More details:
General: http://www.learning-with-kernels.org/
Example of more complex bounds:
http://www.research.ibm.com/people/t/tzhang/papers/jmlr02_cover.ps.gz

PAC-learning, VC Dimension and Margin-based Bounds

Machine Learning – 10701/15781

Carlos Guestrin
Carnegie Mellon University

March 6th, 2006

Announcements 1

- Midterm on Wednesday
  - open book, texts, notes,…
  - no laptops
  - bring a calculator

Announcements 2

- Final project details are out!!!
  - http://www.cs.cmu.edu/~guestrin/Class/10701/projects.html
  - Great opportunity to apply ideas from class and learn more
  - Example project:
    - Take a dataset
    - Define learning task
    - Apply learning algorithms
    - Design your own extension
    - Evaluate your ideas
  - many of suggestions on the webpage, but you can also do your own
- Boring stuff:
  - Individually or groups of two students
  - It's worth 20% of your final grade
  - You need to submit a one page proposal on Wed. 3/22 (just after the break)
  - A 5-page initial write-up (milestone) is due on 4/12 (20% of project grade)
  - An 8-page final write-up due 5/8 (60% of the grade)
  - A poster session for all students will be held on Friday 5/5 2-5pm in NSH atrium (20% of the grade)
  - You can use late days on write-ups, each student in team will be charged a late day per day.
- MOST IMPORTANT:

What now…

- We have explored **many** ways of learning from data
- But…
  - How good is our classifier, really?
  - How much data do I need to make it "good enough"?

Learning Theory

How likely is learner to pick a bad hypothesis

- Prob. *h* with error$_{true}$(h) $\geq \varepsilon$ gets *m* data points right

- There are *k* hypothesis consistent with data
  - How likely is learner to pick a bad one?

P(at least one of the k was bad and it got lucky)?

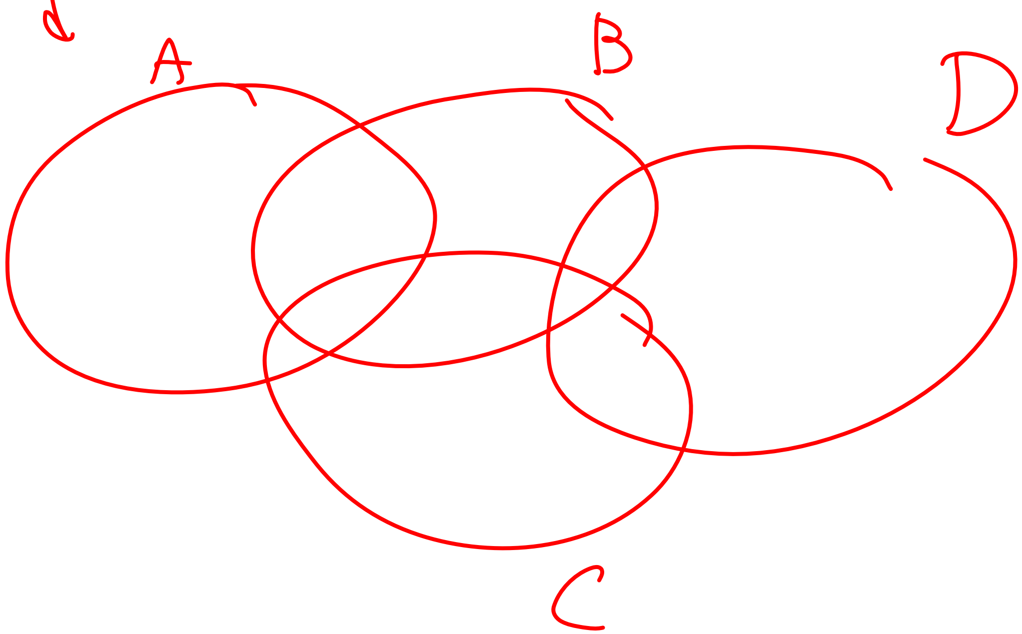
$P(\text{one got lucky}) \leq (1-\varepsilon)^m$

Union bound

bad hyp. got all m right

- P(A or B or C or D or …) $\leq P(A) + P(B) + P(C) + \dots$

area



How likely is learner to pick a bad hypothesis

- Prob. h with $\text{error}_{true}(h) \geq \varepsilon$ gets m data points right

- There are k hypothesis consistent with data
  - How likely is learner to pick a bad one?

$P(h_1 \text{ bad } \& \text{ got lucky or } h_2 \text{ bad } \& \text{ got lucky or } h_3 \ldots)$

$\leq P(h_1 \text{ bad } \& \text{ lucky}) + P(h_2 \text{ bad } \& \text{ lucky}) + P(h_3 \ldots) + \ldots$

$\leq (1-\varepsilon)^m$

how big is K

$K \leq |H|$ (loose bound!)

$1-\varepsilon \leq e^{-\varepsilon}$ (make eq. simpler)

$\leq K(1-\varepsilon)^m$

$\leq |H| e^{-\varepsilon m}$

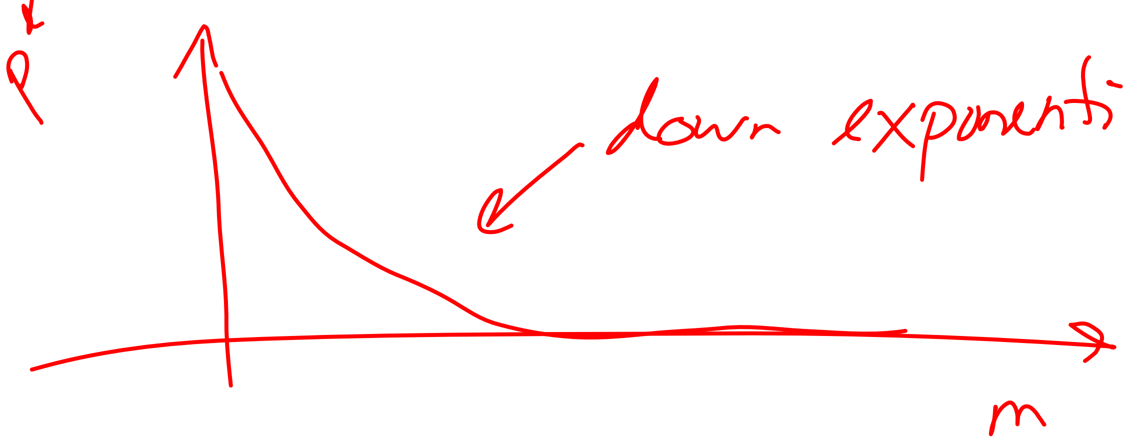
Review: Generalization error in finite hypothesis spaces [Haussler '88]

- ***Theorem***: Hypothesis space H finite, dataset D with m i.i.d. samples, $0 < \varepsilon < 1$: for any learned hypothesis h that is consistent on the training data:

$$P(\text{error}_{\cancel{\mathcal{X}}\ \text{true}}(h) > \epsilon) \leq |H| e^{-m\epsilon} \leq \delta$$

true

0.01



down exponentially fast!

Using a PAC bound

- Typically, 2 use cases:
  - 1: Pick ε and δ, give you *m*
  - 2: Pick m and δ, give you ε

$$P(\text{error}_{\cancel{x}\ \text{true}}(h) > \epsilon) \leq |H|e^{-m\epsilon}$$

① $P \leq |H| e^{-m\varepsilon} \leq \delta$

smaller $\delta \Rightarrow$ need more data

$\ln|H| - m\varepsilon \leq \ln \delta$ less

$\Rightarrow \quad m \geq \frac{1}{\varepsilon}\left(\ln|H| + \ln\frac{1}{\delta}\right)$

smaller ε more data

not as much data as you may think

Case 2 $\delta \leq |H| e^{-m\varepsilon}$

$\ln \delta \leq \ln|H| - m\varepsilon$

$\varepsilon \leq \frac{1}{m}\left(\ln|H| + \ln\frac{1}{\delta}\right)$

learn h

true error ≤

before you run the algorithm

Review: Generalization error in finite hypothesis spaces [Haussler '88]

- ***Theorem***: Hypothesis space *H* finite, dataset *D* with *m* i.i.d. samples, $0 < \varepsilon < 1$: for any learned hypothesis *h* that is consistent on the training data:

$$P(\mathrm{error}_{\mathcal{X}}(h) > \epsilon) \leq |H|e^{-m\epsilon}$$

if I can always learn a consistent classifier

then

Even if *h* makes zero errors in training data, may make errors in test

Limitations of Haussler '88 bound

$$P(\text{error}_{\cancel{x}\,\text{true}}(h) > \epsilon) \leq |H|e^{-m\epsilon}$$

① ■ Consistent classifier

there may not be such h in class!

② ■ Size of hypothesis space

bound depends on |H| → really really large?

→ infinite?

w continuous

Simpler question: What's the expected error of a hypothesis?

- The error of a hypothesis is like estimating the parameter of a coin!

true error ← θ
I don't know it

flip coin m times
(train error)
$\hat{\theta}$

θ v. $\hat{\theta}$

- Chernoff bound: for *m* i.i.d. coin flips, $x_1,\ldots,x_m$, where $x_i \in \{0,1\}$. For $0<\varepsilon<1$:

$$\boxed{P\left(\theta - \frac{1}{m}\sum_i x_i > \epsilon\right) \le e^{-2m\epsilon^2}}$$

truth

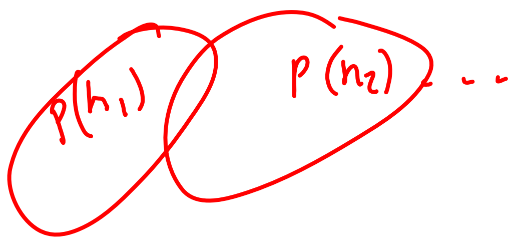
train error
sample average

But we are comparing many hypothesis: **Union bound**

For each hypothesis h_i:

$$P\left(\text{error}_{true}(h_i) - \text{error}_{train}(h_i) > \epsilon\right) \leq e^{-2m\epsilon^2}$$

What if I am comparing two hypothesis, h_1 and h_2?



learner is going to compare all of h_i's

$$P\left(\exists_i \text{ error}_{true}(h_i) - \text{error}_{train}(h_i) > \varepsilon\right) \leq |H|\, e^{-2m\varepsilon^2}\ \delta$$

Generalization bound for |H| hypothesis

- ***Theorem***: Hypothesis space *H* finite, dataset *D* with *m* i.i.d. samples, $0 < \varepsilon < 1$: for any learned hypothesis *h*:

$$P\left(\text{error}_{true}(h) - \text{error}_{train}(h) > \epsilon\right) \leq |H|e^{-2m\epsilon^2}$$

$2m\varepsilon^2 = 20$ not as good !!

side note: Haussler's bound for consistent h:

$$P \leq |H| \, e^{-m\varepsilon}$$

$\varepsilon = 0.1$

$m = 1000$

$\Rightarrow m \cdot \varepsilon = 100$

PAC bound and Bias-Variance tradeoff

$$P\left(\text{error}_{true}(h) - \text{error}_{train}(h) > \epsilon\right) \leq |H|e^{-2m\epsilon^2}$$

or, after moving some terms around, with probability at least 1-δ:

$$\text{error}_{true}(h) \leq \text{error}_{train}(h) + \sqrt{\frac{\ln|H| + \ln\frac{1}{\delta}}{2m}}$$

want to minimize

"bias"	"variance"	
↑ larger	↓ smaller	$\|H\|$ small
↓ smaller	↑ larger	$\|H\|$ large

- **Important: PAC bound holds for all *h*, but doesn't guarantee that algorithm finds best *h*!!!**

What about the size of the hypothesis space?

$$m \geq \frac{1}{2\epsilon^2}\left(\ln|H| + \ln\frac{1}{\delta}\right)$$

need this amount

- How large is the hypothesis space? $|H|$

$\ln|H|$?

Boolean formulas with *n* binary features

$$m \geq \frac{1}{2\epsilon^2}\left(\ln|H| + \ln\frac{1}{\delta}\right)$$

what's $\ln|H|$?

tabular representation

X_1	X_2	X_3	X_4	Y
T	T	T	T	0/1
T	T	T	F	0/1
T	T	F	T	0/1
T	T	F	F	⋮

2^n rows

each 2 possibil.

$|H| = 2^{2^n}$ (really really large)

$\ln|H| = 2^n \ln 2$

too large

conjunctions:

$h_1 = X_1 \wedge X_2 \wedge X_7$

$h_2 = X_2 \wedge \neg X_5 \wedge X_8 \ldots$

n attrib..

$\langle \{\emptyset, 1, \neg\}, \{\emptyset, 1, \neg\}, \ldots \rangle$

$|H| = 3^n$ (really large)

$\ln|H| = n \ln 3$

"small"

Number of decision trees of depth k

$$m \geq \frac{1}{2\epsilon^2}\left(\ln|H| + \ln\frac{1}{\delta}\right)$$

Recursive solution

Given *n* attributes

H_k = Number of decision trees of depth k

$H_0 = 2$

H_{k+1} = (#choices of root attribute) *

(# possible left subtrees) *

(# possible right subtrees)

$= n * H_k * H_k$

Write $L_k = \log_2 H_k$

$L_0 = 1$

$L_{k+1} = \log_2 n + 2L_k$

So $L_k = (2^k-1)(1+\log_2 n) + 1$

PAC bound for decision trees of depth k

$$m \geq \frac{\ln 2}{2\epsilon^2}\left((2^k - 1)(1 + \log_2 n) + 1 + \ln \frac{1}{\delta}\right)$$

- Bad!!!
  - Number of points is exponential in depth!

- But, for *m* data points, decision tree can't get too big…

Number of leaves never more than number data points

Number of decision trees with k leaves

$$m \geq \frac{1}{2\epsilon^2}\left(\ln|H| + \ln\frac{1}{\delta}\right)$$

H_k = Number of decision trees with k leaves

H_0 =2

$$H_{k+1} = n \sum_{i=1}^{k} H_i H_{k+1-i}$$

Loose bound:

$$H_k \leq n^{k-1}(k+1)^{2k-1}$$

Reminder:

$$|\text{DTs depth } k| = 2 * (2n)^{2^k-1}$$

PAC bound for decision trees with k leaves – Bias-Variance revisited

$$H_k = n^{k-1}(k+1)^{2k-1}$$

$$\text{error}_{true}(h) \leq \text{error}_{train}(h) + \sqrt{\frac{\ln|H| + \ln\frac{1}{\delta}}{2m}}$$

$$\boxed{\text{error}_{true}(h) \leq \text{error}_{train}(h) + \sqrt{\frac{(k-1)\ln n + (2k-1)\ln(k+1) + \ln\frac{1}{\delta}}{2m}}}$$

What did we learn from decision trees?

- Bias-Variance tradeoff formalized

$$\text{error}_{true}(h) \leq \text{error}_{train}(h) + \sqrt{\frac{(k-1)\ln n + (2k-1)\ln(k+1) + \ln\frac{1}{\delta}}{2m}}$$

- Moral of the story:

  Complexity of learning not measured in terms of size hypothesis space, but in maximum *number of points* that allows consistent classification

  - Complexity *m* – no bias, lots of variance
  - Lower than *m* – some bias, less variance

What about continuous hypothesis spaces?

$$\text{error}_{true}(h) \leq \text{error}_{train}(h) + \sqrt{\frac{\ln|H| + \ln\frac{1}{\delta}}{2m}}$$

- Continuous hypothesis space:
  - $|H| = \infty$
  - Infinite variance???

- **As with decision trees, only care about the maximum number of points that can be classified exactly!**

How many points can a linear boundary classify exactly? (1-D)

How many points can a linear boundary classify exactly? (2-D)

How many points can a linear boundary classify exactly? (d-D)

Shattering a set of points

Definition: a **dichotomy** of a set S is a partition of S into two disjoint subsets.

Definition: a set of instances S is **shattered** by hypothesis space H if and only if for every dichotomy of S there exists some hypothesis in H consistent with this dichotomy.

VC dimension

Definition: The **Vapnik-Chervonenkis dimension**, $VC(H)$, of hypothesis space H defined over instance space X is the size of the largest finite subset of X shattered by H. If arbitrarily large finite sets of X can be shattered by H, then $VC(H) \equiv \infty$.

PAC bound using VC dimension

- **Number of training points that can be classified exactly is VC dimension!!!**
  - **Measures relevant size of hypothesis space, as with decision trees with k leaves**
  - **Bound for infinite dimension hypothesis spaces:**

$$\text{error}_{true}(h) \leq \text{error}_{train}(h) + \sqrt{\frac{VC(H)\left(\ln \frac{2m}{VC(H)} + 1\right) + \ln \frac{4}{\delta}}{m}}$$

Examples of VC dimension

$$\text{error}_{true}(h) \leq \text{error}_{train}(h) + \sqrt{\frac{VC(H)\left(\ln\frac{2m}{VC(H)}+1\right)+\ln\frac{4}{\delta}}{m}}$$

- Linear classifiers:
  - VC(H) = d+1, for *d* features plus constant term *b*
- Neural networks
  - VC(H) = #parameters
  - Local minima means NNs will probably not find best parameters
- 1-Nearest neighbor?

Another VC dim. example

- What's the VC dim. of decision stumps in 2d?

PAC bound for SVMs

- SVMs use a linear classifier
  - For *d* features, VC(H) = d+1:

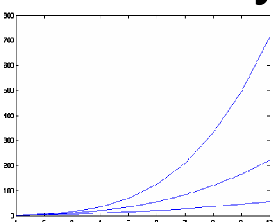
$$\text{error}_{true}(h) \leq \text{error}_{train}(h) + \sqrt{\frac{(d+1)\left(\ln \frac{2m}{d+1} + 1\right) + \ln \frac{4}{\delta}}{m}}$$

VC dimension and SVMs: Problems!!!

Doesn't take margin into account

$$\text{error}_{true}(h) \leq \text{error}_{train}(h) + \sqrt{\frac{(d+1)\left(\ln\frac{2m}{d+1}+1\right)+\ln\frac{4}{\delta}}{m}}$$

- What about kernels?
  - Polynomials: num. features grows really fast = Bad bound



$$\text{num. terms} = \binom{p+n-1}{p} = \frac{(p+n-1)!}{p!(n-1)!}$$

n – input features
p – degree of polynomial

  - Gaussian kernels can classify any set of points exactly

Margin-based VC dimension

- H: Class of linear classifiers: $\mathbf{w}.\Phi(\mathbf{x})$ (b=0)
  - Canonical form: $\min_j |\mathbf{w}.\Phi(\mathbf{x}_j)| = 1$
- $VC(H) = R^2\, \mathbf{w.w}$
  - Doesn't depend on number of features!!!
  - $R^2 = \max_j \Phi(\mathbf{x}_j).\Phi(\mathbf{x}_j)$ – magnitude of data
  - R^2 is bounded even for Gaussian kernels $\rightarrow$ bounded VC dimension
- Large margin, low **w.w**, low VC dimension – Very cool!

Applying margin VC to SVMs?

$$\text{error}_{true}(h) \leq \text{error}_{train}(h) + \sqrt{\frac{VC(H)\left(\ln\frac{2m}{VC(H)}+1\right)+\ln\frac{4}{\delta}}{m}}$$

- VC(H) = R^2 **w.w**
  - $R^2 = \max_j \Phi(\mathbf{x}_j).\Phi(\mathbf{x}_j)$ – magnitude of data, doesn't depend on choice of **w**
- SVMs minimize **w.w**

- **SVMs minimize VC dimension to get best bound?**
- **Not quite right: ☹**
  - **Bound assumes VC dimension chosen before looking at data**
  - **Would require union bound over infinite number of possible VC dimensions…**
  - **But, it can be fixed!**

Structural risk minimization theorem

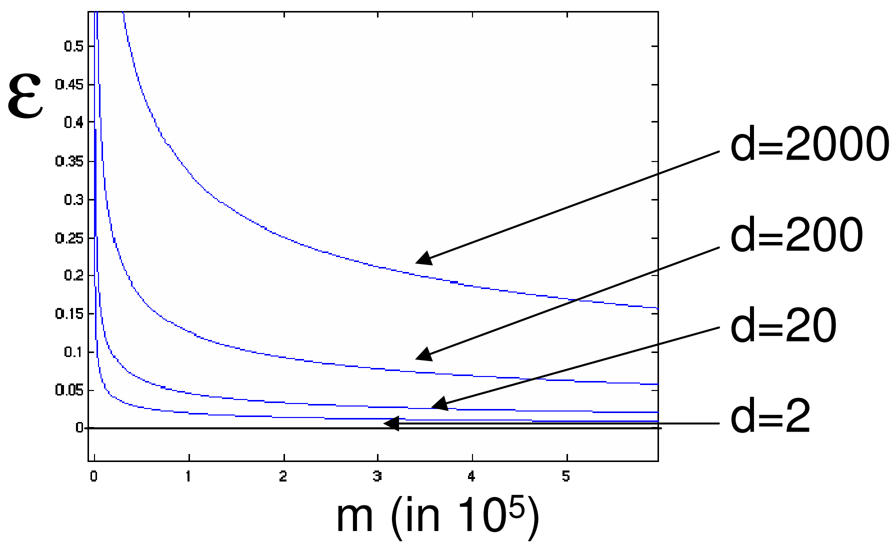
$$\text{error}_{true}(h) \leq \text{error}^{\gamma}_{train}(h) + C\sqrt{\frac{\frac{R^2}{\gamma^2}\ln m + \ln\frac{1}{\delta}}{m}}$$

$$\text{error}^{\gamma}_{train}(h) = \text{num. points with margin} < \gamma$$

- For a family of hyperplanes with margin $\gamma>0$
  - **w.w** ≤ 1
- SVMs maximize margin γ + hinge loss
  - Optimize tradeoff training error (bias) versus margin γ (variance)

Reality check – Bounds are loose

$$\text{error}_{true}(h) \leq \text{error}_{train}(h) + \underbrace{\sqrt{\frac{(d+1)\left(\ln\frac{2m}{d+1}+1\right)+\ln\frac{4}{\delta}}{m}}}_{\epsilon}$$



- Bound can be very loose, why should you care?
  - There are tighter, albeit more complicated, bounds
  - Bounds gives us formal guarantees that empirical studies can't provide
  - Bounds give us intuition about complexity of problems and convergence rate of algorithms

What you need to know

- Finite hypothesis space
  - Derive results
  - Counting number of hypothesis
  - Mistakes on Training data
- Complexity of the classifier depends on number of points that can be classified exactly
  - Finite case – decision trees
  - Infinite case – VC dimension
- Bias-Variance tradeoff in learning theory
- Margin-based bound for SVM
- Remember: will your algorithm find best classifier?

Big Picture

Machine Learning – 10701/15781

Carlos Guestrin
Carnegie Mellon University

March 6th, 2006

What you have learned thus far

- Learning is function approximation
- Point estimation
- Regression
- Naïve Bayes
- Logistic regression
- Bias-Variance tradeoff
- Neural nets
- Decision trees
- Cross validation
- Boosting
- Instance-based learning
- SVMs
- Kernel trick
- PAC learning
- VC dimension
- Margin bounds
- Mistake bounds

Review material in terms of…

- Types of learning problems
- Hypothesis spaces
- Loss functions
- Optimization algorithms

Text Classification

→ Company home page

vs

Personal home page

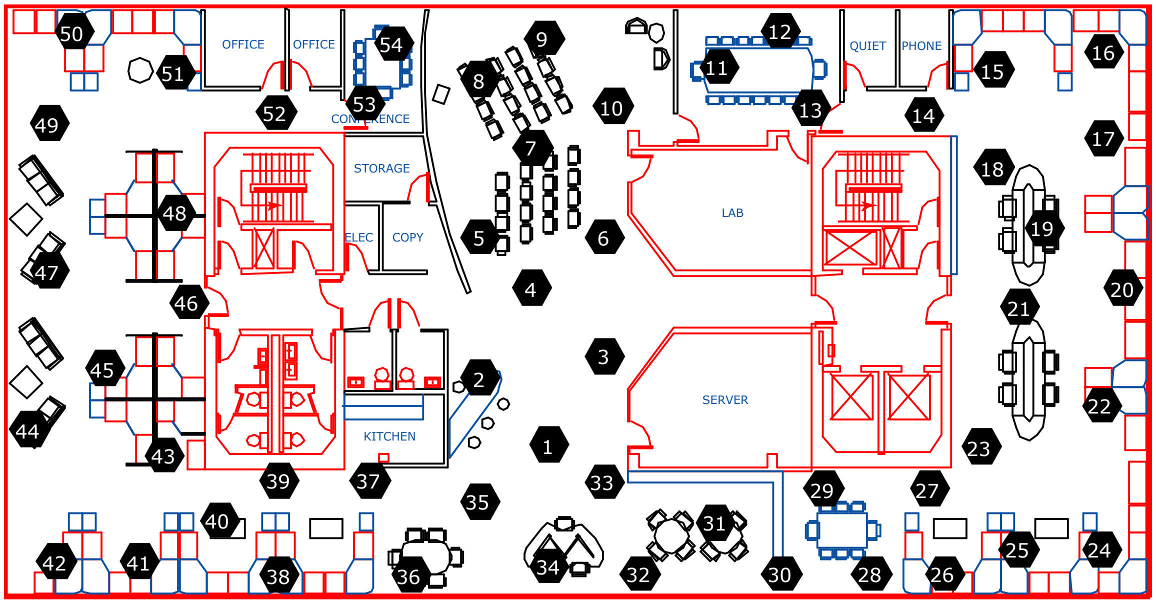
vs

Univeristy home page

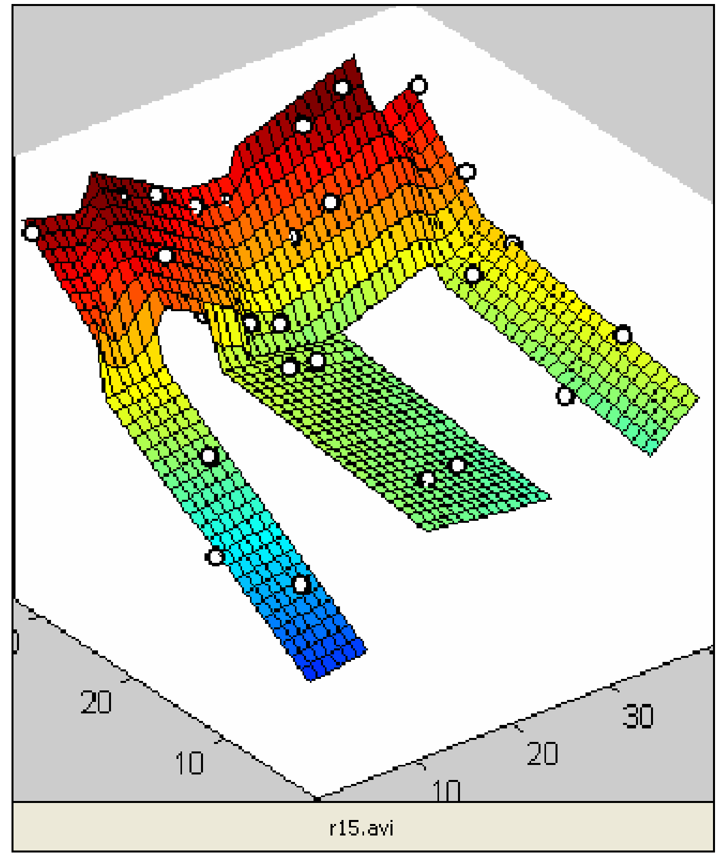
vs

…

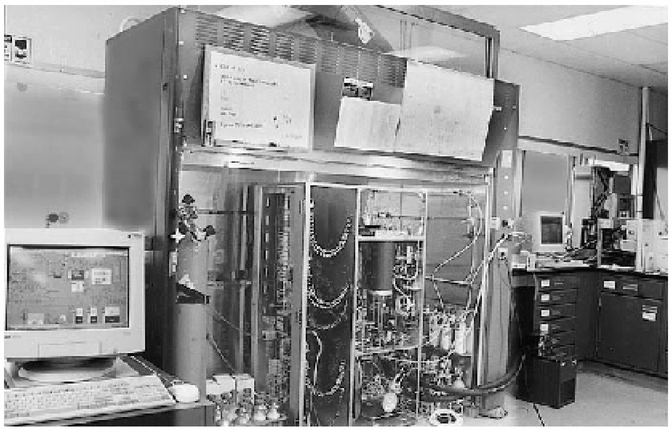
Function fitting

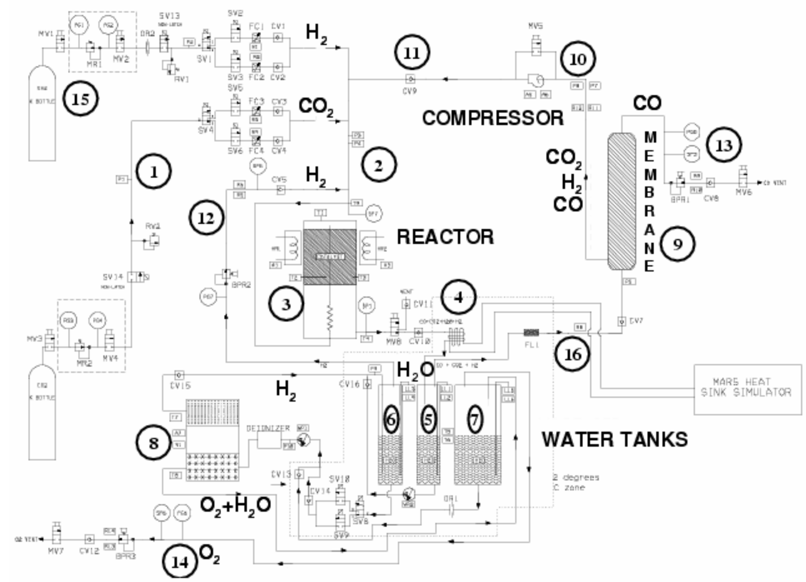


Temperature data



Monitoring a complex system

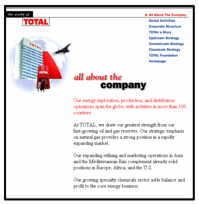




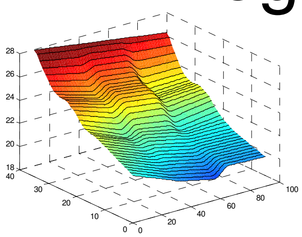
- Reverse water gas shift system (RWGS)
- Learn model of system from data
- Use model to predict behavior and detect faults

Types of learning problems

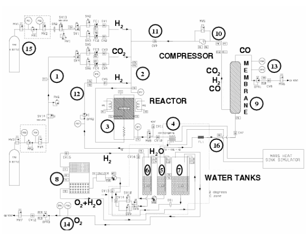
- Classification



- Regression



- Density estimation



Input – Features

Output?

The learning problem

Data

$\langle x_1,\ldots,x_n,y\rangle$

Learning task

Features/Function approximator

Loss function

Optimization algorithm

Learned function

Comparing learning algorithms

- Hypothesis space

- Loss function

- Optimization algorithm

Naïve Bayes versus Logistic regression

Naïve Bayes

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

$$P(X|Y) = \prod_i P(X_i|Y)$$

Logistic regression

$$P(Y=1|x) = \frac{1}{1 + exp(w_0 + \sum_i w_i x_i)}$$

Naïve Bayes versus Logistic regression – Classification as density estimation

$$P(Y|X)$$

- Choose class with highest probability

- In addition to class, we get certainty measure

Logistic regression versus Boosting

Logistic regression

$$P(Y = y_i|\mathbf{x}) = \frac{1}{1 + exp(-y_i(\mathbf{w}.\mathbf{x} + b))}$$

Log-loss

$$\sum_{j=1}^{m} \log \left[1 + exp(-y_i(\mathbf{w}.\mathbf{x}_j + b))\right]$$

Boosting

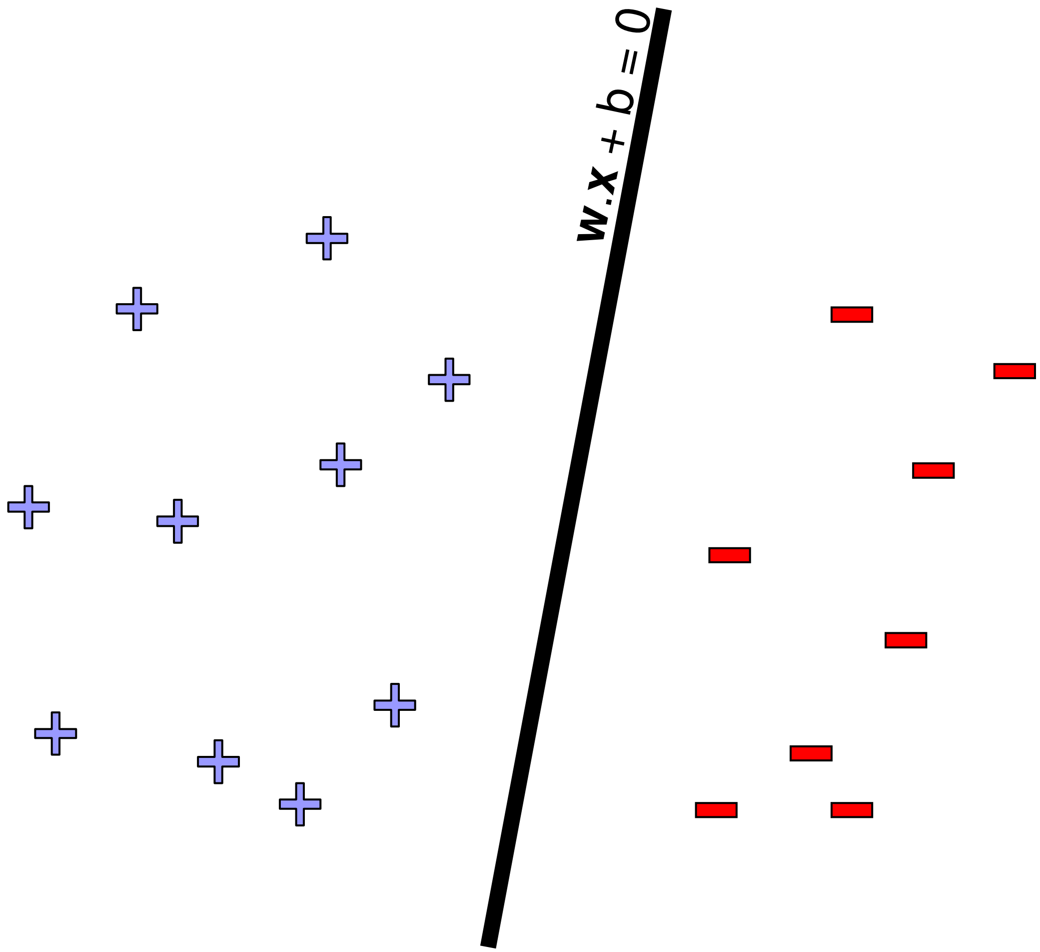
Classifier

$$sign\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$$

Exponential-loss

$$\frac{1}{m}\sum_{j=1}^{m} \exp\left(-y_j \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x_j})\right)$$

Linear classifiers – Logistic regression versus SVMs



What's the difference between SVMs and Logistic Regression? (Revisited again)

	SVMs	**Logistic Regression**
Loss function	Hinge loss	Log-loss
High dimensional features with kernels	Yes!	Yes!
Solution sparse	Often yes!	Almost always no!
Type of learning		

SVMs and instance-based learning

$$\mathbf{w} \cdot \Phi(\mathbf{x}) = \sum_i \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i)$$

$$b = y_k - \sum_i \alpha_i y_i K(\mathbf{x}_k, \mathbf{x}_i)$$

for any k where $C > \alpha_k > 0$

Classify as

SVMs

$$sign\left(\sum_i \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b\right)$$

Instance based learning

$$P(y \mid \mathbf{x}) = \frac{\sum_i y_i K(\mathbf{x}, \mathbf{x}_i)}{\sum_i K(\mathbf{x}, \mathbf{x}_i)} > 0.5?$$

Data

$<x_1,\ldots,x_n,y>$

Classify as

$$sign\left(\sum_i y_i K(\mathbf{x}, \mathbf{x}_i) - 0.5 \sum_i K(\mathbf{x}, \mathbf{x}_i)\right)$$

Instance-based learning versus Decision trees

1-Nearest neighbor	**Decision trees**

Logistic regression versus Neural nets

$$g(w_0 + \sum_i w_i x_i) = \frac{1}{1 + e^{-(w_0 + \sum_i w_i x_i)}}$$

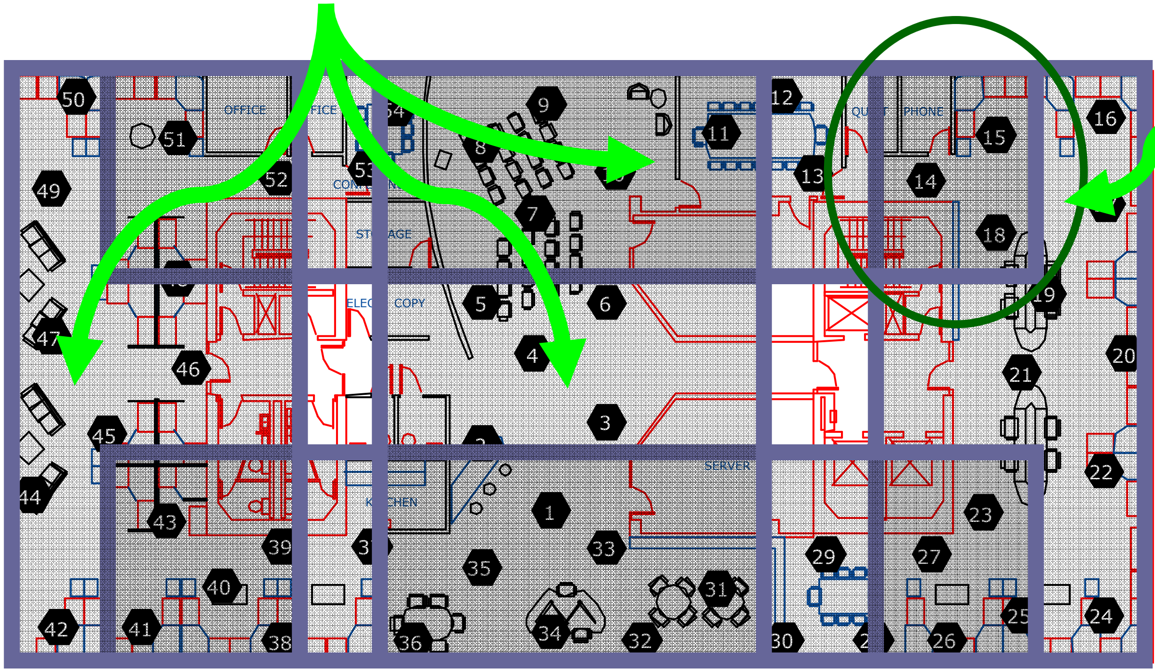
Logistic regression **Neural Nets**

Linear regression versus Kernel regression

Linear Regression	Kernel regression	Kernel-weighted linear regression

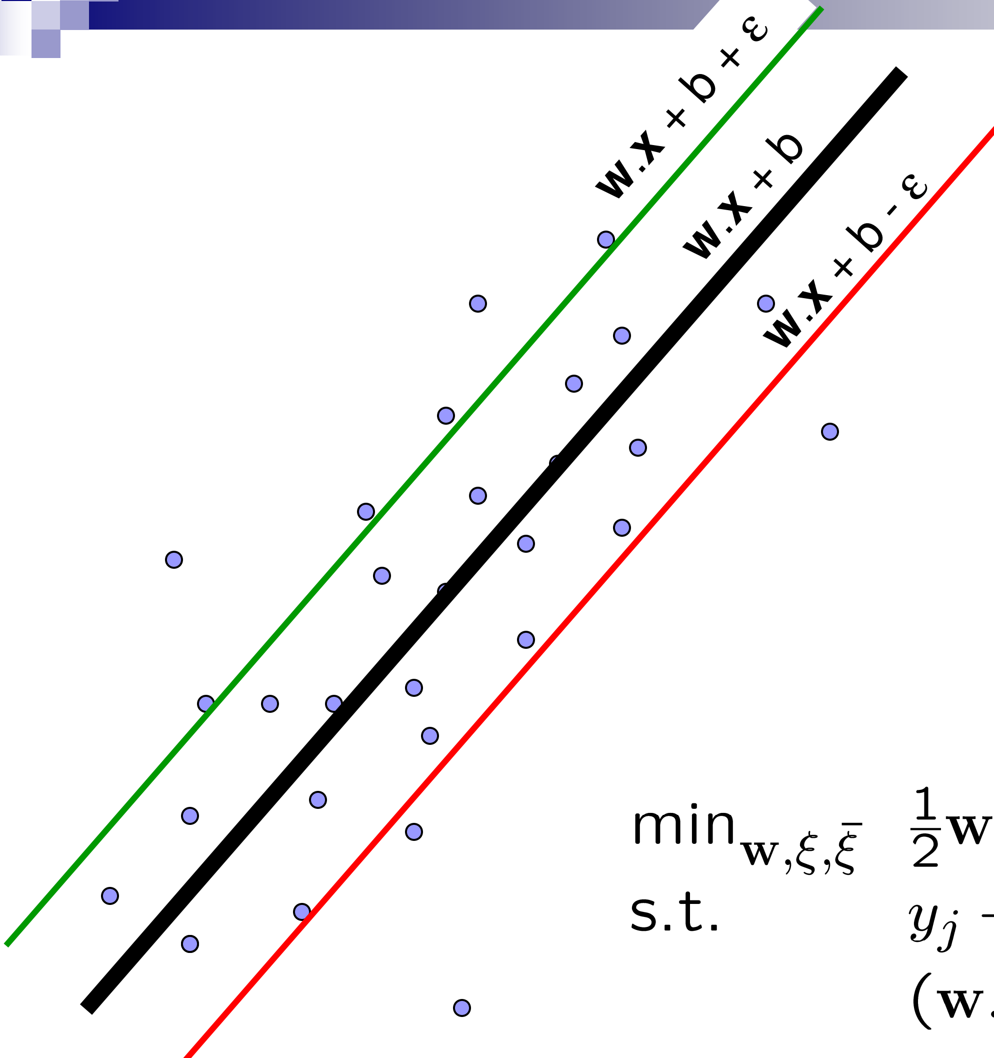
Kernel-weighted linear regression

- Local basis functions for each region



Kernels average between regions

SVM regression



$$
\begin{aligned}
\min_{\mathbf{w},\xi,\bar{\xi}} \quad & \tfrac{1}{2}\mathbf{w}.\mathbf{w} + C\textstyle\sum_{j=1}^{m}(\xi_j + \bar{\xi}_j) \\
\text{s.t.} \quad & y_j - (\mathbf{w}.\mathbf{x}_j + b) \leq \epsilon + \xi_j \\
& (\mathbf{w}.\mathbf{x}_j + b) - y_j \leq \epsilon + \bar{\xi}_j \\
& \xi_j \geq 0, \quad \bar{\xi}_j \geq 0, \quad \forall j
\end{aligned}
$$

BIG PICTURE

(a few points of comparison)

learning task	DE	density estimation
	Cl	Classification
	Reg	Regression
loss function	LL	Log-loss/MLE
	Mrg	Margin-based
	RMS	Squared error

Naïve Bayes
DE, LL

Boosting
Cl, exp-loss

SVM regression
Reg, Mrg

Logistic regression
DE, LL

SVMs
Cl, Mrg

kernel regression
Reg, RMS

Instance-based Learning
DE,Cl,Reg

Neural Nets
DE,Cl,Reg,RMS

Decision trees
DE,Cl,Reg

linear regression
Reg, RMS

This is a very incomplete view!!!