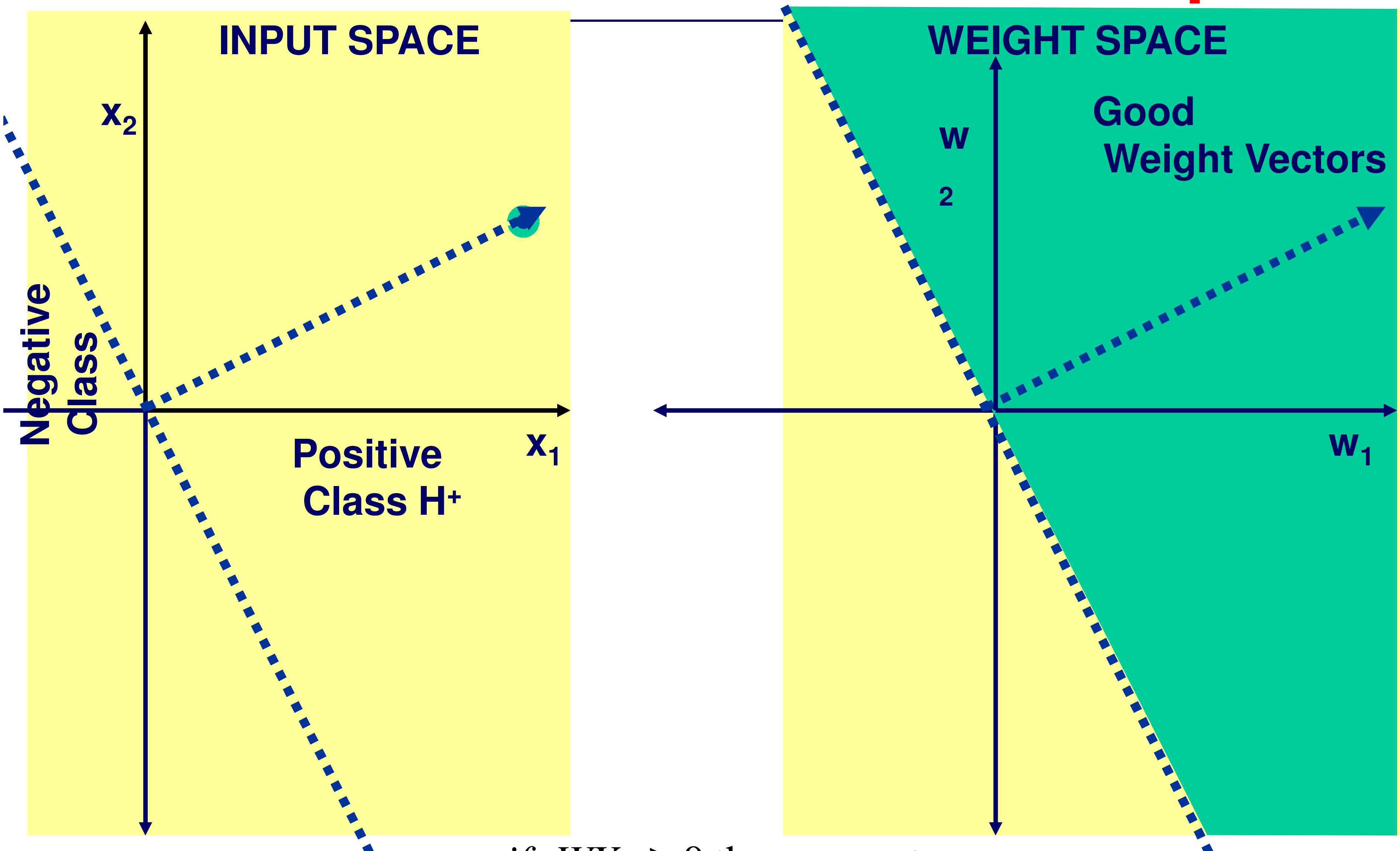
| E.g. | $x_1$ | $x_2$ | y |
|---|---|---|---|
| 1 | 3 | 2 | +1 |




[3,2] is an example for the +Class; so should yield positive value when substituted in to the equation of the hyperplane

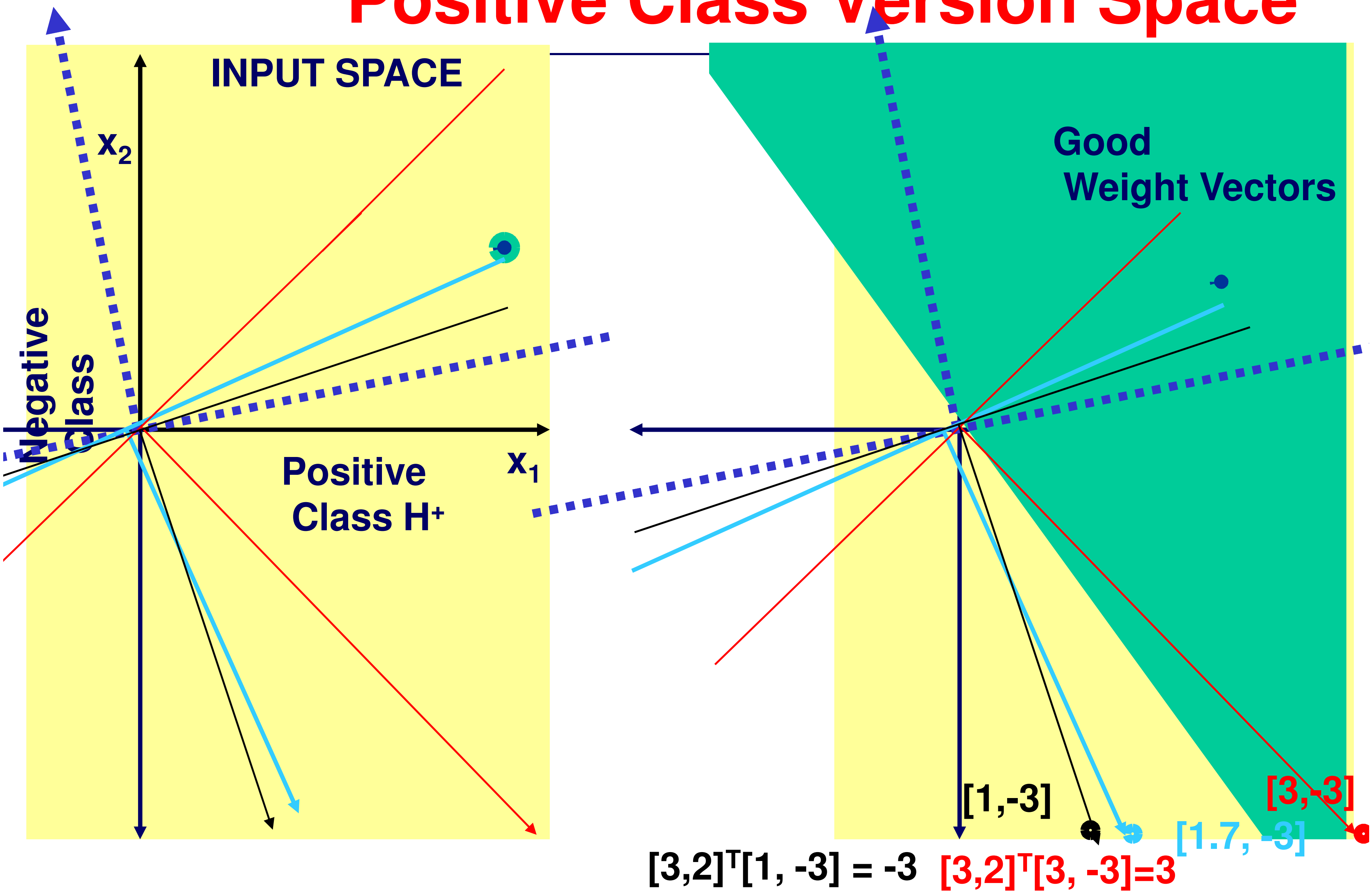
$[3,2]^T[1, -3] = -3$ $[3,2]^T[3, -3]=3$

# Positive Class Version Space

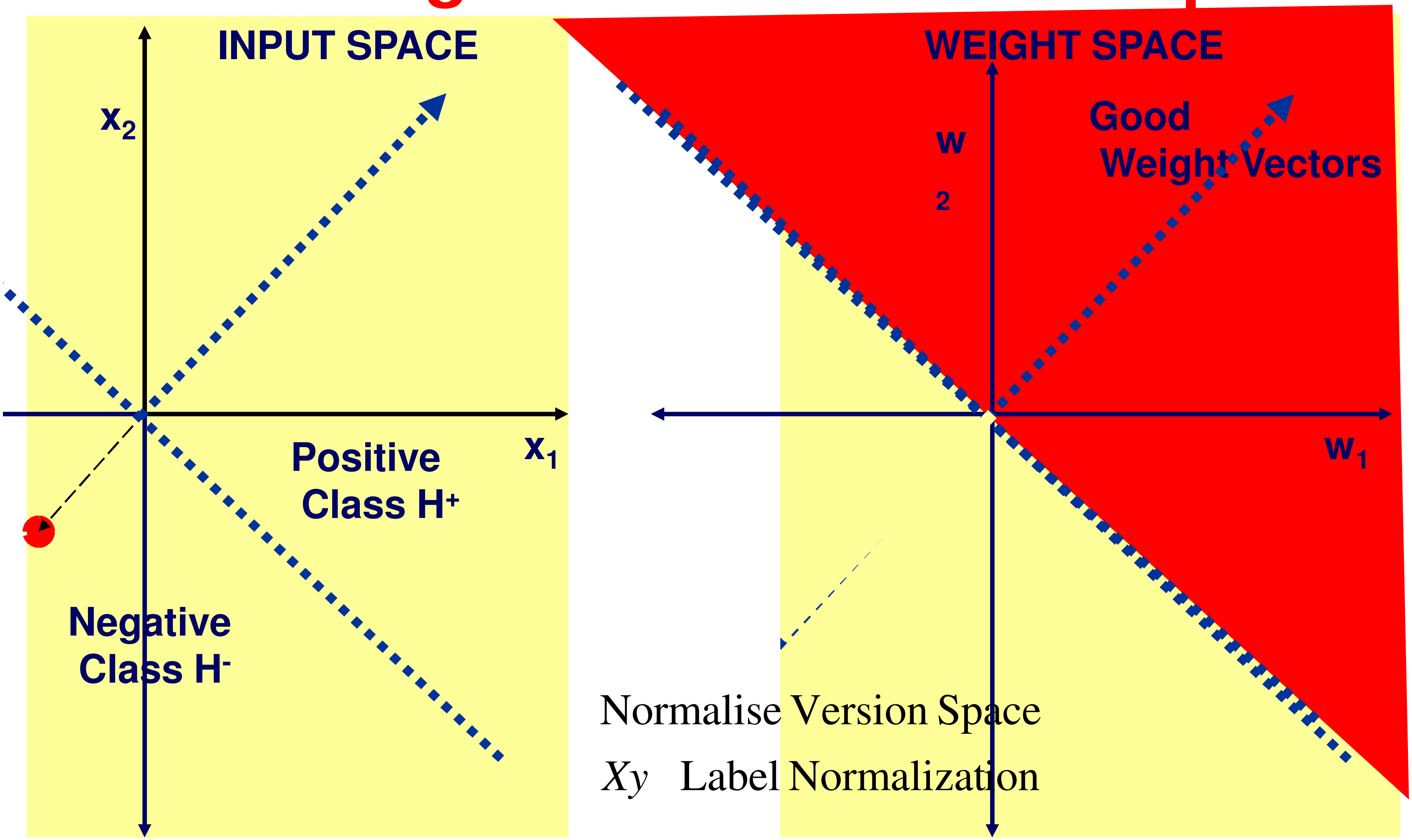



$$if \quad WXy \geq 0 \text{ then correct}$$

# Positive Class Version Space



# Negative Class Version Space

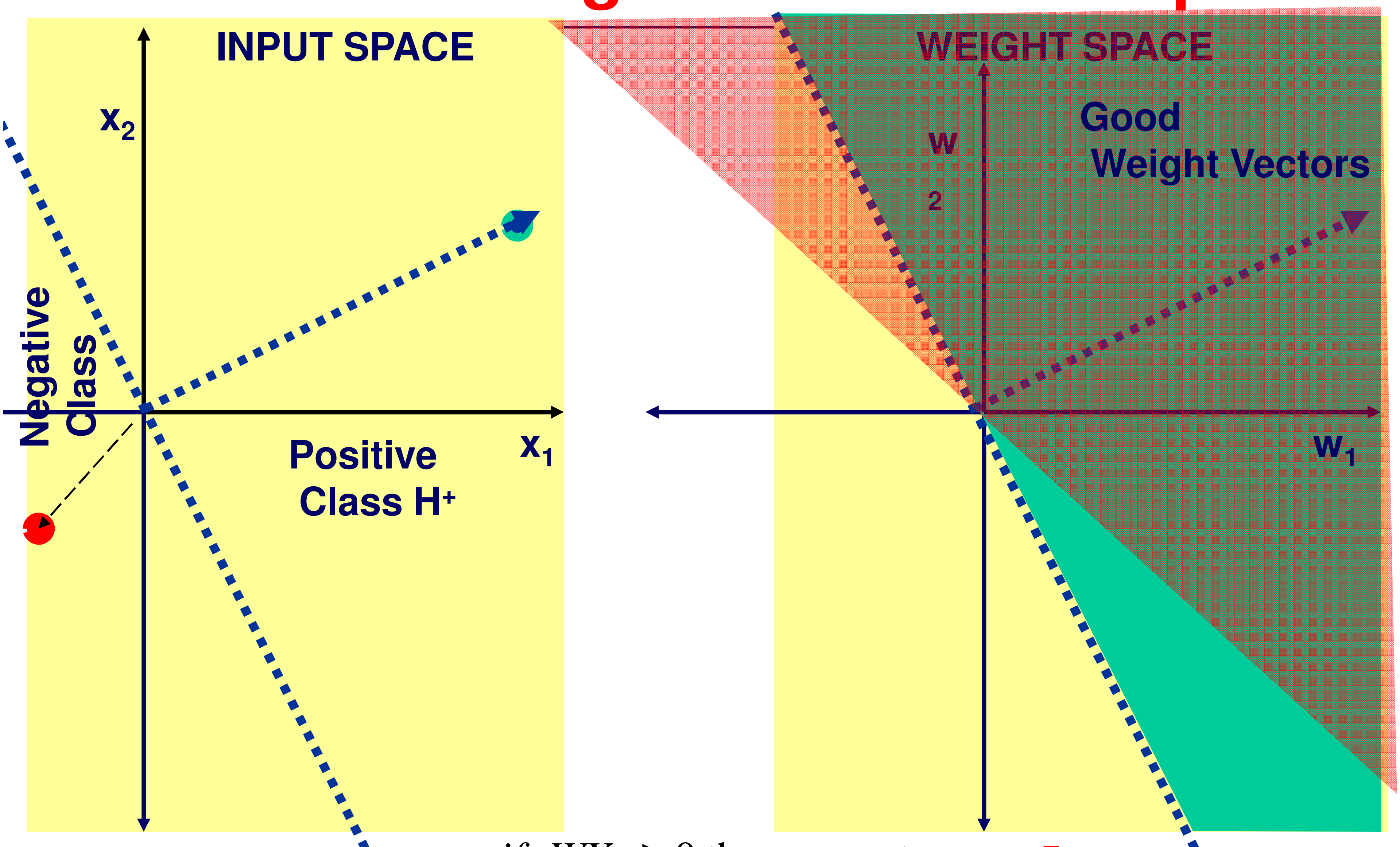



Normalise Version Space

$Xy$ Label Normalization

$if \quad WXy \geq 0$ then correct

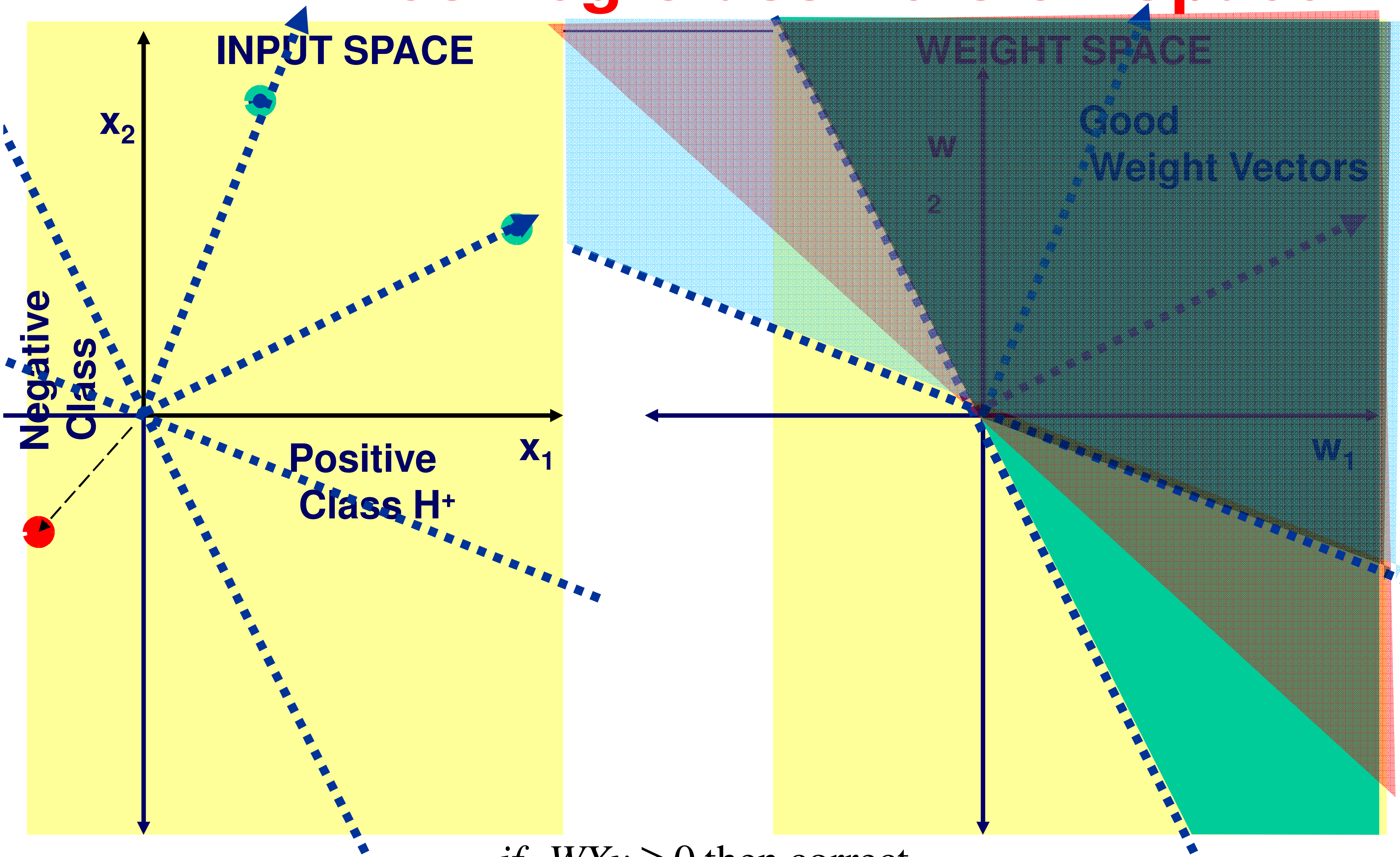
# Pos/Neg Class Version Space




$$if \quad WXy \geq 0 \text{ then correct}$$

**$[3,2]^T[3, -3]=-3$**

# Pos/Neg Class Version Space

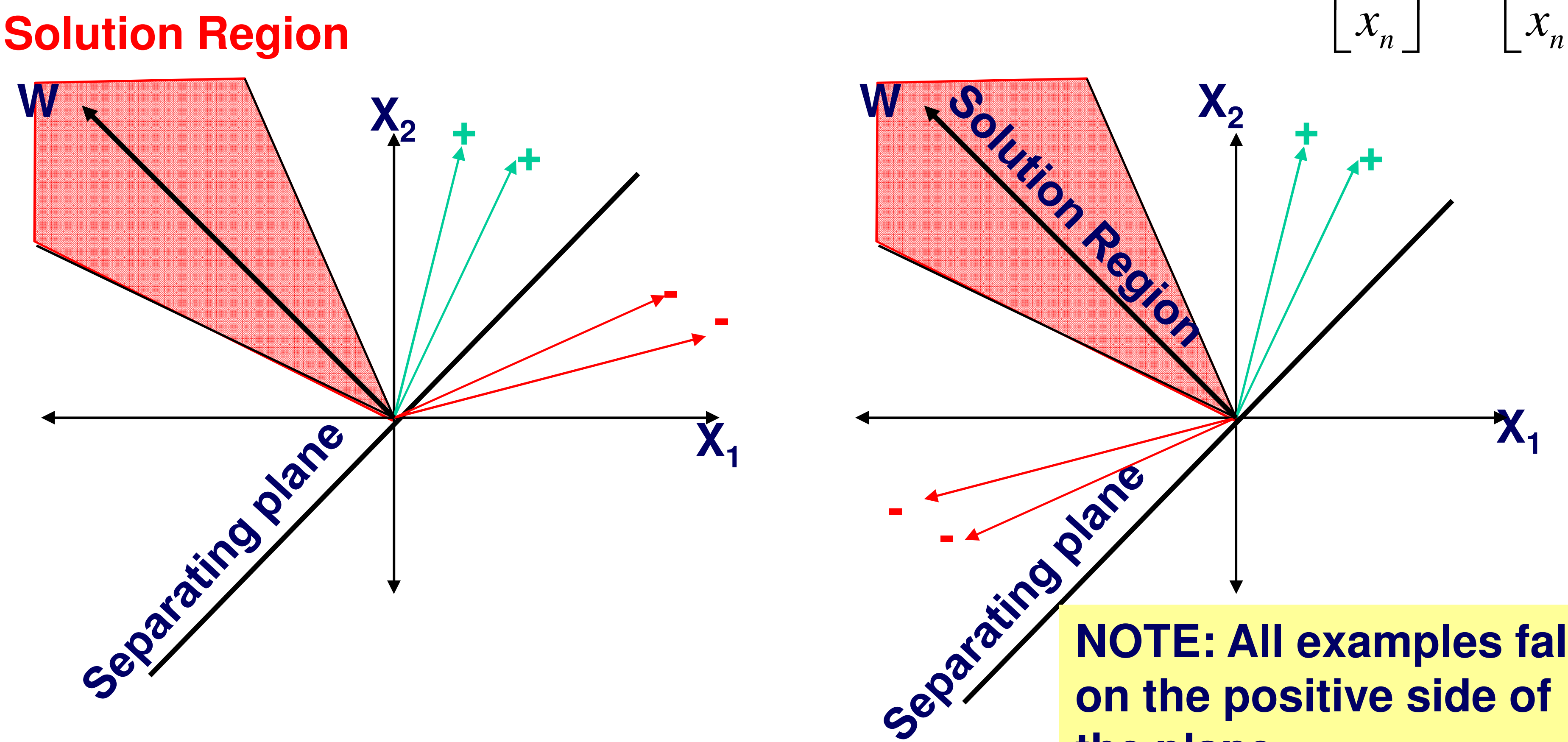



$$if \quad WXy \geq 0 \text{ then correct}$$

# Label Normalization

**Label Normalized Vector**

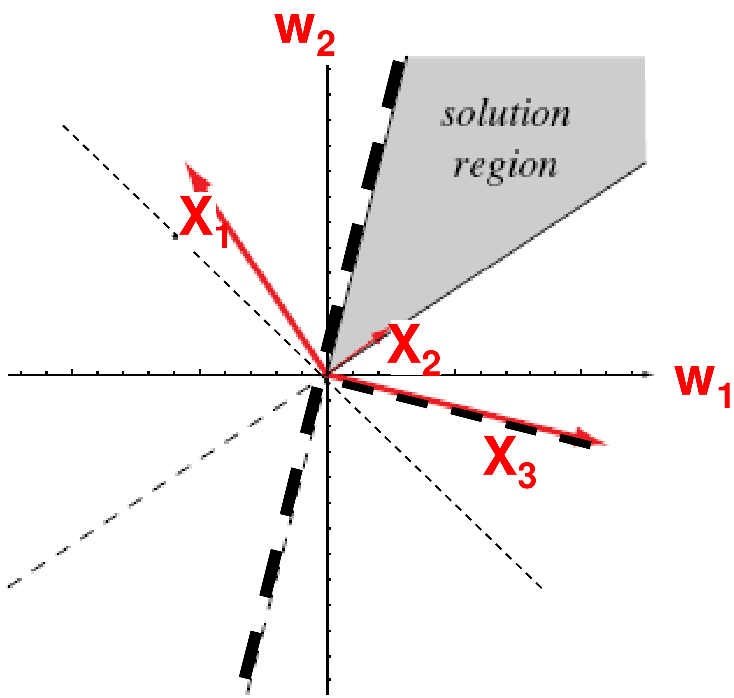
$$Xy = \begin{bmatrix} x_1 \\ x_2 \\ .. \\ x_n \end{bmatrix} y = \begin{bmatrix} x_1 y \\ x_2 y \\ .. \\ x_n y \end{bmatrix}$$




**NOTE: All examples fall on the positive side of the plane**

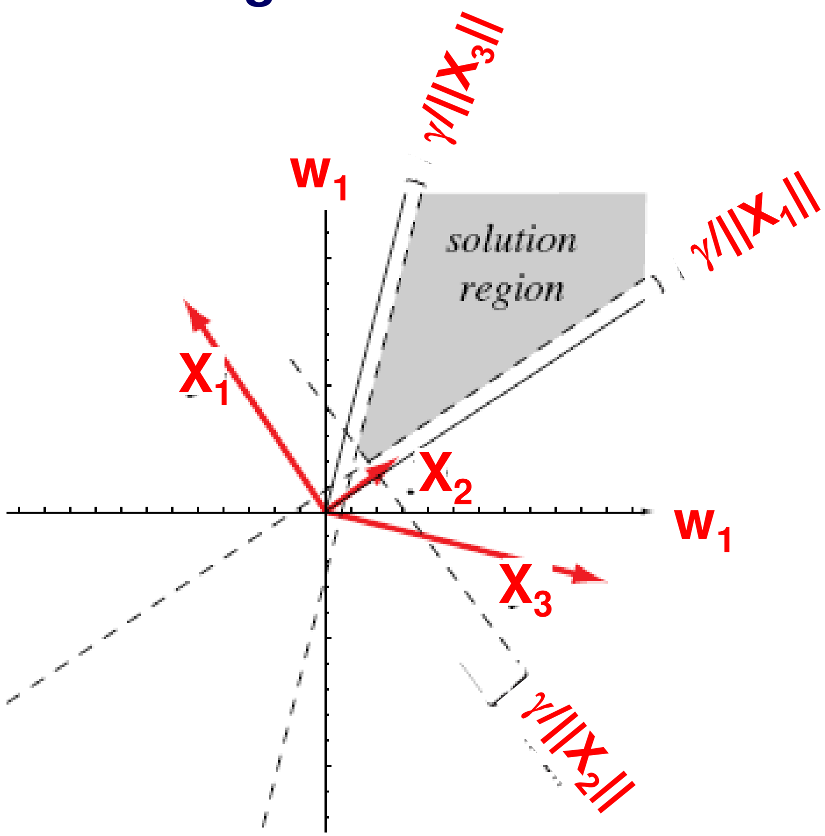
# Version Space with add$^{al.}$ constraints

**Get every example on the right side of the tracks**




$$\langle W, X_i \rangle y_i \geq 0 \; \forall i$$

**Get every example WELL INSIDE the right side of the tracks**



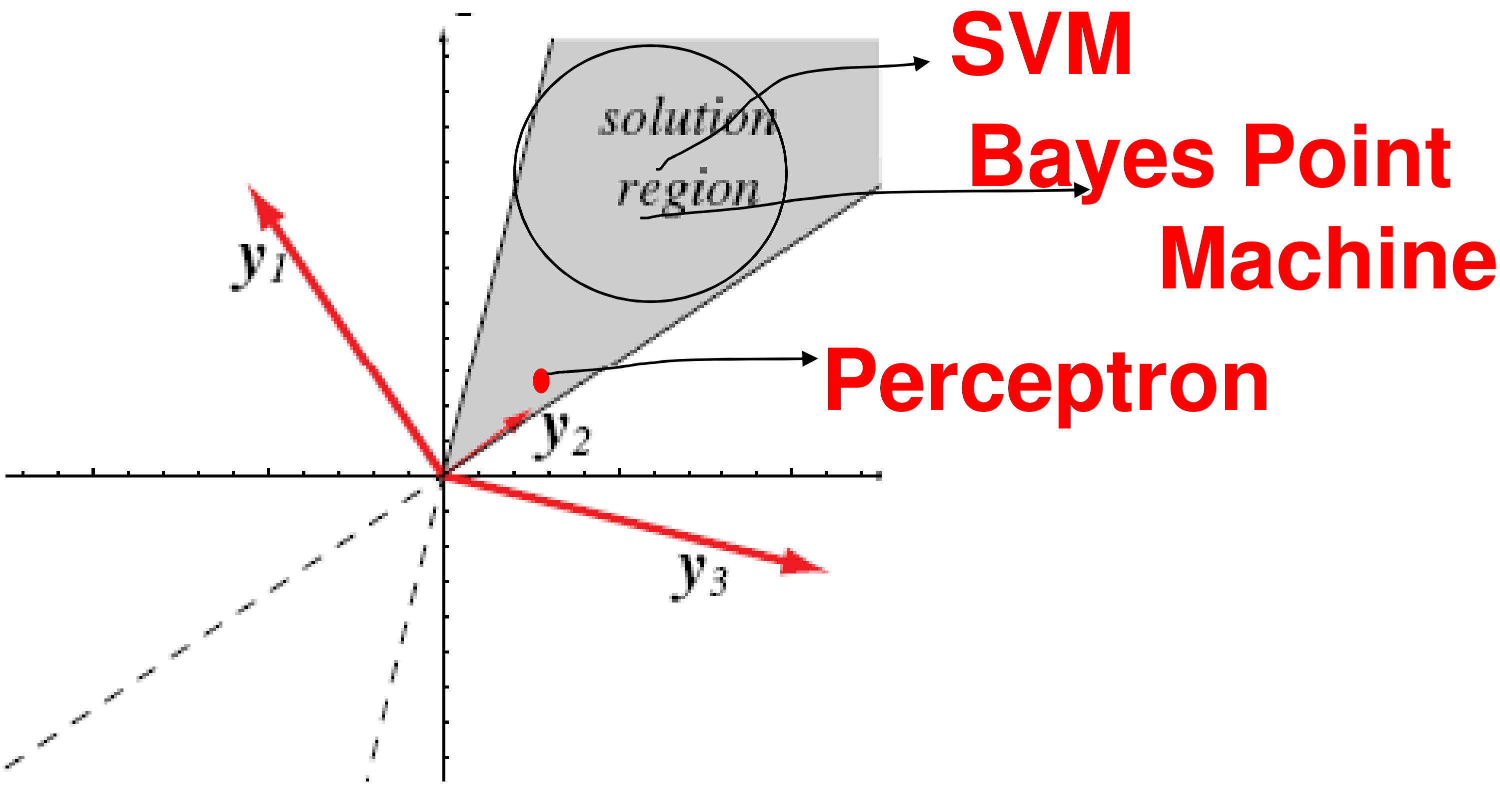

$$\langle W, X_i \rangle y_i \geq \gamma > 0 \; \forall i$$

[Adapted from Duda, Hart, Stork, 2001]

# Weight Vector and Solution Region

- **The hyperplane weight vector, *W*, can be thought of as specifying a point in the weight/version space**
- **Each example places a constraint on *W***
  - $(<W, X_i>)y_i > 0$
- **The solution hyperplane must be on the positive side of each data induced hyperplane**
- **Solution region = the intersection of *L* half-spaces**
- **Impose additional constraints**
  - Find solution that is in the middle of the solution region (i.e., that is insulated from data anomalies)
  - Maximize the minimum distance from the training examples to the separating hyperplane
    - $(<W, X_i>)y_i > \gamma$
  - $\gamma$ is known as the classifier margin

# Learning Algorithms in Version Space




**SVMs find the center of the largest radius hypersphere whose center can be placed in version space and whose surface does not intersect with the hyperplanes corresponding to the labeled instances.**

# Learning a Weight Vector

- **Q: Find a solution to a set of linear inequalities *((<W, Xi>)yi ≥ 0)***
  - Each example acts as a constraint
  - $(\langle W, X_i\rangle)y_i \geq 0$
- **A: Define an objective/criteria function**
  - That is minimized if *W* is a solution vector
  - Simple objective function *J(W)* is the number of mistakes made by *W*
    - *(when 0 then W is a solution)*.
  - Minimize this scalar function *J(W)* using gradient descent procedures

# Gradient Descent

- **To find a solution to the set of linear inequalities $\langle W, X_i \rangle y_i > 0$;**
- **We define a criterion function J(W) that is minimized if *W* is a solution.**
- **This kind of problem can be solved by gradient descent.**
- **General approach**
  - Start with some vector W(1).
  - Generate then W(2) by taking a small step in the direction of *the steepest descent, i.e., "$-\nabla J(W(k))$"*

# Objective Functions

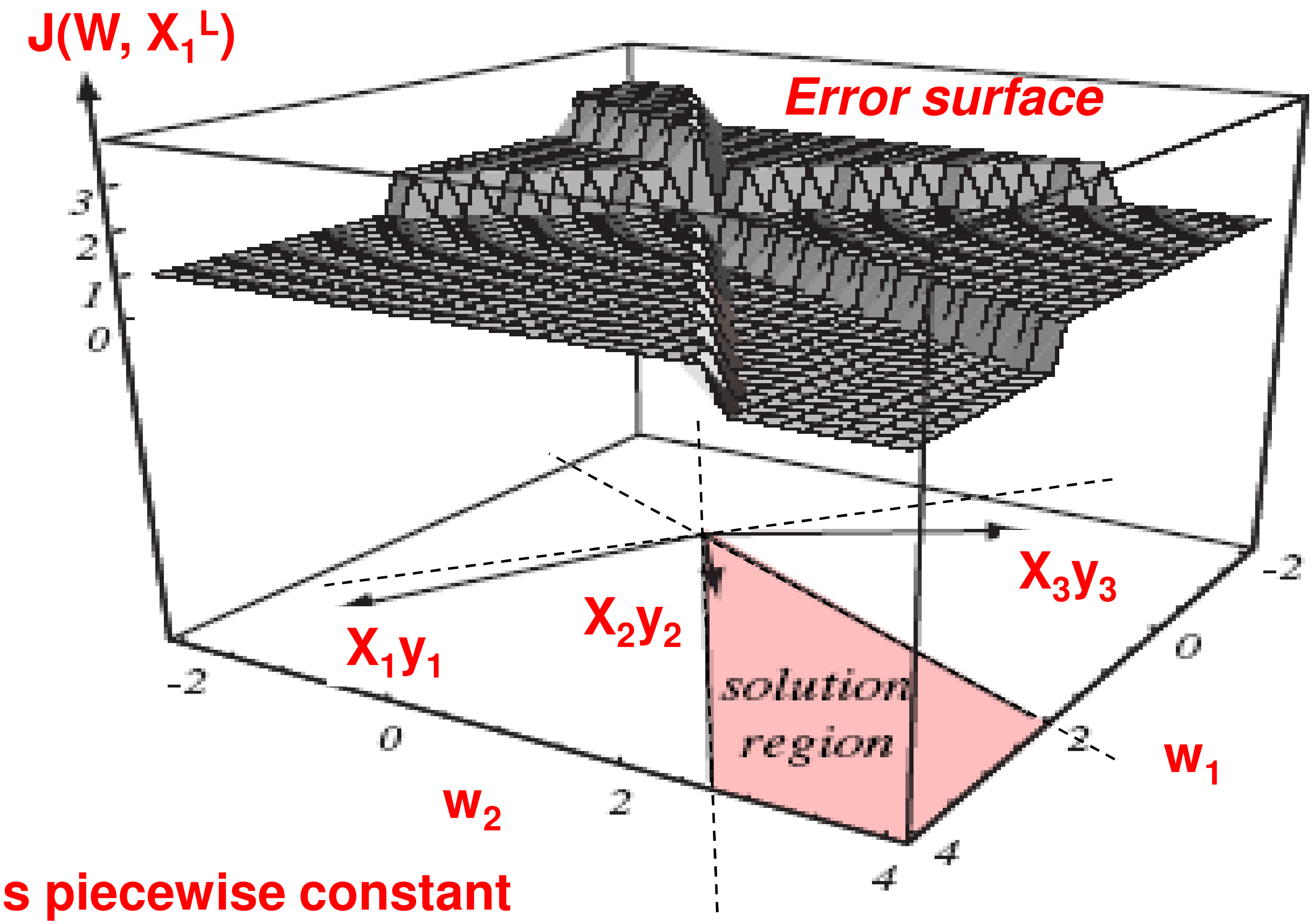
- **Consider the problem of finding a weight vector that satisfies all the training data**
  - $(\langle W, X_i \rangle) y_i > 0$
- **An obvious choice of objective**
  - Let $J(W, X_1, \ldots X_L)$ be the number of examples that are misclassified

$$J(W, X_1^L) = \sum_{\{X_i | y_i \langle W, X_i \rangle < 0\}} y_i y_i$$

# Objective Function: Number of Errors

## $J(W, X_1^L)$ = Number of Errors

| E.g. | $x_1$ | $x_2$ | y |
|---|---|---|---|
| 1 | .. | .. | .. |
| 2 | .. | .. | .. |
| 3 | .. | .. | .. |




**3 examples (label normalized, i.e., Xy)**
**⇒Legal region corresponds to the intersection of positive half-spaces**
**=> Max $J(W, X_1^L)$ =3**

**However, $J(W, X_1^L)$ is piecewise constant**
**=> Very poor candidate for gradient search**

**[Adapted from Duda, Hart, Stork, 2001]**

# Perceptron Objective Function

- **Given linear Constraints** $(\langle W, X_i \rangle) y_i > 0$
- $J(W, X_1, \ldots X_L)$ **or** $J(W, X_1^L)$
  - *The number of examples that are misclassified is not continuous*

$$J(W, X_1^L) = \sum_{\{X_i | y_i \langle W, X_i \rangle < 0\}} y_i y_i$$

- **However,** $J_P(W, X_1^L)$, **the Perceptron Objective Function, is piecewise continuous**

$$J_P(W, X_1^L) = \sum_{\{X_i | y_i \langle W, X_i \rangle < 0\}} (-W^T X_i y_i)$$

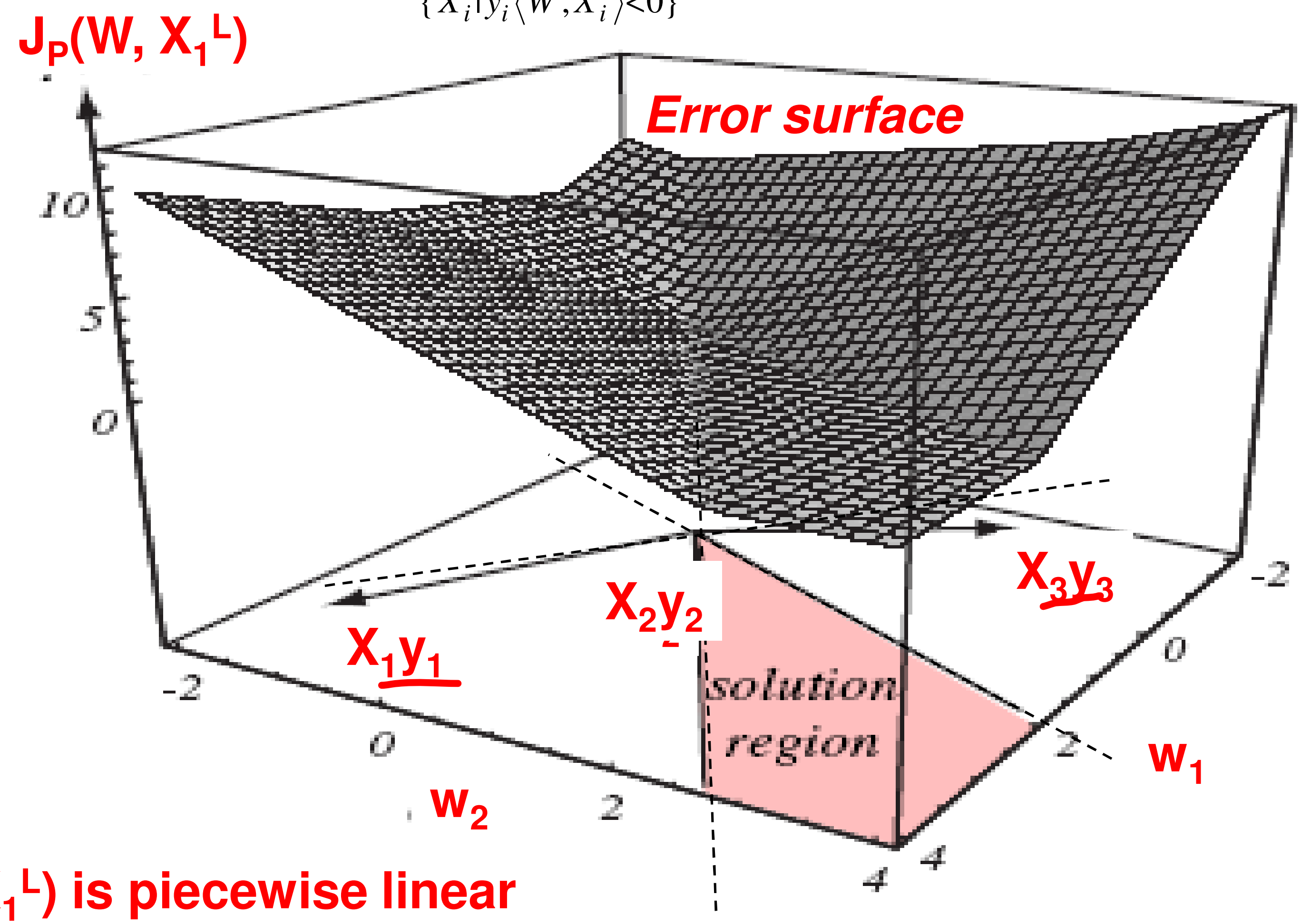
- **Solution Region**
  - If no examples are misclassified then $J_p$ is zero
  - $J_p$ is zero when $W$ is in the solution region
- **Intuitively,** $J_p$ **corresponds to sum of the margins (negative) of misclassified examples**

# Objective Function: Perceptron

$J_p(W, X_1^L)$

| E.g. | $x_1$ | $x_2$ | y |
|---|---|---|---|
| 1 | .. | .. | .. |
| 2 | .. | .. | .. |
| 3 | .. | .. | .. |

$$J_P(W, X_1^L) = \sum_{\{X_i \mid y_i \langle W, X_i \rangle < 0\}} (-W^T X_i y_i)$$




However, $J_p(W, X_1^L)$ is piecewise linear
=> Acceptable for gradient search

[Adapted from Duda, Hart, Stork, 2001]

# Alternative Objective Function

1. ***$J(W, X_1, \ldots X_L)$ or $J(W, X_1^L)$ be the number of examples that are misclassified (Non continuous)***

$$J(W, X_1^L) = \sum_{\{X_i | y_i \langle W, X_i \rangle < 0\}} y_i y_i$$

2. **$J_P(W, X_1^L)$, Perceptron Objective**

$$J_P(W, X_1^L) = \sum_{\{X_i | y_i \langle W, X_i \rangle < 0\}} (-W^T X_i y_i)$$

3. **$J_q(W, X_1^L)$, Squared/quadratic Error (too smooth; converge to boundary point; dominated by longest example vectors)**

$$J_q(W, X_1^L) = \sum_{\{X_i | y_i \langle W, X_i \rangle < 0\}} (-W^T X_i y_i)^2$$

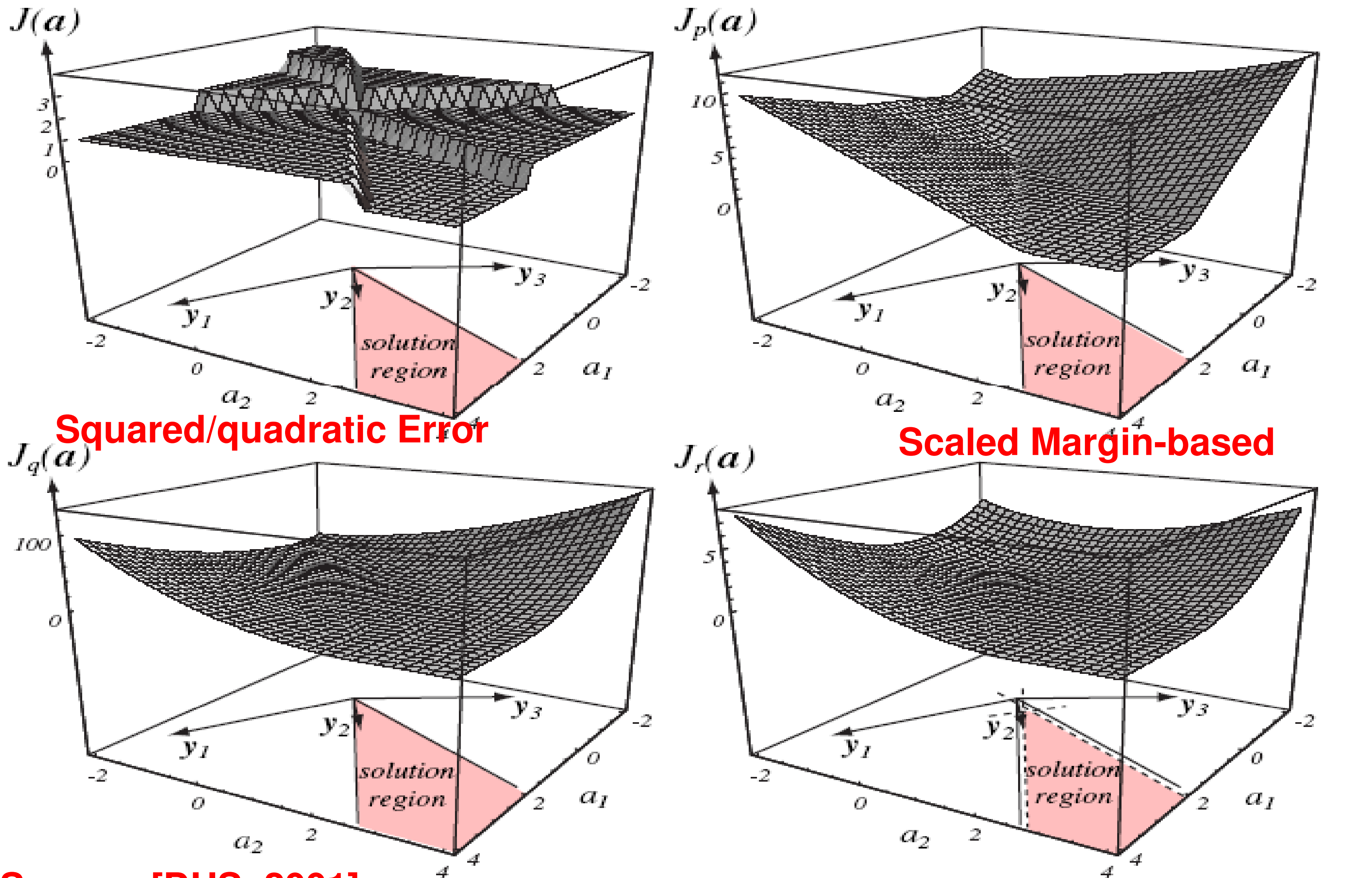
4. **$J_r(W, X_1^L)$, Scaled Margin-based**

$$J_r(W, X_1^L) = \sum_{\{X_i | y_i \langle W, X_i \rangle < \gamma\}} \frac{(W^T X_i y_i - \gamma)^2}{\|X_i\|^2}$$

# Different Objective Functions

***Number misclassified examples*** **Perceptron Objective**

**Squared/quadratic Error** **Scaled Margin-based**




**Source: [DHS, 2001]**

# Perceptron using Gradient Descent

**BATCH Update Rule**

- General Update Rule
  - **W(k+1)=W(k) - $\eta\nabla$J (W(k))**

$$J_P(W, X_1^L) = \sum_{\{X_i | y_i \langle W, X_i \rangle < 0\}} (-W^T X_i y_i)$$

**Perceptron Objective Function**

$$\nabla J_P = \frac{\partial (J_P(W, X_1^L))}{\partial W} = \sum_{\{X_i | y_i \langle W, X_i \rangle < 0\}} (-X_i y_i)$$

**Gradient of Perc. Objective Func.**

$$W(K+1) = W(k) + \eta(k) \sum_{\{X_i | y_i \langle W, X_i \rangle < 0\}} (X_i y_i)$$

**Perceptron BATCH Update Rule**

**Intuitively, drag weight vector closer to the misclassified examples**

# Perceptron using Gradient Descent

## Single Update Rule

- General Update Rule
  - **W(k+1)=W(k) - η∇J (W(k))**

$$J_P(W, X_1^L) = \sum_{\{X_i | y_i \langle W, X_i \rangle < 0\}} (-W^T X_i y_i)$$

**Perceptron Objective Function**

$$\nabla J_P = \frac{\partial (J_P(W, X_1^L))}{\partial w} = \sum_{\{X_i | y_i \langle W, X_i \rangle < 0\}} (-X_i y_i)$$

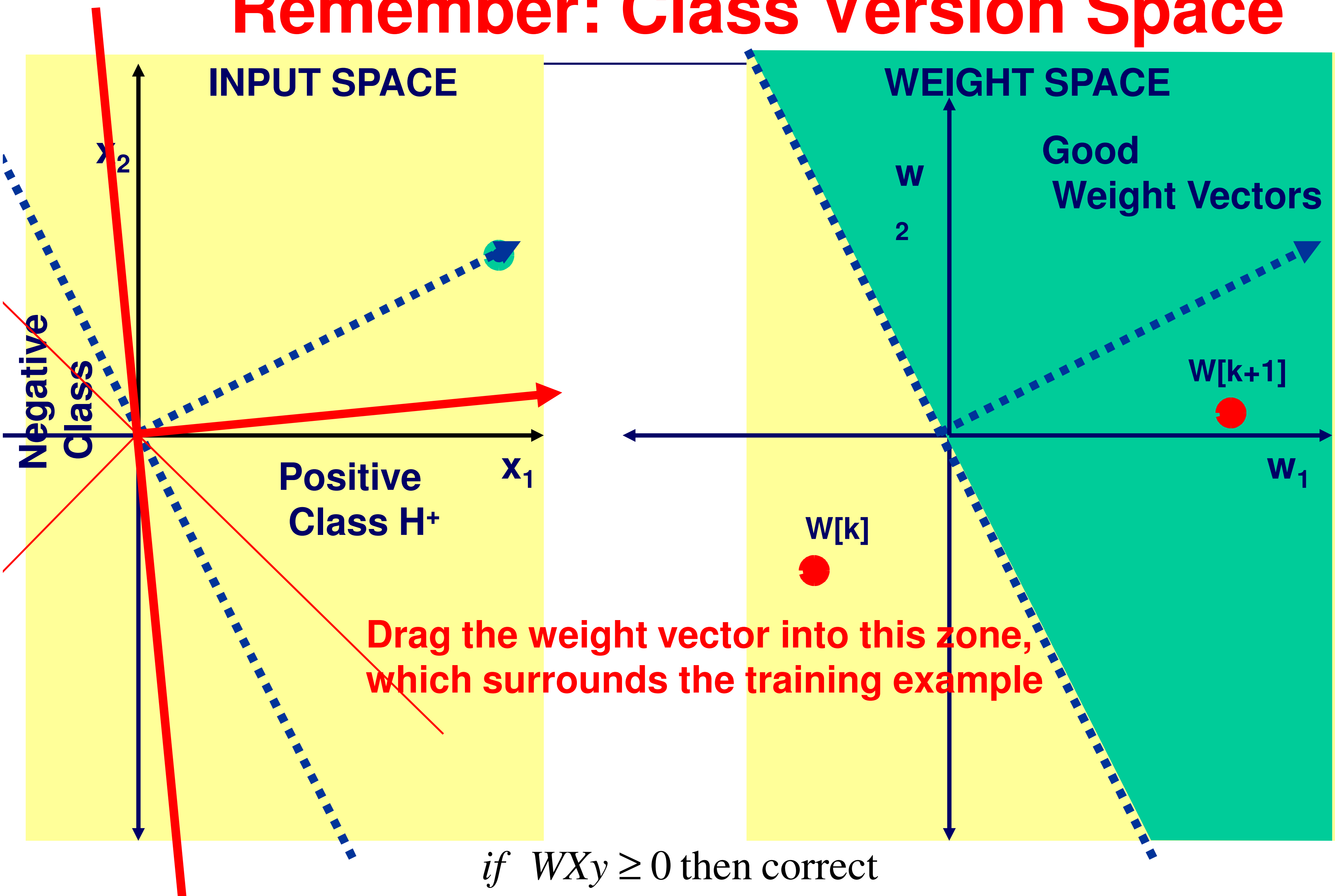
**Gradient**

$$W(K+1) = W(k) + \eta(k) X y_i$$

**Perceptron SINGLE Update Rule**

**Intuitively, drag weight vector closer to the misclassified example**

# Remember: Class Version Space




$$if \quad WXy \geq 0 \text{ then correct}$$

# Perceptron Algorithm

## Single-sample Primal Forn

- **Given Training data S where each example *i* is of the form $(x_{i,1},\ldots, x_{i,n},y_i)$, and a learning rate $\eta$**
- **Set $W_o$ to zeros; $k$=0;**
- **Repeat**
  - For i = 1 to *|Train|* do

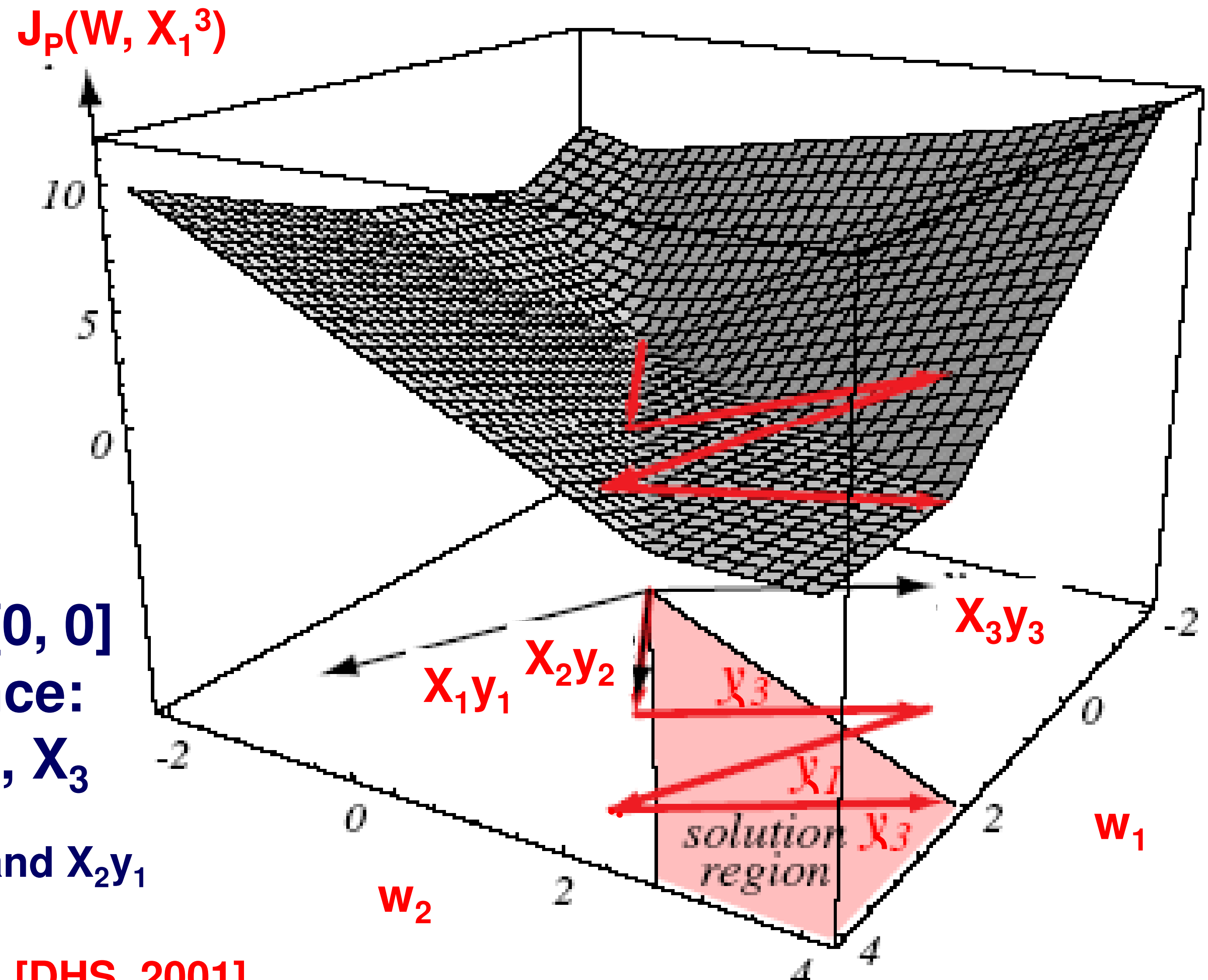
    If $(y_i(<W_{k,} X_i>+b_k)) \leq 0$ then // $y_i \neq$ Sgn $(<W_k X_i>+b_k)$ MISTAKE

    $W_{k+1}= W_k+ \eta\, y_i X_i$ // Update weights with example i

    $k= k + 1$ // *Update number of mistakes*

    End-If
  - End-For
- **Until no mistakes are made**
- **Return *k*, *W***

# Perceptron Update Example



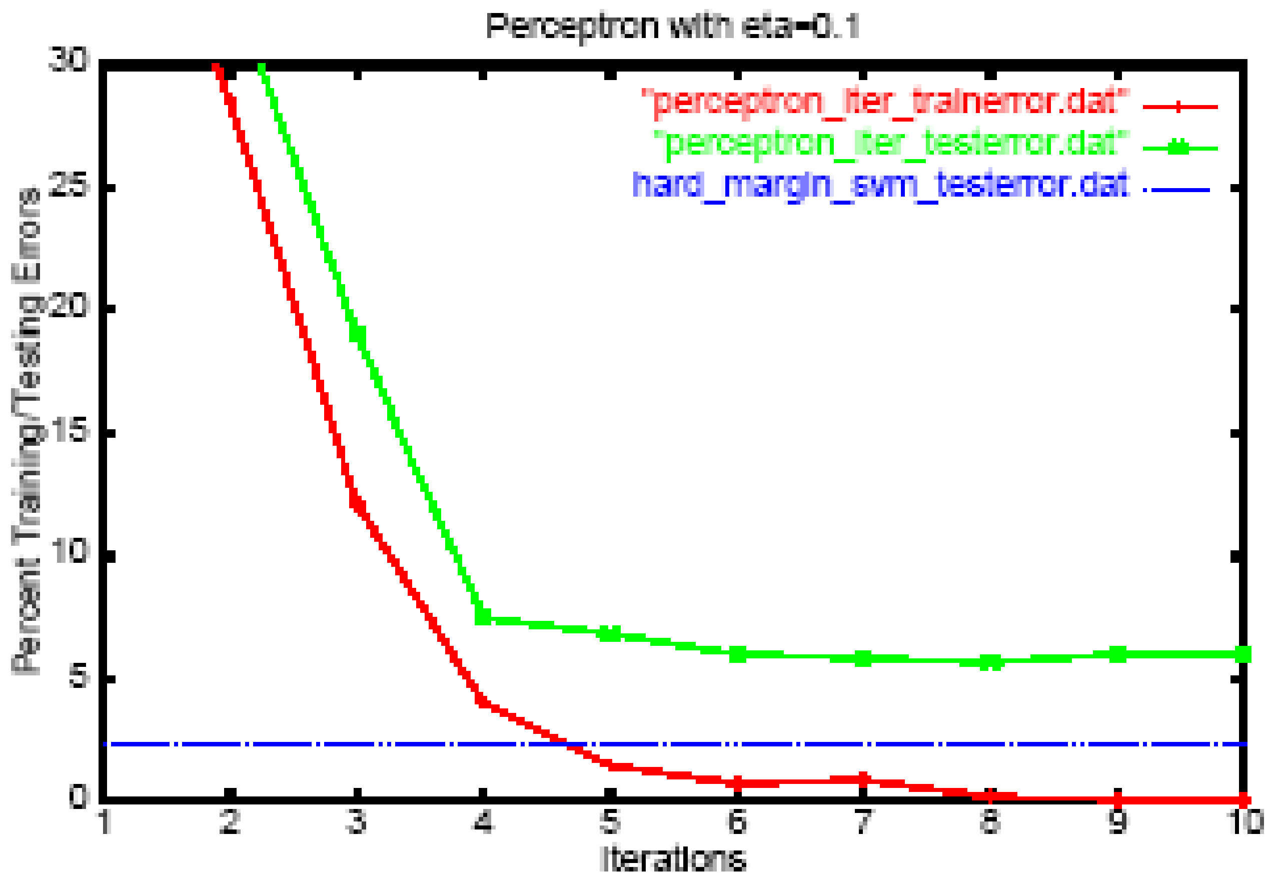

**Start with W = [0, 0]**
**Update Sequence:**
**$X_2$, $X_3$, $X_1$, $X_3$**

**Updating with $X_3y_3$ and $X_2y_1$ cause overshooting**

**Adapted from: [DHS, 2001]**

# Perceptron Learning: Text Example

## Experiment: Perceptron for Text Classification




Train on 1000 pos / 1000 neg examples for " acq" (Reuters-21578).

[Source: http://www.cs.cornell.edu/Courses/CS678/2003sp/slides/perceptron_4up.pdf]

RGui

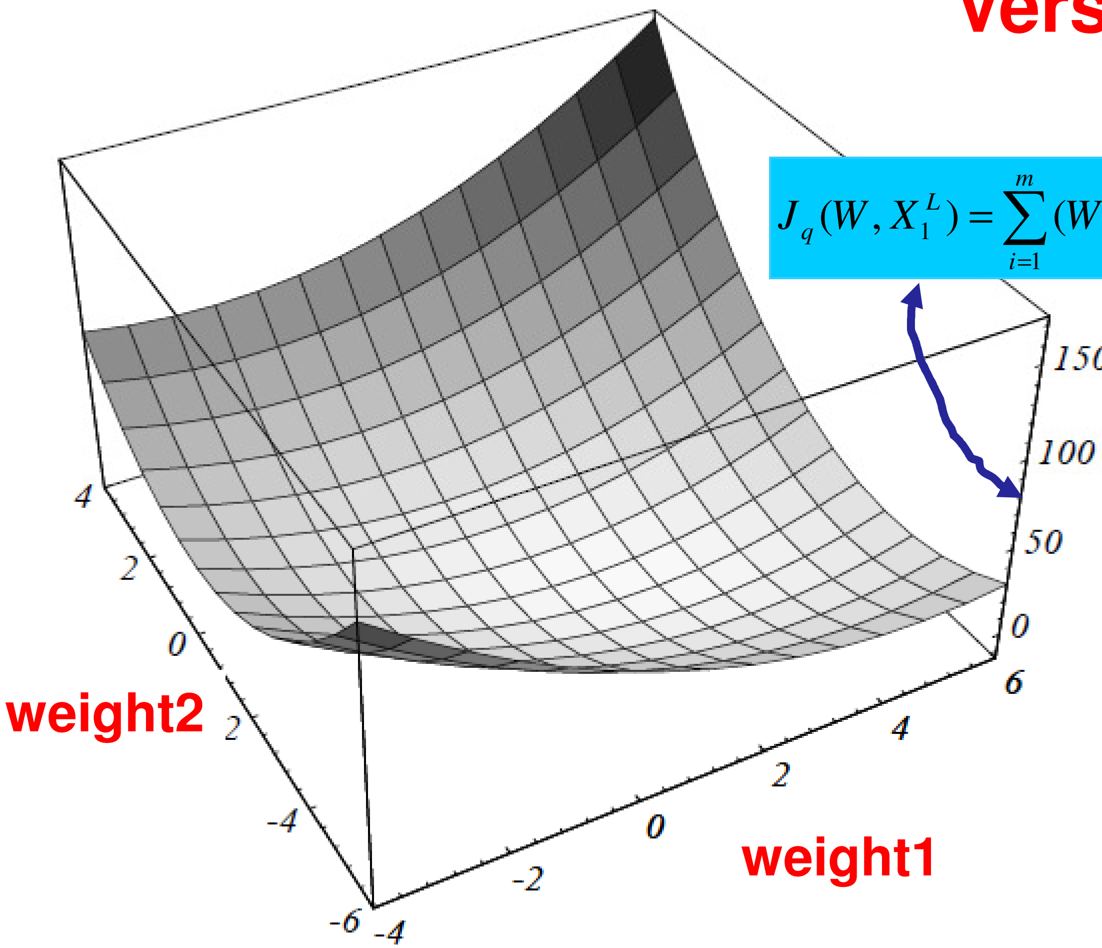
# Perceptron learning

```
plotLinearModel=function(X, w,b){
        margins=classify.linear(X, w, b)
        labels=ifelse(margins>0,1,-1)
        plot(X,pch=ifelse(margins>0,"+","-"),xlim=c(-1,1),ylim=c(-1,1))
        abline(0.5,1)
        points(c(0,0),c(0,0),pch=19)
        lines(c(0,-0.25),c(0,0.25),lty=2)
        arrows(-0.3,0.2,-0.4,0.3)
        text(-0.45,0.35,"W /* weight vector */")

        #points(rnorm(200), rnorm(200), col = "red")
        grid()
        return(cbind(X, labels))
}
classify.linear = function(x,w,b) {
        distance.from.plane = function(z,w,b) { sum(z*w) + b }
        distances = apply(x, 1, distance.from.plane,w=w, b=b)
        return(ifelse(distances < 0, -1, +1))
}

classify.linear.1ex = function(x,w,b) {
        distances =sum(x*w) + b
        return(ifelse(distances < 0, -1, +1))
}
perceptron = function(x, y, learning.rate=1) {
        w = numeric(ncol(x)) # Initialize the parameters
        b = 0
        k = 0 # Keep track of how many mistakes we make
        R = max(euclidean.norm(x))
        #browser()
        made.mistake = TRUE # Initialized so we enter the while loop

        while (made.mistake) {
                made.mistake=FALSE # Presume that everything's OK
                for (i in 1:nrow(x)) {
                        if (y[i] != classify.linear.1ex(x[i,],w,b)) {
                                #browser();
                                w <- w + learning.rate * y[i]*x[i,]
                                b <- b + learning.rate * y[i]*R^2
                                k <- k+1
                                made.mistake=TRUE # Doesn't matter if already set to TRUE previously
                        slope=-1*(w[1]/w[2]);
                                b=-1*b/w[2]
                                #print(paste("slope is ",slope,"b is", b, sep=" "))
                                #abline(b, slope, col="red",lw=1)
                        }

                }

        }
        slope=-1*(w[1]/w[2]);
        b=-1*b/w[2]
        print(paste("slope is ",slope,"b is", b, sep=" "))
        abline(b, slope, col="blue",lw=3)

        return(w=w,b=b,mistakes.made=k)
}
euclidean.norm=function(X) {
        euclidean.norm1 = function(x) {sqrt(sum(x * x))}
        enorms = apply(X, 1, euclidean.norm1 )
        return(enorms)
}
```

```
## Driver code
x1=runif(5, -1, 1);  x2=runif(5, -1, 1)
#slope =1;
#y= slope*x1+b

X=cbind(x1=x1,x2=x2);  #x=runif(10, -1,1)

b=-0.5;   #y= 1*x+b

w=c(-1,1)
par(mfrow=c(1,1)) ## sets up screen for four plots

#trainingDataOld=trainingData
```

# Gradient Descent for Ordinary Least Squares

## Version space (weights)




$$J_q(W, X_1^L) = \sum_{i=1}^{m} (W^T X_i - y_i)^2$$

***Error surface; each point corresponds to a different linear model (hypothesis). The vertical axis indicates the squared error for the training dataset WRT that weight vector.***

***Q: Will this surface change for different datasets?***

**OLS with this objective has no local minima (convex as the Hessian, n by n matrix of second derivatives, of the objective function is positive definite); in this case n=2 variables.**

**Iterative versus closed form solution**

# OLS using Gradient Descent

## BATCH Update Rule

- General Update Rule
  - **W(k+1)=W(k) - η∇J (W(k))**

$$J_q(W, X_1^L) = \sum_{i=1}^{m} (W^T X_i - y_i)^2$$

**OLS Objective Function**

**True gradient is used to update the parameters of the model, corresponding to the sum of the gradients caused by each training example (one sweep)**

$$\nabla J_P = \frac{\partial (J_P(W, X_1^m))}{\partial W} = \sum_{i=1}^{m} (W^T X_i - y_i) X_i$$

**Gradient of OLS Objective Func.**

$$W(K+1) = W(k) + \eta(k) \sum_{i=1}^{m} (WX_i - y_i) X_i$$

**OLS BATCH Update Rule**

**Intuitively, drag weight vector closer to the misclassified examples**

# OLS using Gradient Descent

## Stochastic Gradient Descent
## Online/Single Update Rule

- General Update Rule
  - **W(k+1)=W(k) - η∇J (W(k))**

$$J_q(W, X_1^L) = \sum_{i=1}^{m} (W^T X_i - y_i)^2$$

**OLS Objective Function**

**True gradient is approximated the gradient of the cost function only evaluated at one example; adjust parameters proportional to this approx. gradient. This can be much better for large datasets.**
**E.g., Stochastic Gradient Decision Trees; perceptron**

$$W(k+1) = W(k) + \eta(k)(WX_i - y_i)X_i$$

**OLS Single Update Rule**

**Intuitively, drag weight vector closer to the misclassified example**

# Gradient Vector & Tangent Plane

▲ Note the similarity between the equation of a tangent plane and the equation of a tangent line:

$$y - y_0 = f'(x_0)(x - x_0)$$

**2** Suppose $f$ has continuous partial derivatives. An equation of the tangent plane to the surface $z = f(x, y)$ at the point $P(x_0, y_0, z_0)$ is

$$z - z_0 = f_x(x_0, y_0)(x - x_0) + f_y(x_0, y_0)(y - y_0)$$

**EXAMPLE 1** Find the tangent plane to the elliptic paraboloid $z = 2x^2 + y^2$ at the point (1, 1, 3).

**SOLUTION** Let $f(x, y) = 2x^2 + y^2$. Then

$$f_x(x, y) = 4x \qquad f_y(x, y) = 2y$$

$$f_x(1, 1) = 4 \qquad f_y(1, 1) = 2$$

**Calculate gradient vector by evaluating partial derivates at tangential point**
**Gradient vector at (1, 1) is (4, 2);**
**f'(1,1) = (4,2)**
**f(1,1)=3**

Then (2) gives the equation of the tangent plane at (1, 1, 3) as

$$z - 3 = 4(x - 1) + 2(y - 1)$$

or

$$z = 4x + 2y - 3$$

**Tangent plane at (1, 1, 3) with gradient (4,2)**

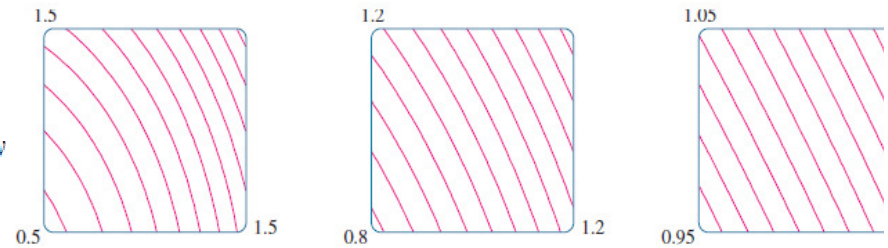



**FIGURE 3**
Zooming in toward (1, 1) on a contour map of $f(x, y) = 2x^2 + y^2$

Figure 2(a) shows the elliptic paraboloid and its tangent plane at (1, 1, 3) that we found in Example 1. In parts (b) and (c) we zoom in toward the point (1, 1, 3) by restricting the domain of the function $f(x, y) = 2x^2 + y^2$. Notice that the more we zoom in, the flatter the graph appears and the more it resembles its tangent plane.

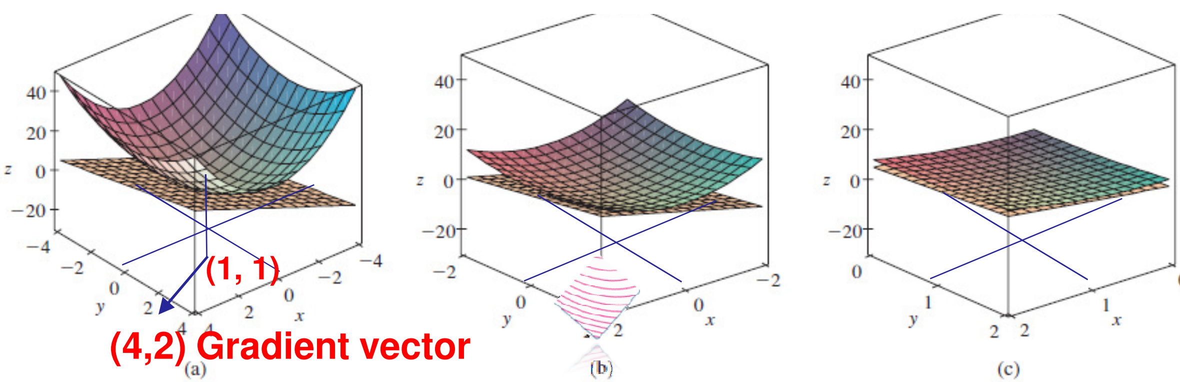



**FIGURE 2** The elliptic paraboloid $z = 2x^2 + y^2$ appears to coincide with its tangent plane as we zoom in toward (1, 1, 3).

**[Adapted from Multivariable Calculus: Concepts and Contexts, James Stewart]**

In Figure 3 we corroborate this impression by zooming in toward the point (1, 1) on a contour map of the function $f(x, y) = 2x^2 + y^2$. Notice that the more we zoom in, the more the level curves look like equally spaced parallel lines, which is charac-

# Gradient as a vector field

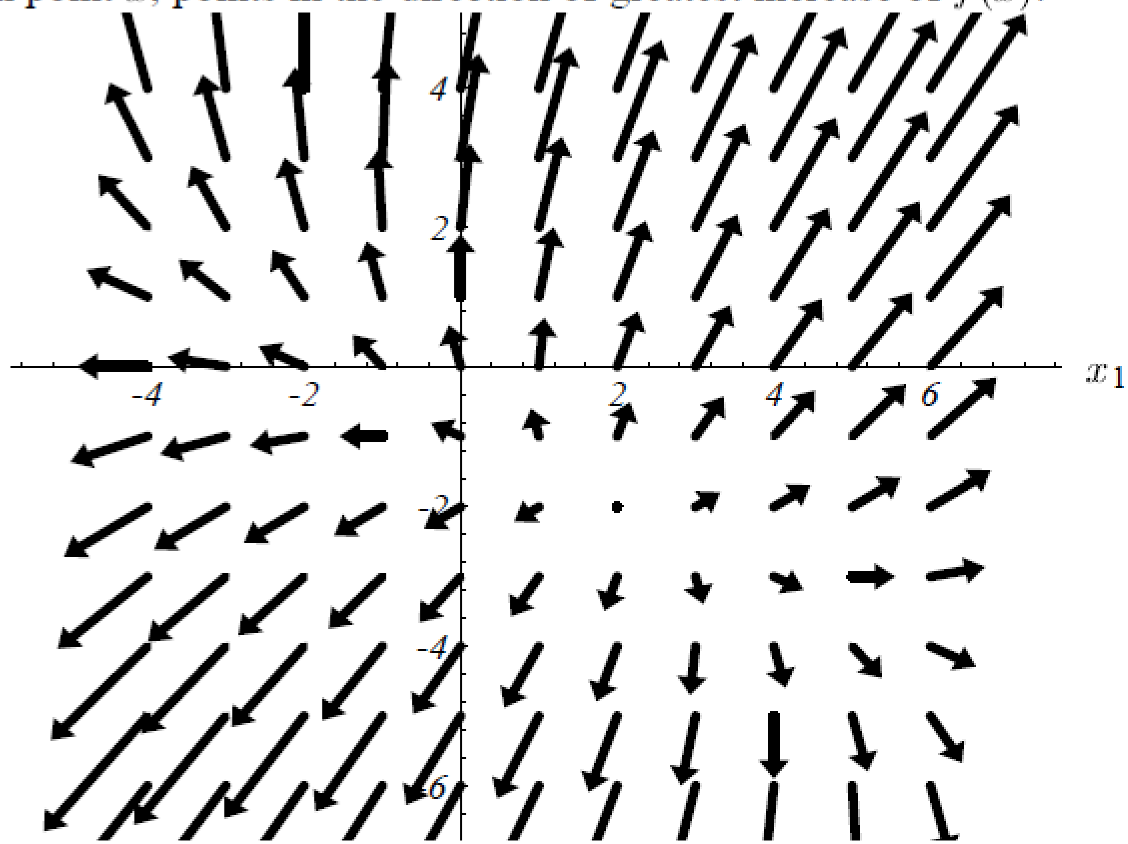
The *gradient* of a quadratic form is defined to be **(corresponds to slope in a single variable function)**

$$f'(x) = \begin{bmatrix} \frac{\partial}{\partial x_1} f(x) \\ \frac{\partial}{\partial x_2} f(x) \\ \vdots \\ \frac{\partial}{\partial x_n} f(x) \end{bmatrix} \quad (5)$$

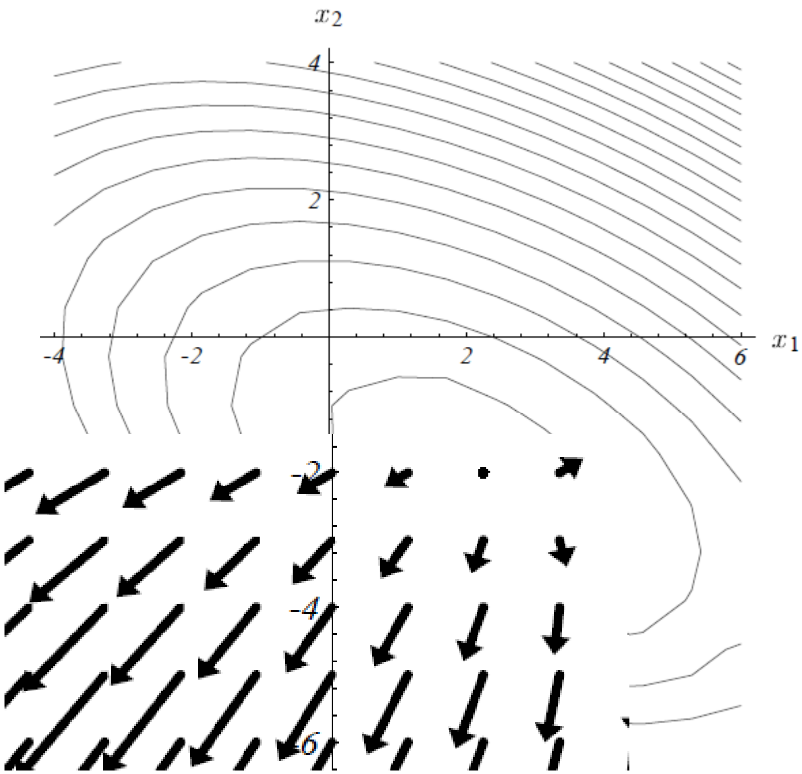
**At each point calculate the tangent plane (this plane approximates the surface at the point and in that point's neighbourhood). Recall Taylors Series?**

The gradient is a vector field that, for a given point $x$, points in the direction of greatest increase of $f(x)$.

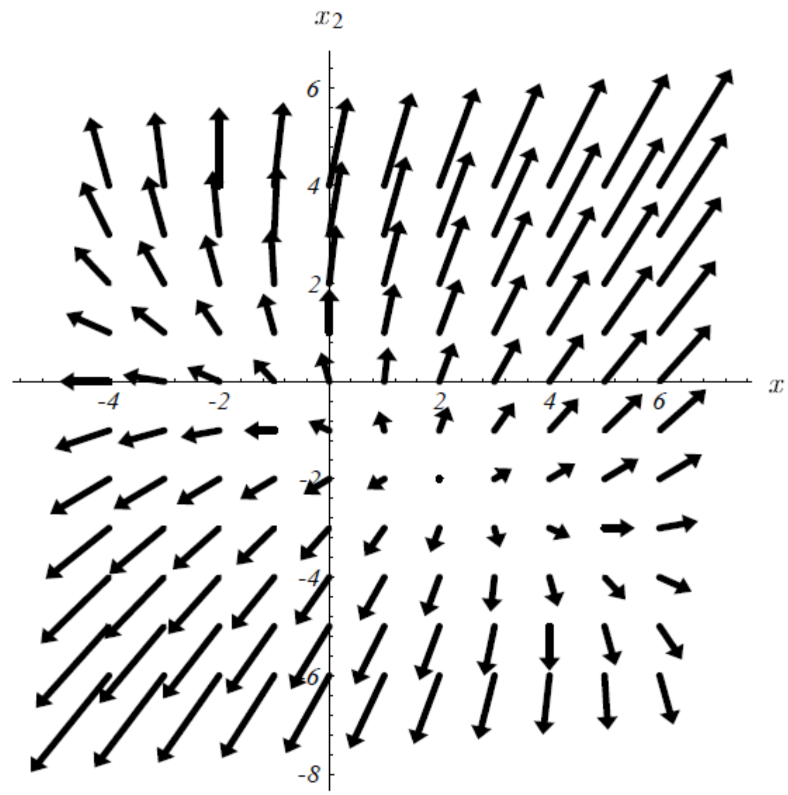
- **Gradient (f'(x)) of the quadratic form. For every point x, the gradient points in the direction of steepest increase f(x), and is orthogonal to the contour lines.**



# Gradient is orthogonal to contour




Contour plot of objective function f'x)), error function in our case. Each ellipsoidal curve has a constant error rate. For every point x, the gradient points in the direction of steepest increase of f(x), and is orthogonal to the contour lines.




Gradient (f'(x)) of the quadratic form for every point x, the gradient points in the direction of steepest increase f(x), and is orthogonal to the contour lines.

# Ordinary Least Squares Algorithm

## Single-sample Primal Forn

- **Given Training data S where each example *i* is of the form $(x_{i,1},\ldots, x_{i,n}, y_i)$, and a learning rate $\eta$**
- **Set $W_o$ to zeros; $k$=0;**
- **Repeat**
  - For i = 1 to |*Train*| do

    $$W_{k+1} = W_k + \eta \ (<W_{k,} X_i> - y_i) X_i$$
  - End-For
- **Until convergence**
- **Return *W***

**Iterative, gradient descent based algorithm (as opposed to other versions, such as closed form version, quadratic programming version, maximum likelihood. What could they look like?)**

# Exercise: predict height from shoe sizes

**Homework**

- **Create a small dataset**
  - Collect height (in centimeters) and shoe sizes in European sizes (e.g., I am 184 cm, with a shoe size of 46).
- **Train a OLS model using gradient descent**
  - Train OLS model using the iterative gradient descent algorithm
  - Plot model after each iteration
  - Compare to model learnt using lm(.) (in R).
- **Bonus: plot gradient, error contours, and error surfaces for bonus credits!**

# Closed form solution to OLS

How do we minimize (3.2)? Denote by $\mathbf{X}$ the $N \times (p+1)$ matrix with each row an input vector (with a 1 in the first position), and similarly let $\mathbf{y}$ be the $N$-vector of outputs in the training set. Then we can write the residual sum-of-squares as

$$\mathrm{RSS}(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta). \tag{3.3}$$

This is a quadratic function in the $p+1$ parameters. Differentiating with respect to $\beta$ we obtain

$$\begin{aligned} \frac{\partial \mathrm{RSS}}{\partial \beta} &= -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta) \\ \frac{\partial^2 \mathrm{RSS}}{\partial \beta \partial \beta^T} &= 2\mathbf{X}^T\mathbf{X}. \end{aligned} \tag{3.4}$$

Assuming (for the moment) that $\mathbf{X}$ has full column rank, and hence $\mathbf{X}^T\mathbf{X}$ is positive definite, we set the first derivative to zero

$$\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta) = 0 \tag{3.5}$$

to obtain the unique solution

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \tag{3.6}$$

[Friedman et al. 2001]

# Gradient Descent: other tidbits

- **Gradient descent can also be used to solve a system of nonlinear equations.**
- **Below is an example that shows how to use the gradient descent to solve for three unknown variables, $x_1$, $x_2$, and $x_3$.**

$$\begin{cases} 3x_1 - \cos(x_2x_3) - \frac{1}{2} = 0 \\ 4x_1^2 - 625x_2^2 + 2x_2 - 1 = 0 \\ \exp(-x_1x_2) + 20x_3 + \frac{10\pi-3}{3} = 0 \end{cases}$$

**[WikiPedia]**

- **A more powerful algorithm is given by the BFGS method which consists in calculating on every step a matrix by which the gradient vector is multiplied to go into a "better" direction, combined with a more sophisticated line search algorithm, to find the "best" value of γ.**

# Newton-Raphson Method

In numerical analysis, **Newton's method** (also known as the **Newton–Raphson method**), named after Isaac Newton and Joseph Raphson, is perhaps the best known method for finding successively better approximations to the zeroes (or roots) of a real-valued function. Newton's method can often converge remarkably quickly, especially if the iteration begins "sufficiently near" the desired root. Just how near "sufficiently near" needs to be, and just how quickly "remarkably quickly" can be, depends on the problem. This is discussed in detail below. Unfortunately, when iteration begins far from the desired root, Newton's method can easily lead an unwary user astray with little warning. Thus, good implementations of the method embed it in a routine that also detects and perhaps overcomes possible convergence failures.

Given a function $f(x)$ and its derivative $f'(x)$, we begin with a first guess $x_0$ . A better approximation $x_1$ is

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

[Wikipedia]

# Univariate Newton-Raphson Example

We wish to solve $4x\cos^2 x + \pi/2 - 2x - \sin(2x) = 0$. Obviously, plotting $f(x) = 4x\cos^2 x + \pi/2 - 2x - \sin(2x)$ and drawing tangents is not going to be very much fun! However, we can perform Newton-Raphson numerically.

**Find roots of equations that are differentiable.**

Our initial point is $x_0$. The gradient of $f(x)$ at $x_0$ is given by $f'(x_0)$, and the tangent line to $f(x)$ at $x_0$ is therefore given by:

$$y - f(x_0) = f'(x_0)(x - x_0)$$

**Given the slope *f'(x0),* and a point *x0*, calculate the tangent line (at approximates *f* in the neighbourhood of *xo***

To find $x_1$, we must find the point where this tangent crosses the $x$-axis, i.e. to let:

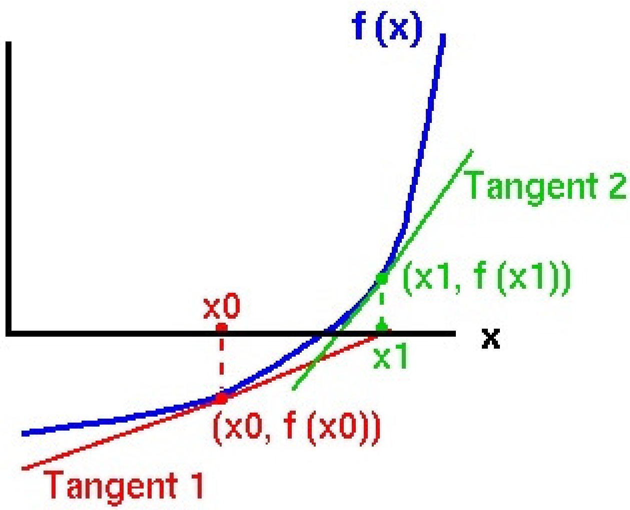
$$0 - f(x_0) = f'(x_0)(x_1 - x_0)$$

and therefore

$$x_1 - x_0 = \frac{-f(x_0)}{f'(x_0)}$$

so that

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$




**[Adapted from http://plus.maths.org/issue9/puzzle/solution.html]**

# Multivariate Newton's Method

Suppose that the objective $f$ is a function of multiple arguments, $f(w_1, w_2, \ldots w_p)$. Let's bundle the parameters into a single vector, $\vec{w}$. Then the Newton update is

$$\vec{w}_{n+1} = \vec{w}_n - H^{-1}(w_n)\nabla f(\vec{w}_n) \tag{16}$$

**Calculating gradient and Hessian not very time-consuming but calculating the inverse of H is!**

where $\nabla f$ is the **gradient** of $f$, its vector of partial derivatives $[\partial f/\partial w_1, \partial f/\partial w_2, \ldots \partial f/\partial w_p]$, and $H$ is the **Hessian** of $f$, its matrix of second partial derivatives, $H_{ij} = \partial^2 f/\partial w_i \partial w_j$.

Calculating $H$ and $\nabla f$ isn't usually very time-consuming, but taking the inverse of $H$ is, unless it happens to be a diagonal matrix. This leads to various **quasi-Newton** methods, which either approximate $H$ by a diagonal matrix, or take a proper inverse of $H$ only rarely (maybe just once), and then try to update an estimate of $H^{-1}(w_n)$ as $w_n$ changes. (See section 8.3 in the textbook for more.)

**In R, have a look at**
**?optim #method=BFGS**

[http://www.stat.cmu.edu/~cshalizi/350/2008/lecture29/lecture-29.pdf]

[Hand, Manilla, Smith, Data Mining, Section 8.3]

# Limitations of Newton's Method

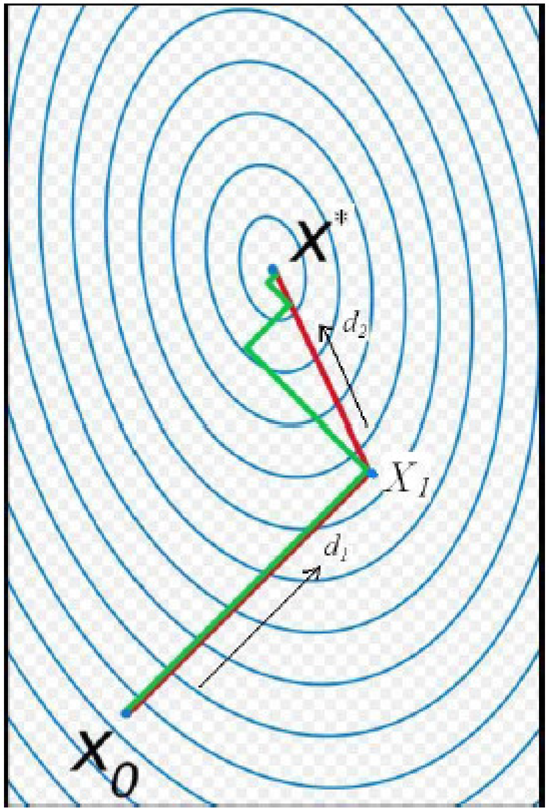
- **Step size can a guessing game in Newton's method**

$$\mathbf{b} = \mathbf{a} - \gamma \nabla F(\mathbf{a})$$

- **There are other methods for finding minimums besides Newton's method**
  - Such as gradient descent, conjugate gradient or variations of the Gauss-Newton method avoid this guessing
- **Applications**
  - Finding minimum or maximum of a function (e.g., linear regression)
  - Neural Networks
  - Linear Programming, quadratic programming.
  - Finding maximum likelihood estimates
    - Unlike EM, such methods typically require the evaluation of first and/or second derivatives of the likelihood function.
    - E.g., Logistic Regression

**In R, have a look at optim()**
**type ?optim #method=BFGS**

# Newton < Gradient Descent < Conjugate




- Newton's method requires evaluation, storage and inversion of matrix; computationally complex.
- Gradient descent typically converges slowly.
- Conjugate direction methods is intermediate b/w the above two, which has proved to be extremely effective in dealing with general objective functions.
- A comparison of the convergence of gradient descent with optimal step size (in green) and conjugate gradient (in red) for minimizing a quadratic function associated with a given linear system. Conjugate gradient, assuming exact arithmetics, converges in at most *n* steps where *n* is the size of the matrix of the system (here *n*=2).

# Forward Markets

- **Gradient Descent**
- **Linear Programming**
- **Quadratic Programming**
- **Allocation of Ads to Publisher real estate**
  - Give ads play in network
    - Optimize *revenue* subject to ….
- **Inventory Management**
  - Contract as many impressions as possible but don't oversell
- **Media Buyer (Arbitrage)**
  - Frame as a non-linear programming (NLP) problem
  - Talks to publisher
  - Determine publisher mix for network
    - Optimize *publisher mix* subject to constraints

# Linear Programming (LP)

*Linear programming* is a mathematical technique that enables a decision maker to arrive at the optimal solution to problems involving the allocation of scarce resources.

Typically, many economic and technical problems involve maximization or minimization of a certain objective subject to some restrictions.

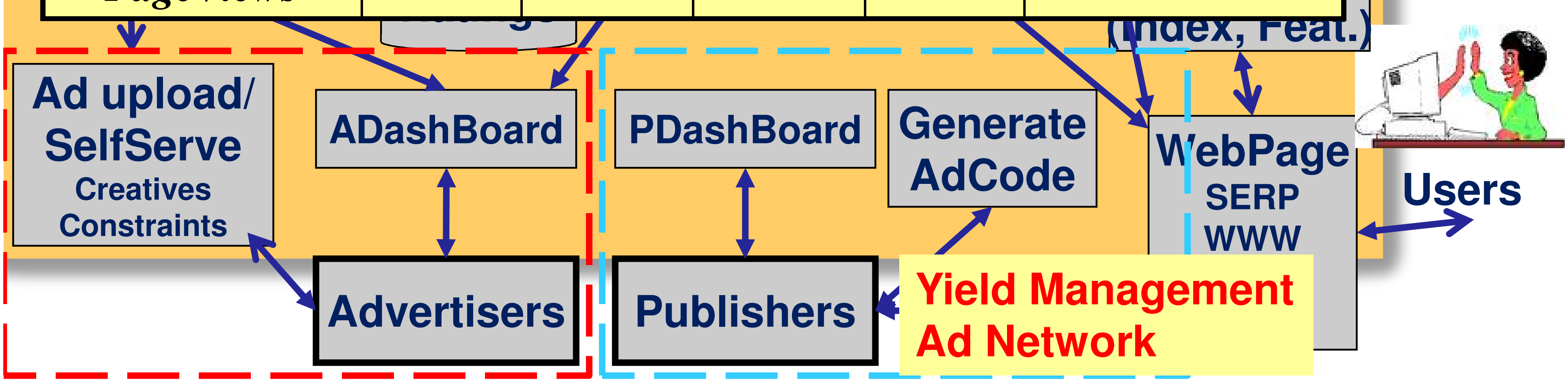
LP is a technique for optimization of a linear objective function, subject to linear equality and linear inequality constraints

# LP Outline

- Introduction and some motivating advertising problems
- Linear Algebra Basics Review
- Fundamental theorem of LP
- Matrix-view and the fundamental insight
- Duality
- Interior point Algorithm
- Transportation Problem
- Applying linear programming to online advertising
- Summary

# Ad Network Architecture:Forward Markets

| *From* | *To* | | | | |
|---|---|---|---|---|---|
| | *Ad1* | *Ad2* | *..Ad$_j$.....* | *Ad$_m$* | *Supply PageViews* |
| ***PubZone 1*** | $d_{ij}$ | $d_{ij}$ | $d_{ij}$ | $d_{ij}$ | 35 |
| ***PubZone 2*** | … | … | … | … | 50 |
| ***PubZone3*** | $d_{ij}$ | $d_{ij}$ | $d_{ij}$ | $d_{ij}$ | 15 |
| ***Demand Contracted PageViews*** | *45* | *20* | *30* | *5* | |

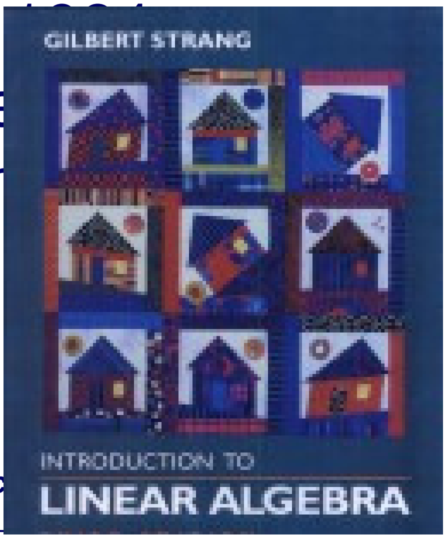


# Ad Networks and Optimisation

- **Allocation of Ads to Publisher real estate**
  - Give ads play in network
    - Optimize *revenue* subject to ….
- **Inventory Management**
  - Contract as many impressions as possible but don't oversell
- **Media Buyer (Arbitrage) (NLP-problem)**
  - Talks to publisher
  - Determine publisher mix for network
    - Optimize *publisher mix* subject to constraints

# History of LP

- The founders of the subject are Leonid Kantorovich,
1939 a Russian mathematician who developed linear programming problems in 1939, George B. Dantzig, who published the simplex method in 1947, John von
1947 Neumann, who developed the theory of the duality in the same year.
- The linear programming problem was first shown to be solvable in polynomial time by Leonid Khachiyan in 1979, but a larger theoretical and
1984 practical breakthrough in the field came in … when Narendra Karmarkar introduced a ne… point method for solving linear programmi… problems.



# What is Linear Programming

- **Linear programming (LP) =**
  - Linear Algebra + inequalities + optimization (minimize or maximize)
- **LP is a technique for optimization of a linear objective function, subject to linear equality and linear inequality constraints.**
  - Informally, linear programming determines the way to achieve the best outcome (such as maximum profit or lowest cost) in a given mathematical model and given some list of requirements represented as linear equations.
  - More formally, given a polytope (for example, a polygon or a polyhedron), and a real-valued affine function

$$\boxed{f(x_1, x_2, \ldots, x_n) = c_1 x_1 + c_2 x_2 + \cdots + c_n x_n + d}$$

  - defined on this polytope, a linear programming method will find a point in the polytope where this function has the smallest (or largest) value.

# Types of LP descriptions

$$3x_1 - 5x_2 \leq 15$$

$$2x_1 + 3x_2 = 12$$

$$3x_1 + x_2 \geq -2$$

$$x_1 \leq 0, (x_2 \, free)$$

$$\max x_1 + x_2$$

**To deal with different types of objectives and constraints we convert each linear program to *standard form*.**

# Standard Form

## (according to Hillier and Lieberman)

$$
\begin{aligned}
&\max\ c_1x_1 + c_2x_2 + \ldots + c_Nx_N \\
&\text{subject to} \\
&\qquad a_{11}x_1 + a_{12}x_2 + \ldots + a_{1N}x_N \le b_1 \\
&\qquad a_{21}x_1 + a_{22}x_2 + \ldots + a_{2N}x_N \le b_2 \\
&\qquad \ldots \\
&\qquad a_{M1}x_1 + a_{M2}x_2 + \ldots + a_{MN}x_N \le b_M \\
&\qquad x_j \ge 0,\ j = 1..N
\end{aligned}
$$

**Concise version:**

$$
\begin{aligned}
&\max\ c'x \\
&\text{subject to}\ Ax \le b \\
&\qquad x \ge 0
\end{aligned}
$$

**A is an *m by n* matrix: n variables, m constraints**

# Converting into Augmented Form

- Slack/surplus variables
- Replacing ‘free’ variables
- Minimization vs maximization

- See Luenburger (page 11,12 etc)

# Standard Form to Augmented Form

$$\begin{aligned} &\max \ c_1x_1 + c_2x_2 + \ldots + c_Nx_N \\ &\text{subject to} \\ &\quad a_{11}x_1 + a_{12}x_2 + \ldots + a_{1N}x_N \le b_1 \\ &\quad a_{21}x_1 + a_{22}x_2 + \ldots + a_{2N}x_N \le b_2 \\ &\quad \ldots \\ &\quad a_{M1}x_1 + a_{M2}x_2 + \ldots + a_{MN}x_N \le b_M \\ &\quad x_j \ge 0,\ j = 1..N \end{aligned}$$

$$\begin{aligned} &\max \ c_1x_1 + c_2x_2 + \ldots + c_Nx_N \\ &\text{subject to} \\ &\quad a_{11}x_1 + a_{12}x_2 + \ldots + a_{1N}x_N + \widehat{x}_1 = b_1 \\ &\quad a_{21}x_1 + a_{22}x_2 + \ldots + a_{2N}x_N + \widehat{x}_2 = b_2 \\ &\quad \ldots \\ &\quad a_{M1}x_1 + a_{M2}x_2 + \ldots + a_{MN}x_N + \widehat{x}_N = b_M \\ &\quad x_j, \widehat{x}_i \ge 0,\ j = 1..N, i = 1..m \end{aligned}$$

$$\begin{aligned} &\max \ c'x \\ &\text{subject to} \, Ax \le b \\ &\quad x \ge 0 \end{aligned}$$

$$\begin{aligned} &\max \ c'x \\ &\text{subject to} [A, I] \begin{bmatrix} x \\ \widehat{x} \end{bmatrix} = b \\ &\quad x \ge 0 \end{aligned}$$

**A is an *m by n* matrix: n variables, m constraints**

# Farmer Example

- Suppose that a farmer has a piece of farm land, say A square kilometers large, to be planted with either wheat or barley or some combination of the two.
- The farmer has a limited permissible amount F of fertilizer and P of insecticide which can be used, each of which is required in different amounts per unit area for wheat (F1, P1) and barley (F2, P2).
- Let S1 be the selling price of wheat, and S2 the price of barley. If we denote the area planted with wheat and barley with x1 and x2 respectively, then the optimal number of square kilometers to plant with wheat vs. barley can be expressed as a linear programming problem

# Farmer Example: LP

maximize $S_1x_1 + S_2x_2$ **( maximize the revenue – this is the "objective function")**

**subject to** $x_1 + x_2 \leq A$ **(limit on total area)**

$F_1x_1 + F_2x_2 \leq F$ **(limit on fertilizer)**

$P_1x_2 + P_2x_2 \leq P$ **(limit on insecticide)**

$x_1 >= 0,\ x_2 \geq 0$ **(cannot plant a negative area)**

**which in matrix form becomes**

**maximize** $$\begin{bmatrix} S_1 & S_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

**subject to** $$\begin{bmatrix} 1 & 1 \\ F_1 & F_2 \\ P_1 & P_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} A \\ F \\ P \end{bmatrix},\ \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \geq 0$$

# Farmer Example: Augmented Form

- **Linear programming problems must be converted into augmented form before being solved by the simplex algorithm. This form introduces non-negative slack variables to replace non-equalities with equalities in the constraints. The problem can then be written on the following form:**

**Maximize Z in:**

$$\begin{bmatrix} 1 & -\mathbf{c}^T & 0 \\ 0 & \mathbf{A} & \mathbf{I} \end{bmatrix} \begin{bmatrix} Z \\ \mathbf{x} \\ \mathbf{x}_s \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix}$$

$$\mathbf{x},\, \mathbf{x}_s \geq 0$$

# Farmer Example: Augmented Form

**The example 1 above becomes as follows when converted Into augmented form:**

| | | |
|---|---|---|
| **maximize** | $S_1x_1 + S_2x_2$ | **(objective function)** |
| **subject to** | $x_1 + x_2 + x3 = A$ | **(augmented constraint)** |
| | $F_1x_1 + F_2x_2 \leq F + x_4 = F$ | **(augmented constraint)** |
| | $P_1x_2 + P_2x_2 + x_5 = P$ | **(augmented constraint)** |

**where $x_3$,$x_4$,$x_5$ are (non-negative) slack variables.**

**Which in matrix form becomes:**

objective

**Maximize Z in:**

$$\begin{bmatrix} 1 & -S_1 & -S_2 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & F_1 & F_2 & 0 & 1 & 0 \\ 0 & P_1 & P_2 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Z \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 0 \\ A \\ F \\ P \end{bmatrix}, \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \geq 0$$

ProblemVars Slacks

# Outline

# Linear Independent.

**DEFINITION** The columns of $A$ are *linearly independent* when the only solution to $Ax = \mathbf{0}$ is $x = \mathbf{0}$. *No other combination $Ax$ of the columns gives the zero vector.*

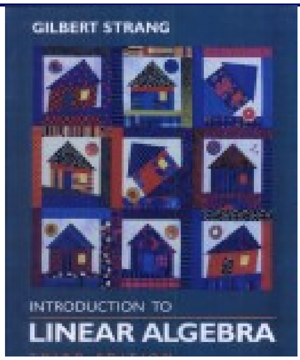
With linearly independent columns, the nullspace $N(A)$ contains only the zero vector. Let me illustrate linear independence (and linear dependence) with three vectors in $\mathbf{R}^3$:

1. If three vectors are *not* in the same plane, they are independent. No combination of $v_1, v_2, v_3$ in Figure 3.4 gives zero except $0v_1 + 0v_2 + 0v_3$.

2. If three vectors $w_1, w_2, w_3$ are *in the same plane*, they are dependent.

This idea of independence applies to 7 vectors in 12-dimensional space. If they are the columns of $A$, and independent, the nullspace only contains $x = \mathbf{0}$. Now we choose different words to express the same idea. The following definition of independence will apply to any sequence of vectors in any vector space. When the vectors are the columns of $A$, the two definitions say exactly the same thing.






**DEFINITION** The sequence of vectors $v_1, \ldots, v_n$ is ***linearly independent*** if the only combination that gives the zero vector is $0v_1 + 0v_2 + \cdots + 0v_n$. Thus linear independence means that

$$x_1 v_1 + x_2 v_2 + \cdots + x_n v_n = 0 \quad \text{only happens when all } x\text{'s are zero.} \qquad (1)$$

If a combination gives $\mathbf{0}$, when the $x$'s are not all zero, the vectors are *dependent*.

Correct language: "The sequence of vectors is linearly independent." Acceptable shortcut: "The vectors are independent." Unacceptable: "The matrix is independent."

A sequence of vectors is either dependent or independent. They can be combined to give the zero vector (with nonzero $x$'s) or they can't. So the key question is: Which combinations of the vectors give zero? We begin with some small examples in $\mathbf{R}^2$:

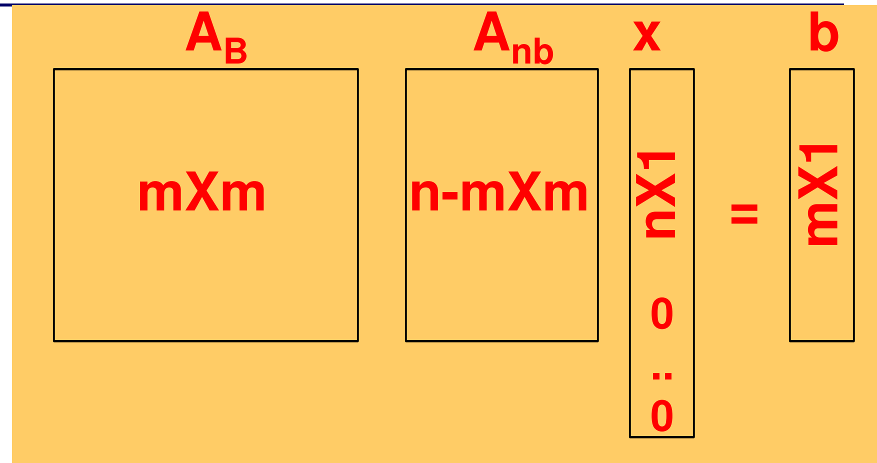
(a) The vectors $(1, 0)$ and $(0, 1)$ are independent.

(a) The vectors $(1, 0)$ and $(0, 1)$ are independent.

James.Shanahan_AT_gmail_DOT_com

# Background on Matrices

$$
\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + \ldots + a_{1N}x_N \le b_1 \\
&a_{21}x_1 + a_{22}x_2 + \ldots + a_{2N}x_N \le b_2 \\
&\ldots \\
&a_{M1}x_1 + a_{M2}x_2 + \ldots + a_{MN}x_N \le b_M
\end{aligned}
$$




$Ax=b$

$A^{-1}Ax=A^{-1}b$

$Ix=A^{-1}b$

Solve this system of equations through
1. Gaussian elimination or
2. Using matrix inverses

Equivalent conditions for invertibility of a square matrix A:

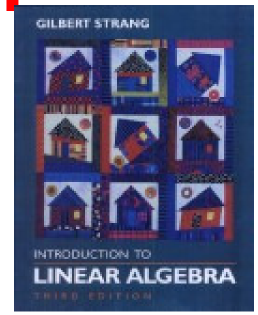
- A invertible
- rank A = n
- det A ≠ 0
- columns (and rows) are linearly independent
- Ax = 0 has a unique solution

If A is invertible, then Ax=b has a unique solution for any b. If A is not invertible, then Ax=b either has no solution or infinitely many solutions.

# Basis of Vector Space

This combination of properties is fundamental to linear algebra. Every vector $v$ in the space is a combination of the basis vectors, because they span the space. More than that, the combination that produces $v$ is *unique*, because the basis vectors $v_1, \ldots, v_n$ are independent:

**There is one and only one way to write $v$ as a combination of the basis vectors.**




### A Basis for a Vector Space

In the $xy$ plane, a set of independent vectors could be quite small—just one vector. A set that spans the $xy$ plane could be large—three vectors, or four, or infinitely many. One vector won't span the plane. Three vectors won't be independent. A "***basis***" is just right. We want ***enough independent vectors to span the space***.

**DEFINITION** A ***basis*** for a vector space is a sequence of vectors that has two properties at once:

1. The vectors are *linearly independent*.
2. The vectors *span the space*.

when A is invertible then for

$$Ax = b$$

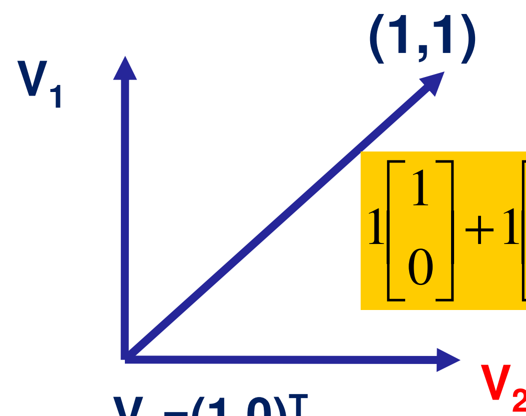
$$x = A^{-1}b$$

**I.e., b can expressed as a unique linear combination of the basis vectors**

This combination of properties is fundamental to linear algebra. Every vector $v$ in the space is a combination of the basis vectors, because they span the space. More than that, the combination that produces $v$ is *unique*, because the basis vectors $v_1, \ldots, v_n$ are independent:

**There is one and only one way to write $v$ as a combination of the basis vectors.**

**Reason:** Suppose $v = a_1v_1 + \cdots + a_nv_n$ and also $v = b_1v_1 + \cdots + b_nv_n$. By subtraction $(a_1 - b_1)v_1 + \cdots + (a_n - b_n)v_n$ is the zero vector. From the independence of the $v$'s, each $a_i - b_i = 0$. Hence $a_i = b_i$.




$$1\begin{bmatrix}1\\0\end{bmatrix}+1\begin{bmatrix}0\\1\end{bmatrix}=\begin{bmatrix}1\\1\end{bmatrix}$$

**$V_1=(1,0)^T$**
**$V_2=(0,1)^T$**

**Standard basis**

**Example 6** The columns of $I = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$ produce the "standard basis" for $\mathbf{R}^2$.

The basis vectors $i = \begin{bmatrix}1\\0\end{bmatrix}$ and $j = \begin{bmatrix}0\\1\end{bmatrix}$ are independent. They span $\mathbf{R}^2$.

Everybody thinks of this basis first. The vector $i$ goes across and $j$ goes straight up. The columns of the 3 by 3 identity matrix are the standard basis $i$, $j$, $k$. The columns of the $n$ by $n$ identity matrix give the "standard basis" for $\mathbf{R}^n$. Now we find other bases.

# Invert Matrix and Basis

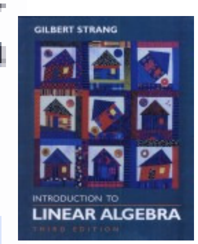
**Example 7** (Important) The columns of *any invertible n by n matrix* give a basis for $\mathbf{R}^n$:

$$A = \begin{bmatrix} 1 & 2 \\ 2 & 5 \end{bmatrix} \quad \text{and} \quad A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad \text{but not} \quad A = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}.$$

When $A$ is invertible, its columns are independent. The only solution to $A\boldsymbol{x} = \mathbf{0}$ is $\boldsymbol{x} = \mathbf{0}$. The columns span the whole space $\mathbf{R}^n$—because every vector $\boldsymbol{b}$ is a combination of the columns. $A\boldsymbol{x} = \boldsymbol{b}$ can always be solved by $\boldsymbol{x} = A^{-1}\boldsymbol{b}$. Do you see how everything comes together for invertible matrices? Here it is in one sentence:

**3J** The vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ are a *basis for* $\mathbf{R}^n$ exactly when they are ***the columns of an n by n invertible matrix***. Thus $\mathbf{R}^n$ has infinitely many different bases.

When any matrix has independent columns, they are a basis for its column space. When the columns are dependent, we keep only the *pivot columns*—the $r$ columns with pivots. They are independent and they span the column space.




**3K** ***The pivot columns of A are a basis for its column space.*** The pivot rows of $A$ are a basis for its row space. So are the pivot rows of its echelon form $R$.



$n$

# Non-invertible matrices

**Example 8** This matrix is not invertible. Its columns are not a basis for anything!

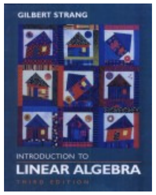
$$A = \begin{bmatrix} 2 & 4 \\ 3 & 6 \end{bmatrix} \text{ which reduces to } R = \begin{bmatrix} 1 & 2 \\ 0 & 0 \end{bmatrix}.$$

Column 1 of $A$ is the pivot column. That column alone is a basis for its column space. The second column of $A$ would be a different basis. So would any nonzero multiple of that column. There is no shortage of bases! So we often make a definite choice: the pivot columns.

Notice that the pivot column of this $R$ ends in zero. That column is a basis for the column space of $R$, but it is not even a member of the column space of $A$. The column spaces of $A$ and $R$ are different. Their bases are different.

The row space of $A$ is the *same* as the row space of $R$. It contains $(2,4)$ and $(1,2)$ and all other multiples of those vectors. As always, there are infinitely many bases to choose from. I think the most natural choice is to pick the nonzero rows of $R$ (rows with a pivot). So this matrix $A$ with rank one has only one vector in the basis:

$$\text{Basis for the column space: } \begin{bmatrix} 2 \\ 3 \end{bmatrix}. \quad \text{Basis for the row space: } \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$




The next chapter will come back to these bases for the column space and row space. We are happy first with examples where the situation is clear (and the idea of a basis is still new). The next example is larger but still clear.

# Find a basis for for b in terms of 4 cols

Matrix-vector multiplication as a linear combination of columns.

Basic solutions of Ax=b

an example

**Reduced row echelon form**
**So can read off the soln**

Consider the matrix $A = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & -1 \end{pmatrix}$. Notice that columns 1,2,3,4 are not a basis. Neither is 2,3,4,5. But 1,2,4,6 is basic.

Let $b = \begin{pmatrix} 3 \\ 2 \\ 5 \\ 4 \end{pmatrix}$. How can we find the unique (basic) solution to Ax=b that uses only columns 1,2,4,6? If we multiply both sides of the equation Ax=b by the inverse of the 4×4 matrix consisting of those columns, we get the answer:

**$B^{-1}A$    $x = B^{-1}b$**

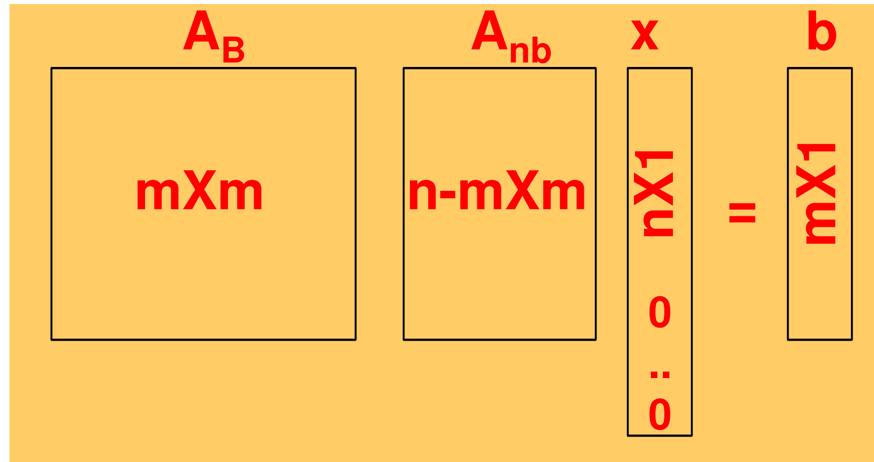
$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}^{-1} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix} \text{ so } \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} x = \begin{pmatrix} 4 \\ -2 \\ 1 \\ 1 \end{pmatrix}$$

is an equivalent system. When the system is rewritten in this equivalent form, the basic solution x=(4,-2,0,1,0,1,0) becomes evident. If we are only interested in solutions for which x≥0, then x would be an *infeasible* basic solution since $x_2$ is negative.

This process could be repeated for every set of basic columns.

# Basic Solution (to a system of eqns.)

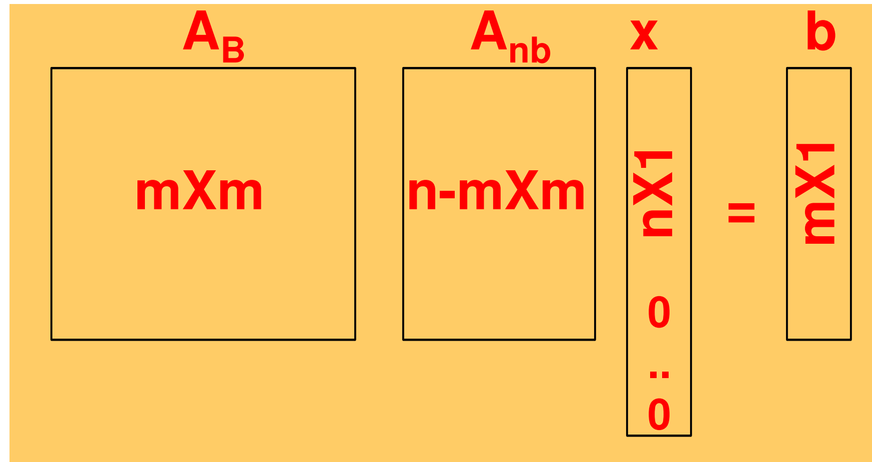
$$
\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + \ldots + a_{1N}x_N \le b_1 \\
&a_{21}x_1 + a_{22}x_2 + \ldots + a_{2N}x_N \le b_2 \\
&\ldots \\
&a_{M1}x_1 + a_{M2}x_2 + \ldots + a_{MN}x_N \le b_M
\end{aligned}
$$




- **Given *Ax=b*** **A system of equations (8)**
- **Definition. Given the set of *m* simultaneous linear equations in *n* unknowns (8), let B (denoted $A_B$) be any nonsingular *m×m* submatrix made up of columns of A.**
- **Then, if all *n–m* components of x not associated with columns of B are set equal to zero, the solution to the resulting set of equations is said to be a *<u>basic solution</u> to (8) with respect to the basis B. The components of x associated* with columns of B are called *<u>basic variables</u>. The remaining n-r variables are non-basic.***
- **Assume that the first m columns of A make up B (denoted as $A_B$)**

# Basic Solution (to a system of eqns.)

$$
\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + \ldots + a_{1N}x_N \le b_1 \\
&a_{21}x_1 + a_{22}x_2 + \ldots + a_{2N}x_N \le b_2 \\
&\ldots \\
&a_{M1}x_1 + a_{M2}x_2 + \ldots + a_{MN}x_N \le b_M
\end{aligned}
$$




- **Given *Ax=b*** **A system of equations (8)**
- **Definition. Given the set of *m* simultaneous linear equations in *n* unknowns (8), let B be any nonsingular *m×m* submatrix made up of columns of A.**
- **In the above definition we refer to *B* as a basis, since *B* consists of *m* linearly independent columns that can be regarded as a basis for the space $R^m$. The basic solution corresponds to an expression for the vector b as a linear combination of these basis vectors. Assume that the first *m* columns of *A* make up *B* (denoted as $A_B$)**

# Full Rank Assumption

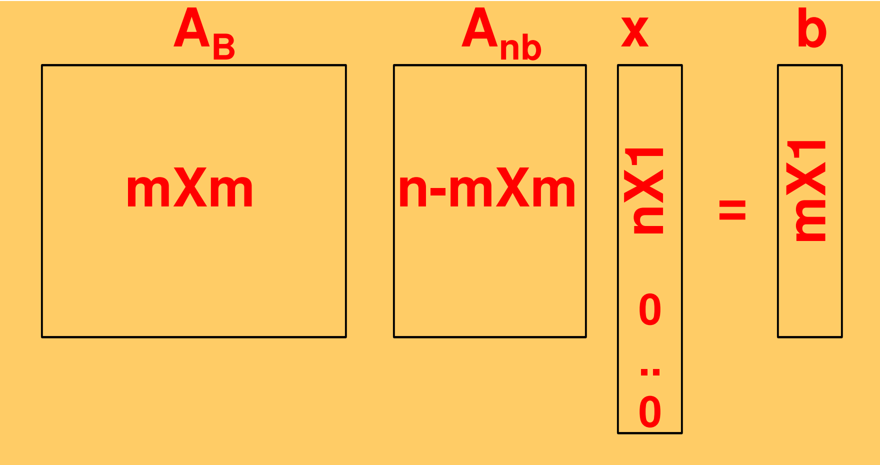
- ***Ax=b*** **A system of equations (8)**

In general, of course, Eq. (8) may have no basic solutions. However, we may avoid trivialities and difficulties of a nonessential nature by making certain elementary assumptions regarding the structure of the matrix $\mathbf{A}$. First, we usually assume that $n > m$, that is, the number of variables $x_i$ exceeds the number of equality constraints. Second, we usually assume that the rows of $\mathbf{A}$ are linearly independent, corresponding to linear independence of the $m$ equations. A linear dependency among the rows of $\mathbf{A}$ would lead either to contradictory constraints and hence no solutions to (8), or to a redundancy that could be eliminated. Formally, we explicitly make the following assumption in our development, unless noted otherwise.

***Full rank assumption***. *The $m \times n$ matrix $\mathbf{A}$ has $m < n$, and the $m$ rows of $\mathbf{A}$ are linearly independent.*

Under the above assumption, the system (8) will always have a solution and, in fact, it will always have at least one basic solution.

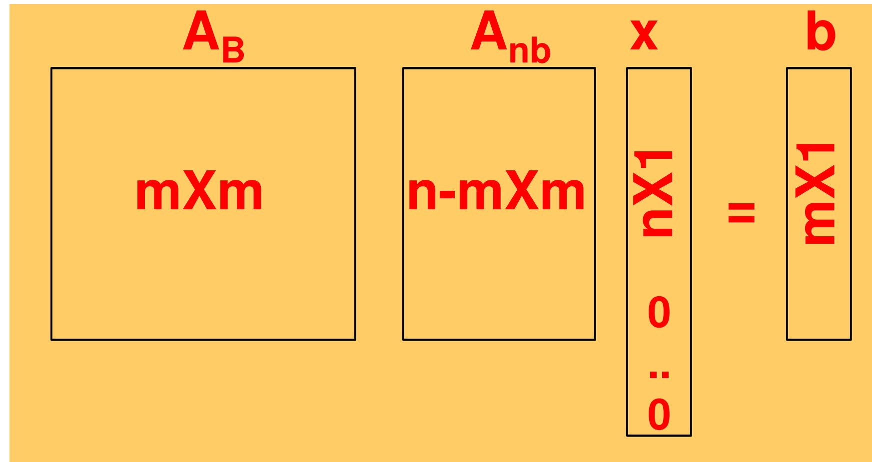
# Degenerate Basic Solution




- The basic variables in a basic solution (i.e., in *x*) are not necessarily all nonzero. This is noted by the following definition.
- If one or more of the basic variables in a basic solution has value zero, that solution is said to be a ***degenerate basic solution***.

# Basic Feasible Solution

$$
\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + \ldots + a_{1N}x_N \le b_1 \\
&a_{21}x_1 + a_{22}x_2 + \ldots + a_{2N}x_N \le b_2 \\
&\ldots \\
&a_{M1}x_1 + a_{M2}x_2 + \ldots + a_{MN}x_N \le b_M \\
&\boxed{x_j \ge 0,\ j = 1..N}
\end{aligned}
$$




**Ax=b**

**x≥0** (eqn. 10)

- **A vector x satisfying (10) is said to be *feasible for these* *constraints*. A feasible solution to the constraints (10) that is also basic is said to be a *basic feasible solution;***
- ***if this solution is also a degenerate basic solution,* it is called a *degenerate basic feasible solution.***

# An Example with 35 (possible) basis

$$
\begin{array}{lllllllll}
x_1 & +x_2 & & +x_4 & & & & = & 3 \\
x_1 & +x_2 & & & -x_5 & & & = & 2 \\
x_1 & & +x_3 & & & +x_6 & & = & 5 \\
x_1 & & +x_3 & & & & -x_7 & = & 4 \\
x_j \geq 0, \; j = 1..7 & & & & & & & &
\end{array}
$$

There are $\binom{7}{4} = 35$ ways to choose four columns from the $4\times7$ coefficient matrix. Each fits into one of 3 categories:

i. The 4 columns do form a basis and the corresponding basic solution is feasible (all variables are nonnegative).

ii. The 4 columns do form a basis (the $4\times4$ matrix is invertible) but the corresponding basic solution is infeasible (one variable is negative).

iii. The corresponding 4 columns of the coefficient matrix form a singular (not invertible) $4\times4$ matrix. In other words, these columns do not form a basis.

# 35 possible basis

Below are all 35 possibilities. For basic solutions, we write the column numbers of the basis, the value of the objective, the solution vector, the equivalent system of equations that displays the solution vector. For example, the first basic feasible solution uses columns 1,3,4,6, the corresponding system of equations is

$$\begin{pmatrix} 1 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} x = \begin{pmatrix} 2 \\ 2 \\ 1 \\ 1 \end{pmatrix}$$

and the solution obtained by setting nonbasic variables equal to 0 is (2,0,2,1,0,1,0), where the value of the objective function is $x_1+2x_2+3x_3=8$.

<u>category i</u>: basic feasible solutions

columns: {1, 3, 4, 6}   objective = 8   x = (2,0,2,1,0,1,0)   vertex A

{{1, 0, 0, 0}, {1, -1, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0},
{-1, 1, 1, 0}, {0, 0, 0, 1}, {0, -1, 0, 1}, {2, 2, 1, 1}}

columns: {1, 3, 4, 7}   objective = 11   x = (2,0,3,1,0,0,1)

{{1, 0, 0, 0}, {1, -1, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0},
{-1, 1, 1, 0}, {0, 1, 0, 1}, {0, 0, 0, 1}, {2, 3, 1, 1}}

**...continued over the next couple of slides**

# Category 1: 8 Basic Feasible Solutions

```
columns: {1, 3, 4, 6}   objective = 8   x = (2,0,2,1,0,1,0)       vertex A
{{1, 0, 0, 0}, {1, -1, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0},
 {-1, 1, 1, 0}, {0, 0, 0, 1}, {0, -1, 0, 1}, {2, 2, 1, 1}}

columns: {1, 3, 4, 7}   objective = 11   x = (2,0,3,1,0,0,1)
{{1, 0, 0, 0}, {1, -1, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0},
 {-1, 1, 1, 0}, {0, 1, 0, 1}, {0, 0, 0, 1}, {2, 3, 1, 1}}

columns: {1, 3, 5, 6}   objective = 6  *MIN*   x = (3,0,1,0,1,1,0)
{{1, 0, 0, 0}, {1, -1, 0, 0}, {0, 1, 0, 0}, {1, -1, 1, 0},
 {0, 0, 1, 0}, {0, 0, 0, 1}, {0, -1, 0, 1}, {3, 1, 1, 1}}

columns: {1, 3, 5, 7}   objective = 9   x = (3,0,2,0,1,0,1)
{{1, 0, 0, 0}, {1, -1, 0, 0}, {0, 1, 0, 0}, {1, -1, 1, 0},
 {0, 0, 1, 0}, {0, 1, 0, 1}, {0, 0, 0, 1}, {3, 2, 1, 1}}

columns: {2, 3, 4, 6}   objective = 16   x = (0,2,4,1,0,1,0)      vertex B
{{1, 1, 0, 0}, {1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0},
 {-1, 0, 1, 0}, {0, 0, 0, 1}, {0, -1, 0, 1}, {2, 4, 1, 1}}

columns: {2, 3, 4, 7}   objective = 19   x = (0,2,5,1,0,0,1)      vertex C
{{1, 1, 0, 0}, {1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0},
 {-1, 0, 1, 0}, {0, 1, 0, 1}, {0, 0, 0, 1}, {2, 5, 1, 1}}

columns: {2, 3, 5, 6}   objective = 18   x = (0,3,4,0,1,1,0)
{{1, 1, 0, 0}, {1, 0, 0, 0}, {0, 1, 0, 0}, {1, 0, 1, 0},
 {0, 0, 1, 0}, {0, 0, 0, 1}, {0, -1, 0, 1}, {3, 4, 1, 1}}

columns: {2, 3, 5, 7}   objective = 21  *MAX*   x = (0,3,5,0,1,0,1)  vertex D
{{1, 1, 0, 0}, {1, 0, 0, 0}, {0, 1, 0, 0}, {1, 0, 1, 0},
 {0, 0, 1, 0}, {0, 1, 0, 1}, {0, 0, 0, 1}, {3, 5, 1, 1}}
```

Max

# Category 2: 13 Basic Infeasible Solutions

```
columns: {1, 2, 4, 6}   objective = 0   x = (4,-2,0,1,0,1,0)
{{1, 0, 0, 0}, {0, 1, 0, 0}, {1, -1, 0, 0}, {0, 0, 1, 0},
 {0, -1, 1, 0}, {0, 0, 0, 1}, {-1, 1, 0, 1}, {4, -2, 1, 1}}
columns: {1, 2, 4, 7}   objective = -1   x = (5,-3,0,1,0,0,1)
{{1, 0, 0, 0}, {0, 1, 0, 0}, {1, -1, 0, 0}, {0, 0, 1, 0},
 {0, -1, 1, 0}, {1, -1, 0, 1}, {0, 0, 0, 1}, {5, -3, 1, 1}}
columns: {1, 2, 5, 6}   objective = 2   x = (4,-1,0,0,1,1,0)
{{1, 0, 0, 0}, {0, 1, 0, 0}, {1, -1, 0, 0}, {0, 1, 1, 0},
 {0, 0, 1, 0}, {0, 0, 0, 1}, {-1, 1, 0, 1}, {4, -1, 1, 1}}
columns: {1, 2, 5, 7}   objective = 1   x = (5,-2,0,0,1,0,1)
{{1, 0, 0, 0}, {0, 1, 0, 0}, {1, -1, 0, 0}, {0, 1, 1, 0},
 {0, 0, 1, 0}, {1, -1, 0, 1}, {0, 0, 0, 1}, {5, -2, 1, 1}}
columns: {1, 4, 5, 6}   objective = 4   x = (4,0,0,-1,2,1,0)
{{1, 0, 0, 0}, {0, 1, -1, 0}, {1, -1, 1, 0}, {0, 1, 0, 0},
 {0, 0, 1, 0}, {0, 0, 0, 1}, {-1, 1, -1, 1}, {4, -1, 2, 1}}
columns: {1, 4, 5, 7}   objective = 5   x = (5,0,0,-2,3,0,1)
{{1, 0, 0, 0}, {0, 1, -1, 0}, {1, -1, 1, 0}, {0, 1, 0, 0},
 {0, 0, 1, 0}, {1, -1, 1, 1}, {0, 0, 0, 1}, {5, -2, 3, 1}}
columns: {1, 4, 6, 7}   objective = 2   x = (2,0,0,1,0,3,-2)
{{1, 0, 0, 0}, {1, 0, -1, 1}, {0, 0, 1, -1}, {0, 1, 0, 0},
 {-1, 1, 1, -1}, {0, 0, 1, 0}, {0, 0, 0, 1}, {2, 1, 3, -2}}
columns: {1, 5, 6, 7}   objective = 3   x = (3,0,0,0,1,2,-1)
{{1, 0, 0, 0}, {1, 0, -1, 1}, {0, 0, 1, -1}, {1, 1, -1, 1},
 {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}, {3, 1, 2, -1}}
columns: {2, 4, 6, 7}   objective = 4   x = (0,2,0,1,0,5,-4)
{{1, 0, 1, -1}, {1, 0, 0, 0}, {0, 0, 1, -1}, {0, 1, 0, 0},
 {-1, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}, {2, 1, 5, -4}}
columns: {2, 5, 6, 7}   objective = 6   x = (0,3,0,0,1,5,-4)
{{1, 0, 1, -1}, {1, 0, 0, 0}, {0, 0, 1, -1}, {1, 1, 0, 0},
 {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}, {3, 1, 5, -4}}
columns: {3, 4, 5, 6}   objective = 12   x = (0,0,4,3,-2,1,0)
{{1, 1, -1, 0}, {0, 1, -1, 0}, {1, 0, 0, 0}, {0, 1, 0, 0},
 {0, 0, 1, 0}, {0, 0, 0, 1}, {-1, 0, 0, 1}, {4, 3, -2, 1}}
columns: {3, 4, 5, 7}   objective = 15   x = (0,0,5,3,-2,0,1)
{{1, 1, -1, 0}, {0, 1, -1, 0}, {1, 0, 0, 0}, {0, 1, 0, 0},
 {0, 0, 1, 0}, {1, 0, 0, 1}, {0, 0, 0, 1}, {5, 3, -2, 1}}
columns: {4, 5, 6, 7}   objective = 0   x = (0,0,0,3,-2,5,-4)
{{1, -1, 1, -1}, {1, -1, 0, 0}, {0, 0, 1, -1}, {1, 0, 0, 0},
 {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}, {3, -2, 5, -4}}
```

# Category 3: 14 Not Basic Solutions

{1, 2, 3, 4}, {1, 2, 3, 5}, {1, 2, 3, 6}, {1, 2, 3, 7}, {1, 2, 4, 5}, {1, 2, 6, 7}, {1, 3, 4, 5}, {1, 3, 6, 7}, {2, 3, 4, 5}, {2, 3, 6, 7}, {2, 4, 5, 6}, {2, 4, 5, 7}, {3, 4, 6, 7}, {3, 5, 6, 7}

# Linear Program

$$\max\ c_1x_1 + c_2x_2 + \ldots + c_Nx_N$$

subject to

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1N}x_N \le b_1$$
$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2N}x_N \le b_2$$
$$\ldots$$
$$a_{M1}x_1 + a_{M2}x_2 + \ldots + a_{MN}x_N \le b_M$$
$$x_j \ge 0,\ j = 1..N$$

**Concise version:**

$$\max\ c'x$$
$$\text{subject to } Ax \le b$$
$$x \ge 0$$

**A is an *m by n* matrix: n variables, m constraints**

# Linear Programming in Standard Form

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}^T \mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0}. \end{aligned}$$

$\{\mathbf{x} : \mathbf{x} \geq \mathbf{0}\}$ is the non-negative authant cone.

# Reduction to Standard Form

- Eliminating "free" variable: use the difference of two nonnegative variables

$$x = x^+ - x^-, \quad x^+, x^- \geq 0.$$

- Eliminating inequality: add slack variable

$$\mathbf{a}^T\mathbf{x} \leq b \Longrightarrow \mathbf{a}^T\mathbf{x} + s = b, \quad s \geq 0$$

$$\mathbf{a}^T\mathbf{x} \geq b \Longrightarrow \mathbf{a}^T\mathbf{x} - s = b, \quad s \geq 0$$

- Eliminating upper bound: move them to constraints

$$x \leq 3 \Longrightarrow x + s = 3, \quad s \geq 0$$

- Eliminating nonzezro lower bound: shift the decision variables

$$x \geq 3 \Longrightarrow x := x - 3$$

- Change $\max\ \mathbf{c}^T\mathbf{x}$ to $\min\ -\mathbf{c}^T\mathbf{x}$

# An Example with 35 (possible) bases

An example with eight basic feasible solutions

Consider the LP:

min or max

$$x_1 + 2x_2 + 3x_3$$

subject to

$$2 \le x_1 + x_2 \le 3$$

$$4 \le x_1 + x_3 \le 5$$

$$x_1 \ge 0, \quad x_2 \ge 0, \quad x_3 \ge 0$$

In standard form, we write this as

minimize or maximize $\quad x_1 + 2x_2 + 3x_3 = \text{objective}$

$$\begin{array}{lllllllll}
\text{subject to} & x_1 + x_2 & + x_4 & & & & = & 3 \\
& x_1 + x_2 & & - x_5 & & & = & 2 \\
& x_1 & + x_3 & & + x_6 & & = & 5 \\
& x_1 & + x_3 & & & - x_7 & = & 4
\end{array}$$

$$x_1 \ge 0,\ x_2 \ge 0,\ x_3 \ge 0,\ x_4 \ge 0,\ x_5 \ge 0,\ x_6 \ge 0,\ x_7 \ge 0$$

There are $\binom{7}{4} = 35$ ways to choose four columns from the $4 \times 7$ coefficient matrix. Each fits into one of 3 categories:

i. The 4 columns do form a basis and the corresponding basic solution is feasible (all variables are nonnegative).

ii. The 4 columns do form a basis (the $4 \times 4$ matrix is invertible) but the corresponding basic solution is infeasible (one variable is negative).

iii. The corresponding 4 columns of the coefficient matrix form a singular (not invertible) $4 \times 4$ matrix. In other words, these columns do not form a basis.

# An Example with 35 (possible) bases

An example with eight basic feasible solutions

Consider the LP:

min or max

$$x_1 + 2x_2 + 3x_3$$

subject to

$$2 \le x_1 + x_2 \le 3$$

$$4 \le x_1 + x_3 \le 5$$

$$x_1 \ge 0, \quad x_2 \ge 0, \quad x_3 \ge 0$$

In standard form, we write this as

minimize or maximize $x_1 + 2x_2 + 3x_3$ = objective

$$\begin{array}{llllllllll} \text{subject to} & x_1 + x_2 & & + x_4 & & & & = & 3 \\ & x_1 + x_2 & & & - x_5 & & & = & 2 \\ & x_1 & + x_3 & & & + x_6 & & = & 5 \\ & x_1 & + x_3 & & & & - x_7 & = & 4 \end{array}$$

$$x_1 \ge 0, \ x_2 \ge 0, \ x_3 \ge 0, \ x_4 \ge 0, \ x_5 \ge 0, \ x_6 \ge 0, \ x_7 \ge 0$$

There are $\binom{7}{4} = 35$ ways to choose four columns from the $4 \times 7$ coefficient matrix. Each fits into one of 3 categories:

i. The 4 columns do form a basis and the corresponding basic solution is feasible (all variables are nonnegative).

ii. The 4 columns do form a basis (the $4 \times 4$ matrix is invertible) but the corresponding basic solution is infeasible (one variable is negative).

iii. The corresponding 4 columns of the coefficient matrix form a singular (not invertible) $4 \times 4$ matrix. In other words, these columns do not form a basis.

# Reduce Search to Basic Solutions

**Optimal feasible solution$\subset$ feasible solutions $\subset$ basic solutions**

This theorem reduces the task of solving a linear program to that of searching over basic feasible solutions. Since for a problem having $n$ variables and $m$ constraints there are at most

$$\binom{n}{m} = \frac{n!}{m!(n-m)!}$$

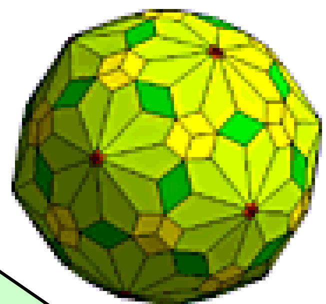
basic solutions (corresponding to the number of ways of selecting $m$ of $n$ columns), there are only a finite number of possibilities. Thus the fundamental theorem yields an obvious, but terribly inefficient, finite search technique. By expanding upon the technique of proof as well as the statement of the fundamental theorem, the efficient simplex procedure is derived.

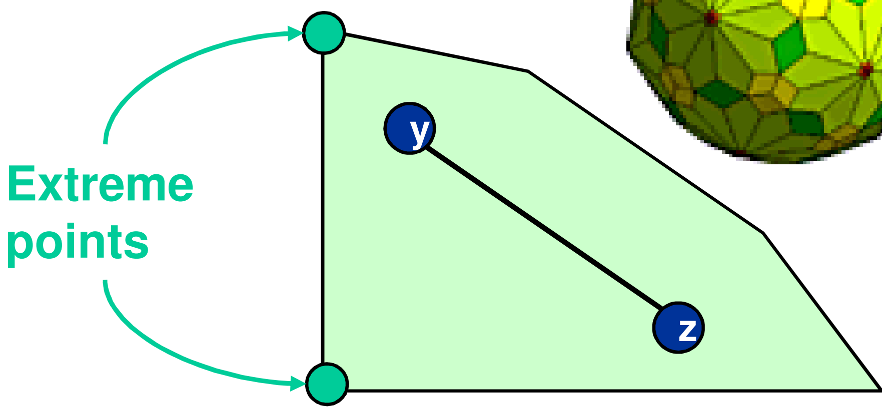
**n variables (including slacks)**
**m Constraints**

# LP: Geometry

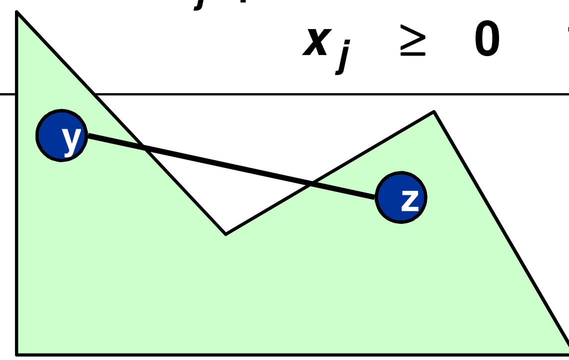
- **Geometry.**
  - Forms an n-dimensional polyhedron/polytope.

$$(P)\quad \max \sum_{j=1}^{n} c_j x_j$$
$$\text{s. t.} \quad \sum_{j=1}^{n} a_{ij} x_j \le b_i \quad 1 \le i \le m$$
$$x_j \ge 0 \quad 1 \le j \le n$$






**Convex**




**Not convex**

  - Convex: if y and z are feasible solutions, then so is ½y + ½z.
  - Extreme point: feasible solution x that can't be written as ½y + ½z for any two distinct feasible solutions y and z.

# Connecting Algebra with Geometry

***Theorem.*** (Equivalence of extreme points and basic solutions). *Let* $\mathbf{A}$ *be an* $m \times n$ *matrix of rank* $m$ *and* $\mathbf{b}$ *an* $m$*-vector. Let* $K$ *be the convex polytope consisting of all* $n$*-vectors* $\mathbf{x}$ *satisfying*

$$\begin{aligned} \mathbf{Ax} &= \mathbf{b} \\ \mathbf{x} &\geqslant \mathbf{0}. \end{aligned} \tag{17}$$

*A vector* $\mathbf{x}$ *is an extreme point of* $\boldsymbol{K}$ *if and only if* $\mathbf{x}$ *is a basic feasible solution to (17).*

# Wyndor Glass (Hillier and Lieberman)

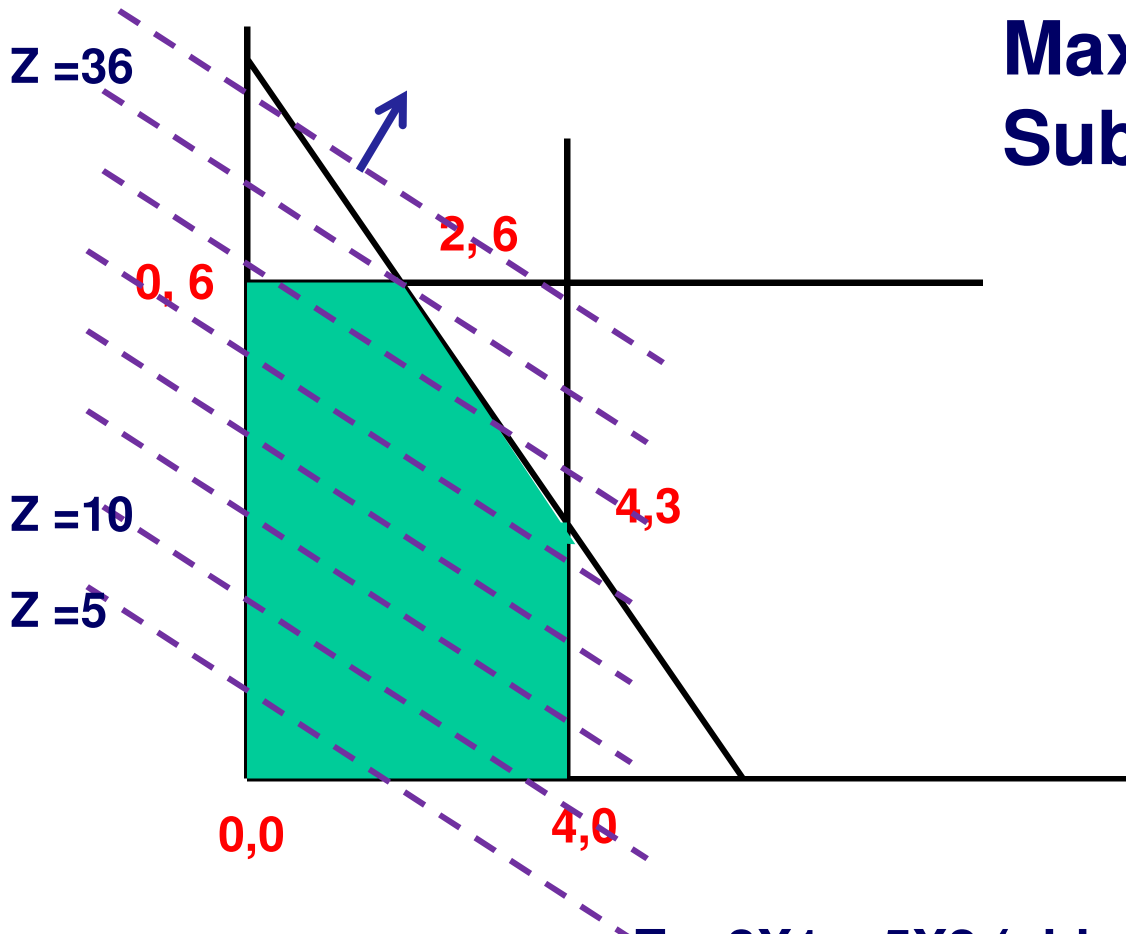
**Maximize $Z = 3X1 + 5X2$**

**Subject to:**

$$
\begin{aligned}
X1 \quad\quad\quad\quad &\leq 4 \\
2X2 &\leq 12 \\
3X1 + 2X2 &\leq 18
\end{aligned}
$$

# Wyndor Glass Example

## Extreme Point Solution




**Maximize Z = 3X1 + 5X2**
**Subject to:**

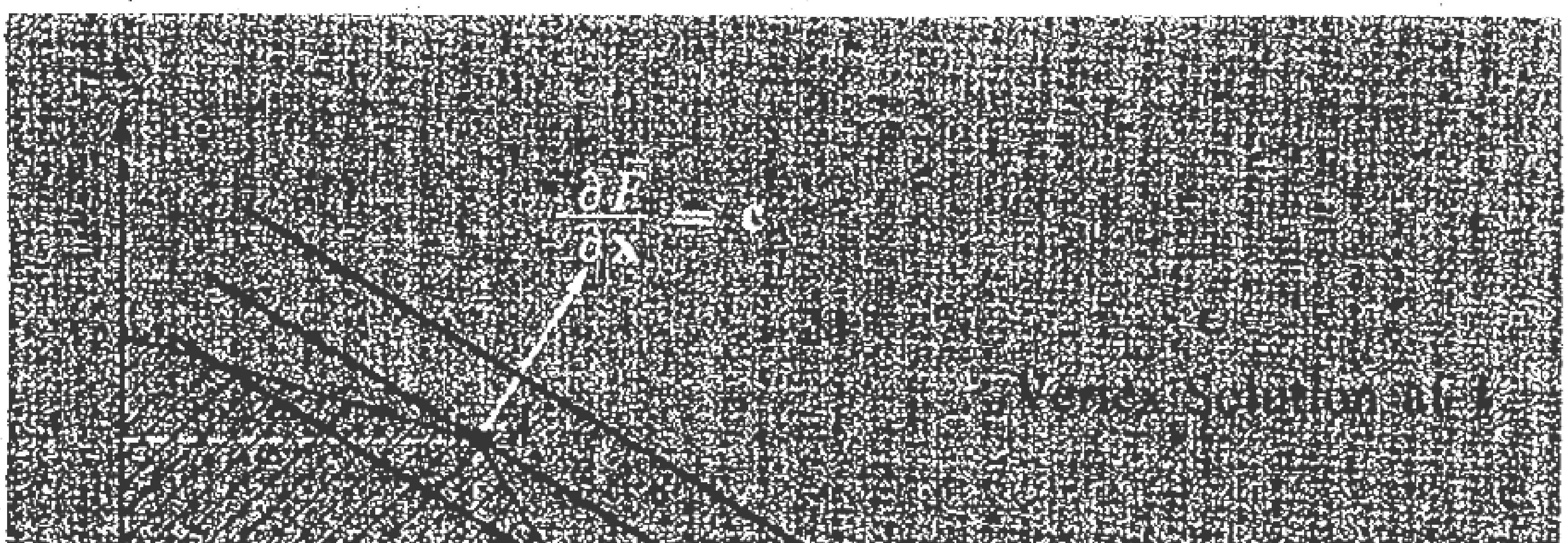
| | | | |
|---|---|---|---|
| X1 | | ≤ | 4 |
| | 2X2 | ≤ | 12 |
| 3X1 + | 2X2 | ≤ | 18 |

**Corners:**
**(0,0) (4,0) (4.3)**
**(2,6) and (0,6)**

**Z = 3X1 + 5X2 (objective hyperplane)**

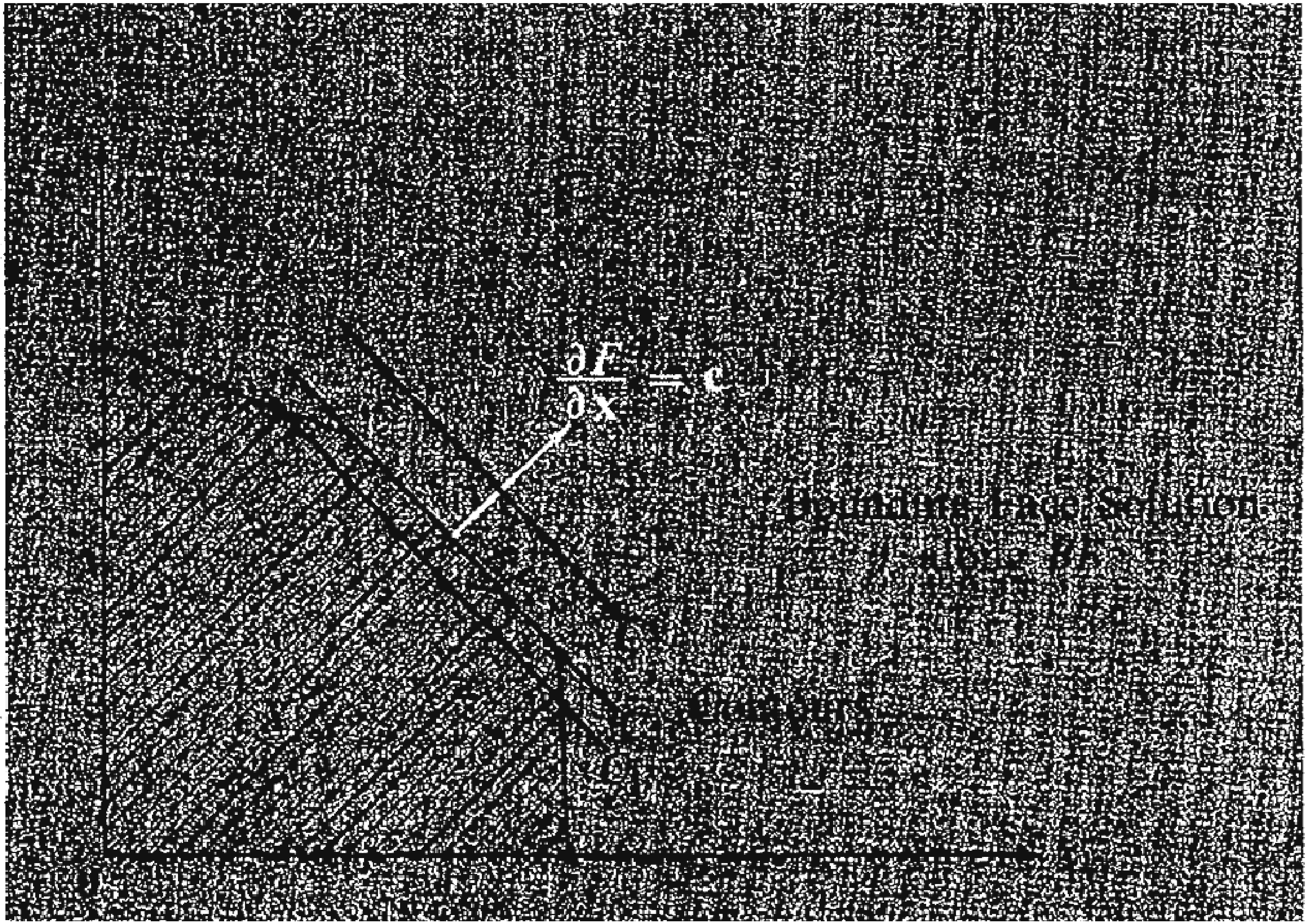
# Extreme Point Solution

## No interior solutions in LPs




Finally, Fig. 2.5 illustrates two possible solutions to the linear programming problem (2.1.18). The linear objective function gives rise to linear contours, defined by the $C_k$, and the linear inequality constraints and non-negativity constraints give rise to the shaded opportunity set bounded by linear segments. Since the objective function is linear $\partial F/\partial \mathbf{x} = \mathbf{c}$, the direction of steepest ascent is the same everywhere. For this reason there cannot be an interior solution: the solution is either at a vertex ($V$) or along a bounding face ($BF$) of the opportunity set.

# Bounding Face Solution



# How solve an LP

- **Extreme point based approaches**
  - Simplex
- **Or interior point approaches**
  - Barrier algorithm

# Fundamental Theorem of LP and Simplex

**Theorem 2** *(LP fundamental theorem in Algebraic form) Given (LP) and (LD) where $A$ has full row rank $m$,*

**i)** *if there is a feasible solution, there is a basic feasible solution;*

**ii)** *if there is an optimal solution, there is an optimal basic solution.*

The simplex method is to proceed from one BFS (a corner point of the feasible region) to an adjacent or neighboring one, in such a way as to continuously improve the value of the objective function.

# Simplex Method

- **The idea of the simplex method is to proceed from one basic feasible solution (that is, one extreme point) of the constraint set of a problem in standard form to another, in such a way as to continually decrease the value of the objective function until a minimum is reached.**
- **The Fundamental Theorem of LP assure us that it is sufficient to consider**
  - only basic feasible solutions in our search for an optimal feasible solution.
- **The Simplex method is an efficient method for moving among basic solutions to the minimum.**

# Systems of Equations

If the first m columns of A are linearly independent then the system can be reduced to a canonical form through multiples of equations being added/subtracted to one another

**Standard Form**

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \ldots + a_{1N}x_N = b_1 \\ &a_{21}x_1 + a_{22}x_2 + \ldots + a_{2N}x_N = b_2 \\ &\ldots \\ &a_{M1}x_1 + a_{M2}x_2 + \ldots + a_{MN}x_N = b_M \end{aligned}$$

**Canonical Form**

$$\begin{array}{ccccccccc} x_1 & & & + y_{1,m+1}x_{m+1} & + y_{1,m+2}x_{m+2} & + \ldots & + y_{1,n}x_n & = & y_{1,0} \\ & x_2 & & + y_{2,m+1}x_{m+1} & + y_{2,m+2}x_{m+2} & & + y_{2,n}x_n & = & y_{2,0} \\ & & & . & & & & & . \\ & & & . & & & & & . \\ & & & . & & & & & . \\ & & x_m & + y_{m,m+1}x_{m+1} & + y_{m,m+2}x_{m+2} & & + y_{m,n}x_n & = & y_{m,0} \end{array}$$

**Matrix Form**

$$\boxed{Ax = b}$$

# Basic vs.non-Basic &Canonical Form

- System of equations in canonical form:

$$
\begin{array}{lllllllll}
x_1 & & & + y_{1,m+1}x_{m+1} & + y_{1,m+2}x_{m+2} & + \dots & + y_{1,n}x_n & = & y_{1,0} \\
 & x_2 & & + y_{2,m+1}x_{m+1} & + y_{2,m+2}x_{m+2} & & + y_{2,n}x_n & = & y_{2,0} \\
 & & & & & & & & \\
 & & x_m & + y_{m,m+1}x_{m+1} & + y_{m,m+2}x_{m+2} & & + y_{m,n}x_n & = & y_{m,0}
\end{array}
$$

- Corresponding to this canonical representation of the system, the variables $x_1, x_2, \dots, x_m$ are called *basic and the other variables are nonbasic. The corresponding* basic solution is then:
- $X_1 = y_{10}$, $x2 = y_{20}$, $x_m = y_{m0}$ $x_{m+1} = 0, \dots,$ $x_n = 0$
- or in vector form: $x = (y_0, 0)$ where $y_0$ is *m-n*-dimensional and *0* is the *n–m*-dimensional zero vector.

# Pivoting and Canonical Form

- **Or in terms of the corresponding array of coefficients (or tableau):**

$$
\begin{array}{ccccccccc}
1 & 0 & \ldots. & & +y_{1,m+1} & +y_{1,m+2} & +\ldots & +y_{1,n} & = & y_{1,0} \\
0 & 1 & \ldots. & & +y_{2,m+1} & +y_{2,m+2} & & +y_{2,n} & = & y_{2,0} \\
0 & 0 & \ldots. & & & & & & & \\
 & & & & & & & & & \\
0 & 0 & \ldots. & 1 & +y_{m,m+1} & +y_{m,m+2} & & +y_{m,n} & = & y_{m,0}
\end{array}
$$

- **Use pivoting as a means to change the basis of this system (in canonical form)**
  - what is the new canonical form corresponding to the new set of basic variables?

# Pivoting: entering and leaving

- **Suppose we have a system of equations in canonical form where we wish to replace the basic variable $x_p$, $1 \le p \le m$ by the non-basic variable $x_q$, $m+1 \le q \le n$**

Let q=n

Let p=2

$$
\begin{array}{cccccccccc}
1 & 0 & \ldots. & & +\,y_{1,m+1} & +\,y_{1,m+2} & +\ldots & +\,y_{1,n} & = & y_{1,0} \\
0 & 1 & \ldots. & & +\,y_{2,m+1} & +\,y_{2,m+2} & & \boxed{+\,y_{2,n}} & = & y_{2,0} \\
0 & 0 & \ldots. & & & & & & & \\
 & & & & & & & & & \\
0 & 0 & \ldots. & 1 & +\,y_{m,m+1} & +\,y_{m,m+2} & & +\,y_{m,n} & = & y_{m,0}
\end{array}
$$

pivot

- **Can only happen if $y_{pq}$ is non-zero ($y_{2,n}$ in our case);**
- **it is accomplished by:**
  - dividing row p by $y_{pq}$ to get a unit coefficient for xq in the pth equation,
  - and then subtracting suitable multiples of row p from each of the other rows in order to get a zero coefficient for xq in all other equations.
- **This transforms**
  - the qth column of the tableau so that it is zero except in its pth entry (which is unity)
  - and does not affect the columns of the other basic variables.

**Example 1.** Consider the system in canonical form:

$$\begin{aligned} x_1 \quad\quad\quad &+ x_4 + x_5 - x_6 = 5 \\ x_2 \quad\quad &+ 2x_4 - 3x_5 + x_6 = 3 \\ x_3 &- x_4 + 2x_5 - x_6 = -1. \end{aligned}$$

Let us find the basic solution having basic variables $x_4, x_5, x_6$. We set up the coefficient array below:

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | −1 | 5 |
| 0 | 1 | 0 | 2 | −3 | 1 | 3 |
| 0 | 0 | 1 | −1 | 2 | −1 | −1 |

The circle indicated is our first pivot element and corresponds to the replacement of $x_1$ by $x_4$ as a basic variable. After pivoting we obtain the array

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | −1 | 5 |
| −2 | 1 | 0 | 0 | −5 | 3 | −7 |
| 1 | 0 | 1 | 0 | 3 | −2 | 4 |

and again we have circled the next pivot element indicating our intention to replace $x_2$ by $x_5$. We then obtain

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | |
|---|---|---|---|---|---|---|
| 3/5 | 1/5 | 0 | 1 | 0 | −2/5 | 18/5 |
| 2/5 | −1/5 | 0 | 0 | 1 | −3/5 | 7/5 |
| −1/5 | 3/5 | 1 | 0 | 0 | −1/5 | −1/5 |

Continuing, there results

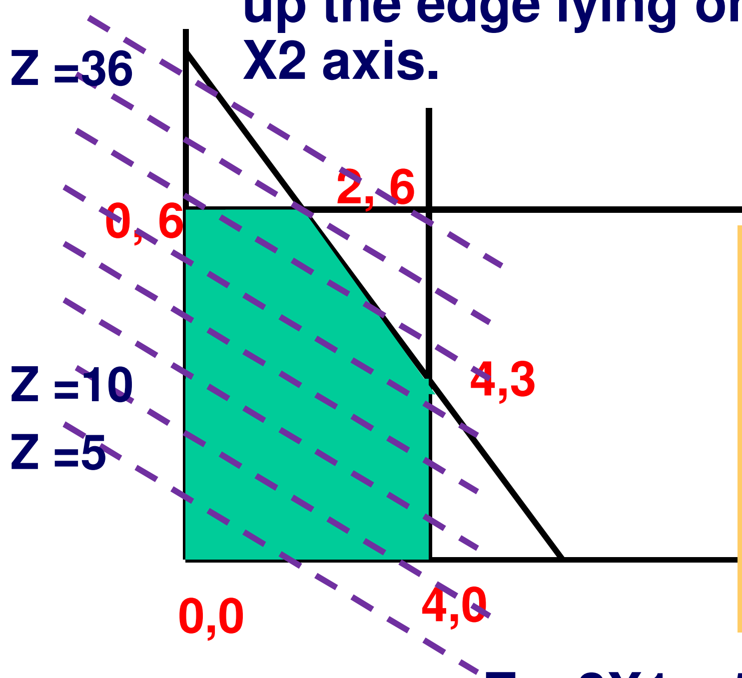
| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | |
|---|---|---|---|---|---|---|
| 1 | −1 | −2 | 1 | 0 | 0 | 4 |
| 1 | −2 | −3 | 0 | 1 | 0 | 2 |
| 1 | −3 | −5 | 0 | 0 | 1 | 1 |

From this last canonical form we obtain the new basic solution

$$x_4 = 4, \qquad x_5 = 2, \qquad x_6 = 1.$$

# Geometric | Algebraic

- Choose (0,0) as initial CPF solution.
- Optimality test: not optimal because moving along either edge increases Z.
- Iteration 1, step 1: Move up the edge lying on the X2 axis.




- Choose X1and X2 to be non-basic for initial BFS (0,0,4,12,18)
- Not optimal because increasing either non-basic variable increases Z.
- Iteration 1, step 1: Increase X2 while adjusting other variable values to satisfy the system of equations.

Maximize $Z = 3X1 + 5X2 + 0X3 + 0X4 + 0X5$

Subject to:

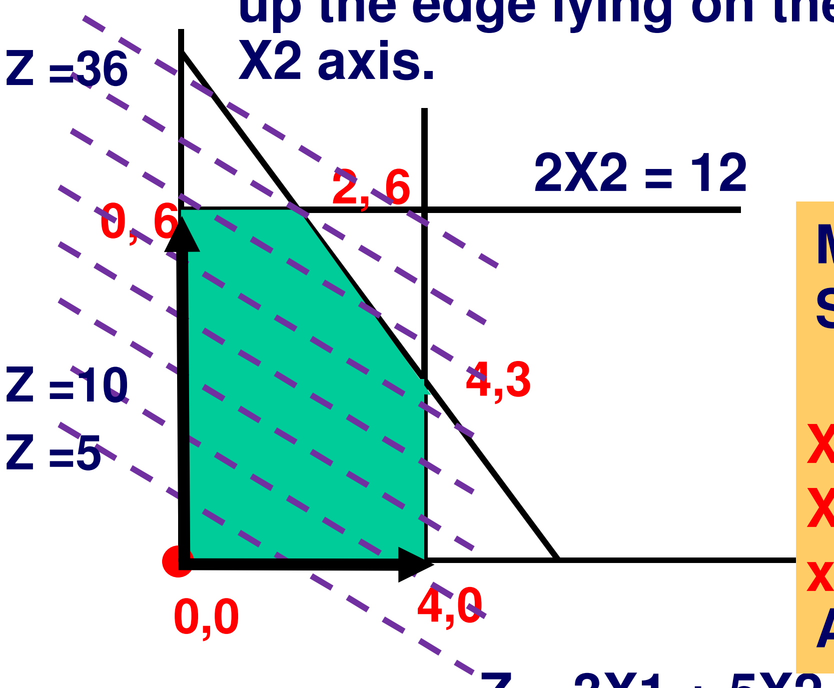
$$\begin{aligned} X1 \quad\quad &+ X3 &&= 4 \\ 2X2 &\quad + X4 &&= 12 \\ 3X1 + 2X2 &\quad\quad + X5 &&= 18 \end{aligned}$$

# Geometric / Algebraic

- Choose (0,0) as initial CPF solution.
- Optimality test: not optimal because moving along either edge increases Z.
- Iteration 1, step 1: Move up the edge lying on the X2 axis.

- Choose X1and X2 to be non-basic for initial BFS (0,0,4,12,18)
- Not optimal because increasing either <u>non-basic variable</u> increases Z.
- Iteration 1, step 1: Increase X2 while adjusting other variable values to satisfy the system of equations.

Gradient




**Maximize Z** $\quad Z = 3X_1 + 5X_2$

**Subject to:**

| | | | |
|---|---|---|---|
| | $Z - 3X_1 - 5X_2 - 0X_3 - 0X_4 - 0X_5$ | = | 0 |
| X3 | $X_1 + X_3$ | = | 4 |
| X4 | $\boxed{2X_2 + X_4 = 12}$ | | |
| x5 | $3X_1 + 2X_2 + X_5$ | = | 18 |

And $X_j \geq 0$ for j=1, …, 5

# Simplex Method

- The Simplex method is a matrix procedure for solving linear programs in standard form:

  *optimize $C^T x$*

  *subject to $Ax=b$ with $x \geq 0$ and* $b \geq 0$

  where a basic feasible solution $x_0$ is known.

- The Simplex method is a method that proceeds from one BFS or extreme point of the feasible region of an LP problem expressed in tableau form to another neighboring BFS, in such a way as to continually increase (or decrease) the value of the objective function until optimality is reached.
- For maximization programs, the simplex utilizes a tableau in which $C_o$ designates the cost vector associated with the variables $X_{0:}X_0$ is the basis

| Minimize | $x^T$ <br> $c^T$ | |
|---|---|---|
| $x_0$ , $c_0$ | A | b |
| | $C^T – C_0^T A$ | $-– C_0^T b$ |

| Maximize | $x^T$ <br> $c^T$ | |
|---|---|---|
| $x_0$ , $c_0$ | A | b |
| | $C_0^T b – C^T$ | $C_0^T b$ |

# Duality

- **Provides an alternative/dual LP (introduced in 1940s)**
- **Dual algorithms**
- **When both LP problems have feasible vectors, they have optimal $x^*$ and $y^*$. The minimum cost cx* equals the maximum income y*b. If yb =cx then x and y are optimal.  [Duality Theorem]**
- **If x and y are feasible in the primal and dual problems then yb ≤cx [weak duality].**
- **Provides a means to conduct sensitivity analysis easily**
  - Resource amounts can be estimates; so maybe want to engage in a what-if analysis

# Duality

*Primal Problem*

Maximize $\quad Z = \sum_{j=1}^{n} c_j x_j,$

subject to

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \qquad \text{for } i = 1, 2, \ldots, m$$

and

$$x_j \geq 0, \qquad \text{for } j = 1, 2, \ldots, n.$$

*Dual Problem*

Minimize $\quad W = \sum_{i=1}^{m} b_i y_i,$

subject to

$$\sum_{i=1}^{m} a_{ij} y_i \geq c_j, \qquad \text{for } j = 1, 2, \ldots, n$$

and

$$y_i \geq 0, \qquad \text{for } i = 1, 2, \ldots, m.$$

*Primal Problem*

Maximize $\quad Z = \mathbf{cx},$

subject to

$$\mathbf{Ax} \leq \mathbf{b}$$

and

$$\mathbf{x} \geq \mathbf{0}.$$

*Dual Problem*

Minimize $\quad W = \mathbf{yb},$

subject to

$$\mathbf{yA} \geq \mathbf{c}$$

and

$$\mathbf{y} \geq \mathbf{0}.$$

**TABLE 6.1** Primal and dual problems for the Wyndor Glass Co. example

| *Primal Problem in Algebraic Form* | *Dual Problem in Algebraic Form* |
|---|---|
| Maximize $Z = 3x_1 + 5x_2,$ subject to $x_1 \le 4$, $2x_2 \le 12$, $3x_1 + 2x_2 \le 18$ and $x_1 \ge 0, \quad x_2 \ge 0.$ | Minimize $W = 4y_1 + 12y_2 + 18y_3,$ subject to $y_1 + 3y_3 \ge 3$, $2y_2 + 2y_3 \ge 5$ and $y_1 \ge 0, \quad y_2 \ge 0, \quad y_3 \ge 0.$ |

| *Primal Problem in Matrix Form* | *Dual Problem in Matrix Form* |
|---|---|
| Maximize $Z = [3, 5]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$ subject to $\begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 3 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \le \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix}$ and $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \ge \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$ | Minimize $W = [y_1, y_2, y_3]\begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix}$ subject to $[y_1, y_2, y_3]\begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 3 & 2 \end{bmatrix} \ge [3, 5]$ and $[y_1, y_2, y_3] \ge [0, 0, 0].$ |

# LP Outline

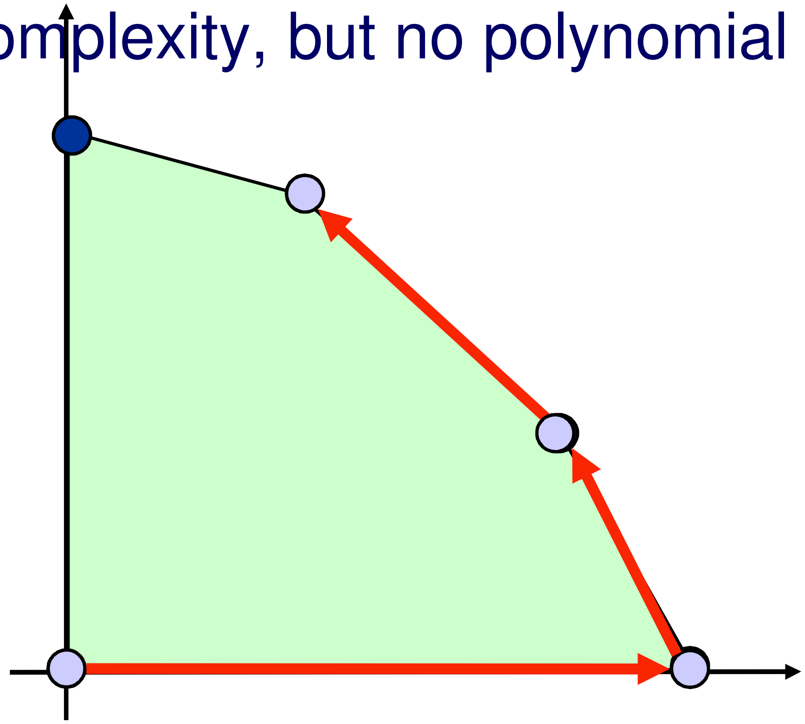
- Introduction and some motivating advertising problems
- Linear Algebra Basics Review
- Fundamental theorem of LP
- Matrix-view and the fundamental insight
- Duality
- Interior point Algorithm
- Transportation Problem
- Applying linear programming to online advertising
- Summary

# Interior Point Solution

- Starts from inside the feasible region
- Moves along a path from the interior to the boundary
- Large problems can be solved more efficiently

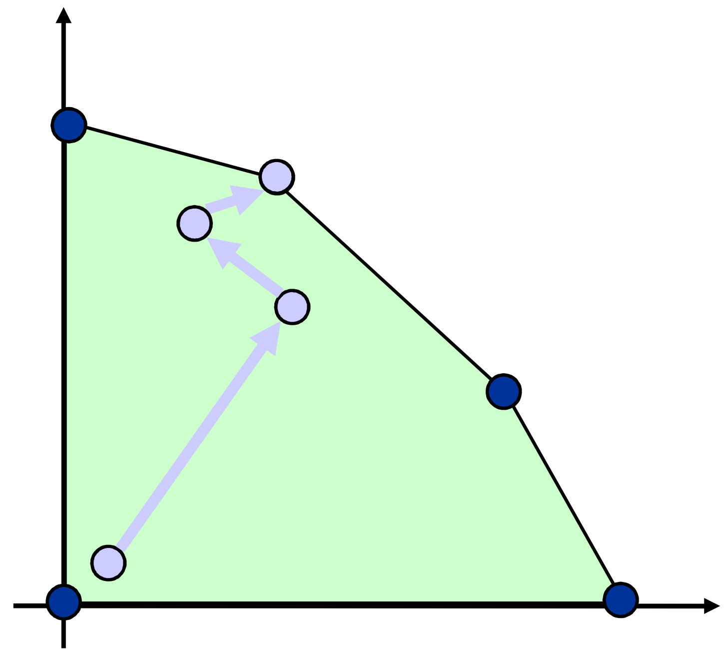
# LP:  Algorithms

- **Simplex. (Dantzig 1947)**
  - Developed shortly after WWII in response to logistical problems: used for 1948 Berlin airlift.
  - Practical solution method  that moves from one extreme point to neighboring extreme point.
  - Finite (exponential) complexity, but no polynomial implementation known.



# LP: Polynomial Algorithms

- **Ellipsoid. (Khachian 1979, 1980)**
  - Solvable in polynomial time: $O(n^4 L)$ bit operations.
    - n = # variables
    - L = # bits in input
  - Theoretical tour de force.
  - Not remotely practical.
- **Karmarkar's algorithm. (Karmarkar 1984)**
  - $O(n^{3.5} L)$.
  - Polynomial and reasonably efficient implementations possible.
- **Interior point algorithms.**
  - $O(n^3 L)$.
  - Competitive with simplex!
    - will likely dominate on large problems soon
  - Extends to even more general problems.



# LP Outline

- Introduction and some motivating advertising problems
- Linear Algebra Basics Review
- Fundamental theorem of LP
- Matrix-view and the fundamental insight
- Duality
- Interior point Algorithm
- Transportation Problem
- Applying linear programming to online advertising
- Summary

# Transportation Problem Description

**A transportation problem basically deals with the problem, which aims to find the best way to fulfill the demand of n demand points using the capacities of m supply points. While trying to find the best way, generally a variable cost of shipping the product from one supply point to a demand point or a similar constraint should be taken into consideration.**

# Linear Programming Summary

**Linear programs are problems that can be expressed in canonical form:**

**Maximize XXXX**

**Subject to f(x)….**

**Problem 1.** Maximize

$$\sum_{i=1}^{n} \sum_{j=1}^{m} c_{i,j} k_i d_{i,j} \tag{1}$$

under the conditions

$$\sum_{i=1}^{n} k_i d_{i,j} = h_j \quad \text{for } j = 1, \ldots, m, \tag{2}$$

$$\sum_{j=1}^{m} d_{i,j} = 1 \quad \text{for } i = 1, \ldots, n, \tag{3}$$

$$d_{i,j} \geq 0 \quad \text{for } i = 1, \ldots, n, \; j = 1, \ldots, m. \tag{4}$$

# LP Algorithms Summary

- **Many algorithms can be used to solve the LP**
- **Simplex algorithm (most popular)**
  - Searches for an optimal solution by moving from one basic solution to another, along the edges of the feasible polygon, in direction of cost decrease (Graphically, moves from corner to corner)
- **Interior Point Methods (more recent)**
  - Approaches the situation through the interior of the convex polygon
  - Affine Scaling
  - Log Barrier Methods
  - Primal-dual methods
- **Bounded regions and corner points**

# Scheduling Web Adveristisments

- **Predictive Clustering + Linear Programming = Web Adverstisment Scheduler**
  – Partition the world of “webpages X users X Ads” as it is sparse
  – Schedule which ads get displayed

- **Limited context to show ads**
- **Many advertisers want their ads shown and are willing to pay**

- **Maximize profit (or some proxy for profit) given limited realestate (contexts) and many ads.**

# Sample Problem

Before describing the formalization, we first show an example which helps illustrate the problem and the need for an LP solution. Assume that the accurately estimated numbers of page views for combinations of attribute values (afternoon, sports), (afternoon, not sports), (not afternoon, sports) and (not afternoon, not sports) are 10,000, 10,000, 5,000 and 5,000, respectively. Also assume that there are three ads for each of which 10,000 impressions have been promised, and that the accurately estimated click-through rates of these ads for the combinations of attribute values are as shown in Table 1.

*Table 1.* Case that the local strategy fails.

| Time of day | Page category | Number of page views | Click-through rate (%) | | |
|---|---|---|---|---|---|
| | | | ad 1 | ad 2 | ad 3 |
| afternoon | sports | 10,000 | 2.2 | 1.1 | 1.0 |
| afternoon | not sports | 10,000 | 2.2 | 2.1 | 1.0 |
| not afternoon | sports | 5,000 | 2.2 | 2.1 | 2.0 |
| not afternoon | not sports | 5,000 | 2.2 | 2.1 | 2.0 |

http://www.research.ibm.com/people/n/nabe/JECR05-NA.pdf

# Greedy vs Random Vs LP

- **Assume that page views for all combinations of attribute values occur randomly.**
- **The greedy strategy always selects ad 1 for the first 10,000**
- **page views, ad 2 for the second 10,000 page views and ad 3 for the last 10,000 page views, because *(the click-through rate of ad 1) > (the click-through rate of ad 2) > (the click-through rate of ad 3) holds for all combinations of attribute values.***
  - *As a result,* we find that the actual click-through rates for ad 1, ad 2 and ad 3 are 2.2%, 1*.76 . . .% and* 1*.33 . . .%,*
  - *the total click-through rate for all ads is **1.76**%, which is the same rate* as what would be obtained by the **random selection** strategy.
- **According to the optimal display schedule in the LP model, click-through rate is 2.1%**

| CTR Ad1 | CTR Ad2 | CTR Ad3 | AvgAdCtr | proportion of impressions | sumproduct (CTRAd*proportion) |
|---|---|---|---|---|---|
| 2.2 | 1.1 | 1 | 1.433333 | 0.333333 | 1.766667 |
| 2.2 | 2.1 | 1 | 1.766667 | 0.333333 | |
| 2.2 | 2.1 | 2 | 2.1 | 0.166667 | |
| 2.2 | 2.1 | 2 | 2.1 | 0.166667 | |

| CTR Ad2 | proportion of impressions | sumproduct(CTRAd2*proportion2) |
|---|---|---|
| 1.1 | 0.333333 | **1.766667** |
| 2.1 | 0.333333 | |
| 2.1 | 0.166667 | |
| 2.1 | 0.166667 | |

# Transportation Problem Description

A transportation problem basically deals with the problem, which aims to find the best way to fulfill the demand of *n* demand points using the capacities of *m* supply points.

While trying to find the best way, generally a variable cost of shipping the product from one supply point to a demand point or a similar constraint should be taken into consideration.

# Maximize Revenue: Ad Allocation Example

| ***From*** | ***To*** | | | | |
|---|---|---|---|---|---|
| | ***Ad1*** | ***Ad2*** | ***..$Ad_j$.....*** | ***$Ad_m$*** | ***Supply PageViews*** |
| ***PubZone 1*** | $CTR_{ij}$ | $CTR_{ij}$ | $CTR_{ij}$ | $CTR_{ij}$ | 35 |
| ***PubZone 2*** | … | … | … | … | 50 |
| ***PubZone3*** | $CTR_{3j}$ | $CTR_{3j}$ | $CTR_{3j}$ | $CTR_{3j}$ | 15 |
| ***Demand Contracted PageViews*** | *45* | *20* | *30* | *5* | |

**Given this Transportation Tableau generate the ad display schedule (explore R's lp_solve)**

# Maximize Revenue: Ad Allocation Example

| ***From*** | ***To*** | | | | |
|---|---|---|---|---|---|
| | ***Ad1*** | ***Ad2*** | ***..Ad$_j$.....*** | ***Ad$_m$*** | ***Supply PageViews*** |
| ***PubZone 1*** | $d_{ij}$ | $d_{ij}$ | $d_{ij}$ | $d_{ij}$ | 35 |
| ***PubZone 2*** | … | … | … | … | 50 |
| ***PubZone3*** | $d_{ij}$ | $d_{ij}$ | $d_{ij}$ | $d_{ij}$ | 15 |
| ***Demand Contracted PageViews*** | *45* | *20* | *30* | *5* | |

**Use LP to generate the Ad display schedule to maximize my revenue (or rev proxy, .i.e., CTR)**

# LP Formulation of Powerco's Problem

$$\text{Min } Z = 8X_{11} + 6X_{12} + 10X_{13} + 9X_{14} + 9X_{21} + 12X_{22} + 13X_{23} + 7X_{24} + 14X_{31} + 9X_{32} + 16X_{33} + 5X_{34}$$

S.T.: $X_{11}+X_{12}+X_{13}+X_{14} <= 35$ (Supply Constraints)

$X_{21}+X_{22}+X_{23}+X_{24} <= 50$

$X_{31}+X_{32}+X_{33}+X_{34} <= 40$

$X_{11}+X_{21}+X_{31} >= 45$ (Demand Constraints)

$X_{12}+X_{22}+X_{32} >= 20$

$X_{13}+X_{23}+X_{33} >= 30$

$X_{14}+X_{24}+X_{34} >= 30$

$Xij >= 0$ ($i= 1,2,3$; $j= 1,2,3,4$)

## $X_{ij}$ = number of units shipped from *supply point i* to *demand point j*

$$\min \sum_{i=1}^{i=m} \sum_{j=1}^{j=n} c_{ij} X_{ij}$$

$$s.t. \sum_{j=1}^{j=n} X_{ij} \leq s_i (i = 1,2,\ldots,m)$$

$$\sum_{i=1}^{i=m} X_{ij} \geq d_j ( j = 1,2,\ldots,n)$$

$$X_{ij} \geq 0 (i = 1,2,\ldots,m; j = 1,2,\ldots,n)$$

# Optimal LP Strategy for Example

- **In the above case, according to the optimal display schedule in the LP model, ad 1 is always selected for (afternoon, sports), ad 2 for (afternoon, not sports) and ad 3, otherwise.**
- **The total click-through rate of this optimal schedule is 2.1%**
  - and the actual click-through rates for ad 1, ad 2 and ad 3 are 2.2%, 2.1% and 2.0%, respectively.
- **Both greedy and random selection strategy have a CTR of *1.76%,***

# Partition using a predictive clustering

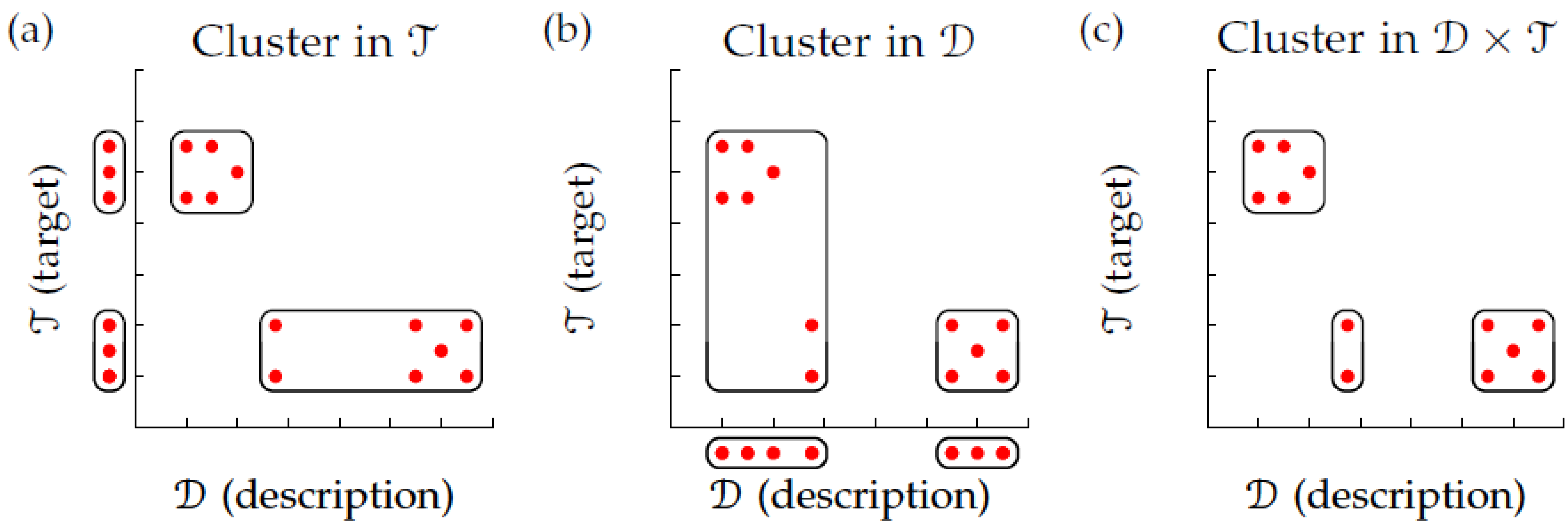



**Figure 2.1:** Illustration of predictive modeling (a), clustering (b), and predictive clustering (c). Figure taken from (Blockeel, 1998).

**Partition "webpages X users X Ads" into zones of self-similarity (using page, user, Ad and <u>CTR-based variables</u>) Vs (page, user, Ad )**

**[Chickering et al. 2001]**

# Maximize Revenue: Ad Allocation Example

| *From* | *To* | | | | |
|---|---|---|---|---|---|
| | *Ad1* | *Ad2* | *..Ad$_{j}$.....* | *Ad$_{m}$* | *Supply PageViews* |
| *PubZone 1* | $d_{ij}$ | $d_{ij}$ | $d_{ij}$ | $d_{ij}$ | 35 |
| *PubZone 2* | … | … | … | … | 50 |
| *PubZone3* | $d_{ij}$ | $d_{ij}$ | $d_{ij}$ | $d_{ij}$ | 15 |
| *Demand Contracted PageViews* | *45* | *20* | *30* | *5* | |

**Use LP to generate the Ad display schedule to maximize my revenue (or rev proxy, .i.e., CTR)**

# Results at msnbc.com

- **1.5 Million impress/Day, Dec 1998**

Table 1: Lifts for models.

| Cluster source | Lift |
|---|---|
| Predictive clusters | 38% |
| Standard clusters | 0% |
| Hand-assigned clusters | 18% |

[Chickering et al. 2001]

$$LL(\theta_{\mathcal{P}}) = \sum_{Z \in \mathcal{P}} \sum_{j=1}^{m} -\big(C_{Z,j} \log P(click|Z, j)$$
$$+ (D_{Z,j} - C_{Z,j}) \log\big(1 - P(click|Z, j)\big)\big),$$

where $D_{Z,j}$ is the number of displays (or impressions) for the cluster ad pair $Z, j$ and $C_{Z,j}$ is the number of clicks observed for the same pair. It is well known and straightforward to verify that this is minimized by letting $P(click|Z, j) = C_{Z,j}/D_{Z,j}$, so the *minimum* minus log likelihood for a given partition $\mathcal{P}$ is given as follows.

$$LL(\theta_{\mathcal{P}}) = \sum_{Z \in \mathcal{P}} \sum_{j=1}^{m} -\left(C_{Z,j} \log \frac{C_{Z,j}}{D_{Z,j}} + (D_{Z,j} - C_{Z,j}) \log \frac{D_{Z,j} - C_{Z,j}}{D_{Z,j}}\right).$$

The penalty term, according to AIC, is the number of free (probability) parameters in a model, and is simply

$$PT(\mathcal{P}) = m|\mathcal{P}|.$$

We let $I(\mathcal{P})$ denote $I(\{C_{Z,j}/D_{Z,j} \colon Z \in \mathcal{P},\ j = 1, \ldots, m\})$. Now, the minimization of $I(\theta_{\mathcal{P}})$ is reduced to that of $I(\mathcal{P})$.

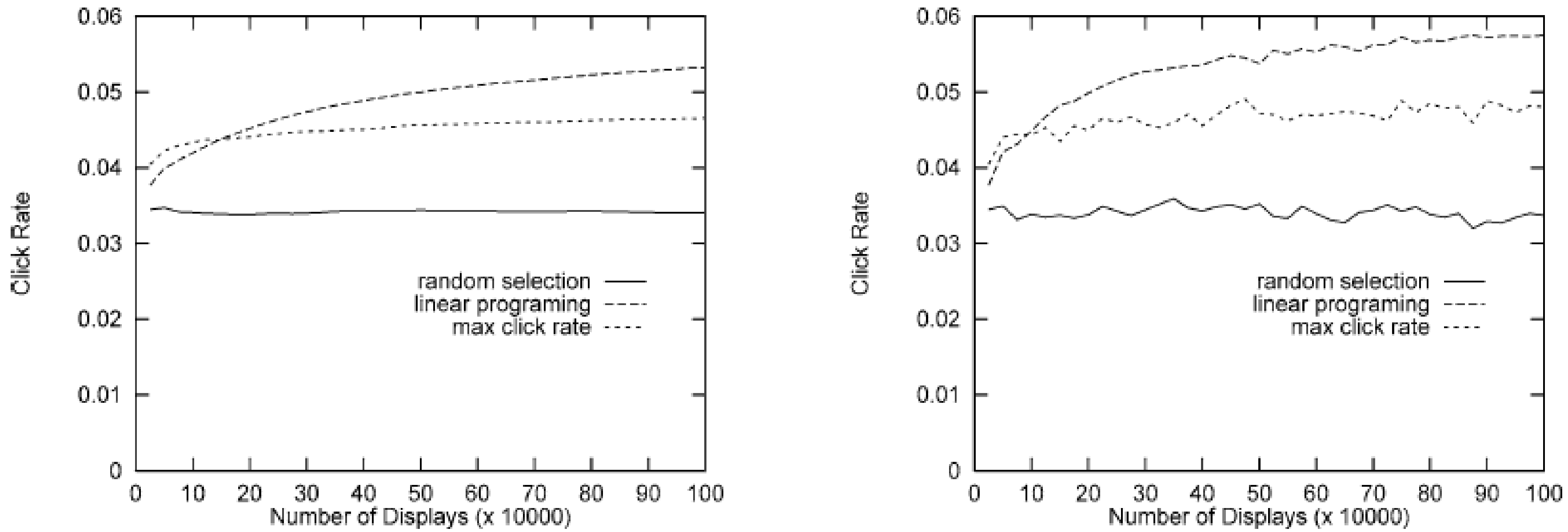
**Agglomerative Clustering using a likelihood-based objective**

**Greedy heuristic to search the best P; then get the click-through rate**

**[Nakamura and Abe]**

# Results for Nakamura, Abe

- **Simulation Results**
  - 32 Ads, 128 serving contexts (reduced to 32 clusters)




(a) Cumulative click rate

(b) Instantaneous click rate

**Modified Interior Point Alg**

# Adapting LP for "important" Advertisers

example, wherein 10,000 impressions each have been promised for two ads. Assume that the accurately estimated click-through rates (%) of these ads for combination 1 and 2 of attribute values are as shown in the following table.

| | ad 1 | ad 2 |
|---|---|---|
| Combination 1 of attribute values | 4.0 | 2.5 |
| Combination 2 of attribute values | 2.0 | 1.0 |

Then, the optimal solution[2] is the one that always displays ad 1 for combination 1 and ad 2 for combination 2. With this solution, ad 1 will have a high click-through rate of 4.0% while ad 2 suffers from having a low click-through rate of 1.0. This can be a problem, for example, if the advertiser of ad 2 is more important than the advertiser of ad 1. This problem can be dealt with, to some extent, by introducing the 'degree of importance' $g_j > 0$ for each ad $j$. The default value of $g_j$ is 1.0, and a greater value indicates a greater importance being assigned to ad $j$. We modify the objective function (1), in the linear programming formulation, by the following modification, which incorporates the degrees of importance:

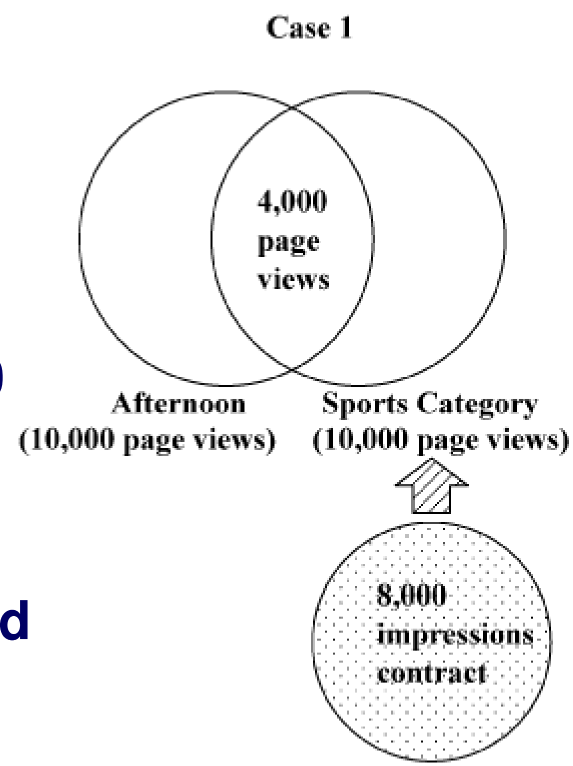
$$\sum_{i=1}^{n} \sum_{j=1}^{m} g_j c_{i,j} k_i d_{i,j}. \tag{5}$$

# Forward Markets

- **Linear Programming**
- **Quadratic Programming**
- **Allocation of Ads to Publisher real estate**
  - Give ads play in network
    - Optimize *revenue* subject to ….
- **Inventory Management**
  - Contract as many impressions as possible but don't oversell
- **Media Buyer (Arbitrage)**
  - Frame as a non-linear programming (NLP) problem
  - Talks to publisher
  - Determine publisher mix for network
    - Optimize *publisher mix* subject to constraints

# Problem 2: Ad allocation problem

- Ad agencies wish to contract as many ad impressions as possible to earn more revenue.
- But overselling is dangerous. So they need to grasp how many sellable impressions remain.
- In case 1, 8000 sellable impression remain for afternoon constraint, since at least 2000 views in (afternoon, sports) are needed for the contract of sports constraint.
- The calculation of the remaining sellable impressions for a certain constraint *t* should consider contracts for other constraints which overlap constraint *t*.
- t (how many impressions remain the target afternoon (as opposed to afternoon only))

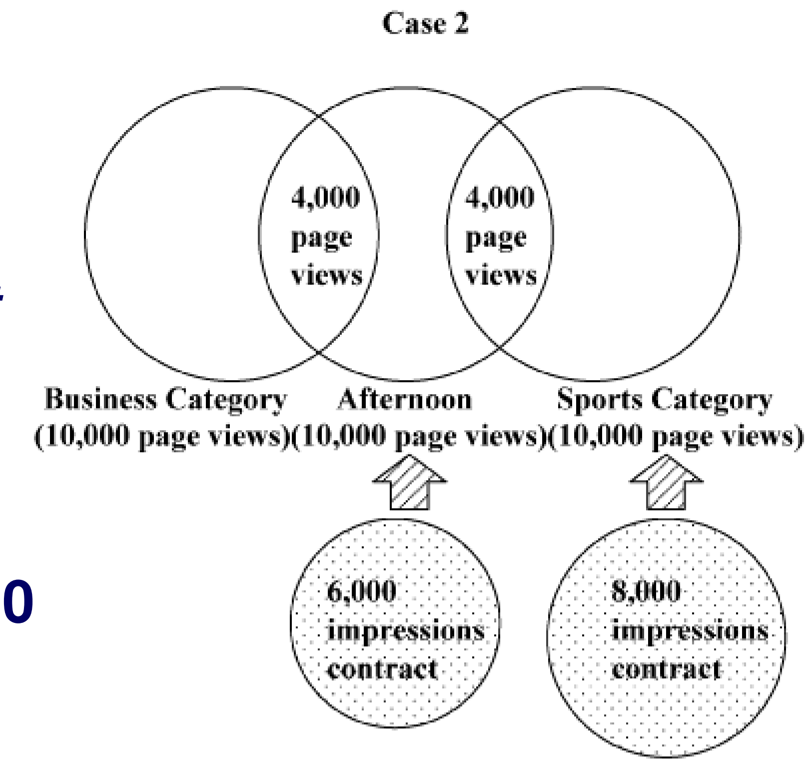



**How many page views can I sell for a publisher zone?**

# Ad allocation problem

***Should only overlapping constraints be considered?***

- **Case 2 says NO!**
- **The sellable impressions for business constraint is 8000, *not* 10000. The sports-constraint contract indirectly affects it, though they don't overlap.**
- **t is the business constraint (8000 possible pageviews)**



# LP: Intermediate Conclusions

- **Linear Programming and Machine learning work hand in hand to serve ads**
  - E.g., Advertising.com, Microsoft
- **Constraint optimization is critical in ad serving (especially in forward markets)**