

CS 221: Artificial Intelligence

Lecture 5: Machine Learning

Peter Norvig and Sebastian Thrun

Outline

- Machine Learning
- Classification (Naïve Bayes)
- Regression (Linear, Smoothing)
- Linear Separation (Perceptron, SVMs)
- Non-parametric classification (KNN)

Outline

- Machine Learning
- Classification (Naïve Bayes)
- Regression (Linear, Smoothing)
- Linear Separation (Perceptron, SVMs)
- Non-parametric classification (KNN)

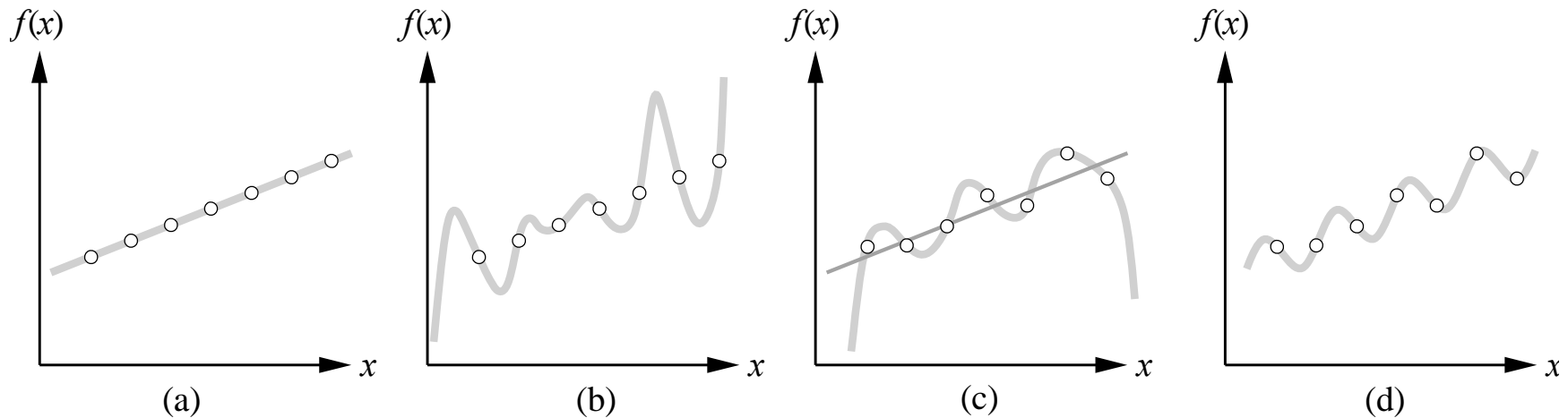
Machine Learning

- Up until now: how to reason in a give model
- Machine learning: how to acquire a model on the basis of data / experience
 - Learning parameters (e.g. probabilities)
 - Learning structure (e.g. BN graphs)
 - Learning hidden concepts (e.g. clustering)

Machine Learning Lingo

What?	Parameters	Structure	Hidden concepts	
What from?	Supervised	Unsupervised	Reinforcement	Self-supervised
What for?	Prediction	Diagnosis	Compression	Discovery
How?	Passive	Active	Online	Offline
Output?	Classification	Regression	Clustering	
Details??	Generative	Discriminative	Smoothing	

Supervised Machine Learning



Given a training set:

$$(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots, (x_n, y_n)$$

Where each y_i was generated by an unknown $y = f(x)$,
Discover a function h that approximates the true function f .

Outline

- Machine Learning
- Classification (Naïve Bayes)
- Regression (Linear, Smoothing)
- Linear Separation (Perceptron, SVMs)
- Non-parametric classification (KNN)

Classification Example: Spam Filter

- Input: x = email
- Output: y = “spam” or “ham”
- Setup:
 - Get a large collection of example emails, each labeled “spam” or “ham”
 - Note: someone has to hand label all this data!
 - Want to learn to predict labels of new, future emails
- Features: The attributes used to make the ham / spam decision
 - Words: FREE!
 - Text Patterns: \$dd, CAPS
 - Non-text: SenderInContacts
 -



Dear Sir.

First, I must solicit your confidence in this transaction, this is by virtue of its nature as being utterly confidential and top ugetgw

TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.

99 MILLION EMAIL ADDRESSES
FOR ONLY \$99

Ok, I know this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.

A Spam Filter

- Naïve Bayes spam filter

- Data:

- Collection of emails, labeled spam or ham
- Note: someone has to hand label all this data!
- Split into training, held-out, test sets



Dear Sir.

First, I must solicit your confidence in this transaction, this is by virtue of its nature as being utterly confidential and top ugetgw



TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.

99 MILLION EMAIL ADDRESSES
FOR ONLY \$99

- Classifiers

- Learn on the training set
- (Tune it on a held-out set)
- Test it on new emails



Ok, I know this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.

Naïve Bayes for Text


- Bag-of-Words Naïve Bayes:

- Predict unknown class label (spam vs. ham)
- Assume evidence features (e.g. the words) are independent

- Generative model

$$P(C, W_1 \dots W_n) = P(C) \prod_i P(W_i|C)$$

*Word at position
i, not ith word in
the dictionary!*



- Tied distributions and bag-of-words

- Usually, each variable gets its own conditional probability distribution $P(F|Y)$
- In a bag-of-words model
 - Each position is identically distributed
 - All positions share the same conditional probs $P(W|C)$
 - Why make this assumption?

General Naïve Bayes

- General probabilistic model:

$$P(Y, F_1 \dots F_n)$$

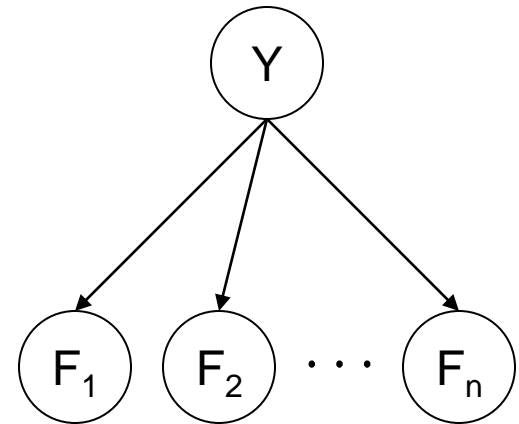
$|Y| \times |F|^n$ parameters

- General *naïve Bayes* model:

$$P(Y, F_1 \dots F_n) = P(Y) \prod_i P(F_i | Y)$$

$|Y|$ parameters

$n \times |F| \times |Y|$
parameters



Example: Spam Filtering

- **Model:** $P(C, W_1 \dots W_n) = P(C) \prod_i P(W_i|C)$
- What are the parameters?

$P(C)$

ham : 0.66
spam: 0.33

$P(W|\text{spam})$

the : 0.0156
to : 0.0153
and : 0.0115
of : 0.0095
you : 0.0093
a : 0.0086
with: 0.0080
from: 0.0075
...

$P(W|\text{ham})$

the : 0.0210
to : 0.0133
of : 0.0119
2002: 0.0110
with: 0.0108
from: 0.0107
and : 0.0105
a : 0.0100
...

- Where do these tables come from? Counts from examples!

Spam Example

$$P(\text{spam} \mid \text{words}) = e^{-76} / (e^{-76} + e^{-80.5}) = 98.9\%$$

Example: Overfitting

- Posterior determined by *relative* probabilities (odds ratios):

$$\frac{P(W|\text{ham})}{P(W|\text{spam})}$$

south-west	:	inf
nation	:	inf
morally	:	inf
nicely	:	inf
extent	:	inf
seriously	:	inf
...		

$$\frac{P(W|\text{spam})}{P(W|\text{ham})}$$

screens	:	inf
minute	:	inf
guaranteed	:	inf
\$205.00	:	inf
delivery	:	inf
signature	:	inf
...		

What went wrong here?

Generalization and Overfitting

- Raw counts will **overfit** the training data!
 - Unlikely that every occurrence of “minute” is 100% spam
 - Unlikely that every occurrence of “seriously” is 100% ham
 - What about all the words that don’t occur in the training set at all? 0/0?
 - In general, we can’t go around giving unseen events zero probability
- At the extreme, imagine using the entire email as the only feature
 - Would get the training data perfect (if deterministic labeling)
 - Wouldn’t *generalize* at all
 - Just making the bag-of-words assumption gives us some generalization, but isn’t enough
- To generalize better: we need to **smooth** or **regularize** the estimates

Estimation: Smoothing

- Maximum likelihood estimates:

$$P_{\text{ML}}(x) = \frac{\text{count}(x)}{\text{total samples}}$$



$$P_{\text{ML}}(\textcolor{red}{r}) = 1/3$$

- Problems with maximum likelihood estimates:

- If I flip a coin once, and it's heads, what's the estimate for $P(\text{heads})$?
- What if I flip 10 times with 8 heads?
- What if I flip 10M times with 8M heads?

- Basic idea:

- We have some prior expectation about parameters (here, the probability of heads)
- Given little evidence, we should skew towards our prior
- Given a lot of evidence, we should listen to the data

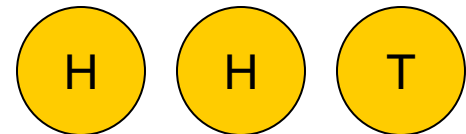
Estimation: Laplace Smoothing

- Laplace's estimate (extended):

- Pretend you saw every outcome k extra times

$$P_{LAP,k}(x) = \frac{c(x) + k}{N + k|X|}$$

- What's Laplace with $k = 0$?
- k is the **strength** of the prior



$$P_{LAP,0}(X) =$$

$$P_{LAP,1}(X) =$$

$$P_{LAP,100}(X) =$$

- Laplace for conditionals:

- Smooth each condition independently:

$$P_{LAP,k}(x|y) = \frac{c(x, y) + k}{c(y) + k|X|}$$

Estimation: Linear Interpolation

- In practice, Laplace often performs poorly for $P(X|Y)$:
 - When $|X|$ is very large
 - When $|Y|$ is very large
- Another option: linear interpolation
 - Also get $P(X)$ from the data
 - Make sure the estimate of $P(X|Y)$ isn't too different from $P(X)$

$$P_{LIN}(x|y) = \alpha \hat{P}(x|y) + (1.0 - \alpha) \hat{P}(x)$$

- What if α is 0? 1?

Real NB: Smoothing

- For real classification problems, smoothing is critical
- New odds ratios:

$$\frac{P(W|\text{ham})}{P(W|\text{spam})}$$

helvetica	:	11.4
seems	:	10.8
group	:	10.2
ago	:	8.4
areas	:	8.3
...		

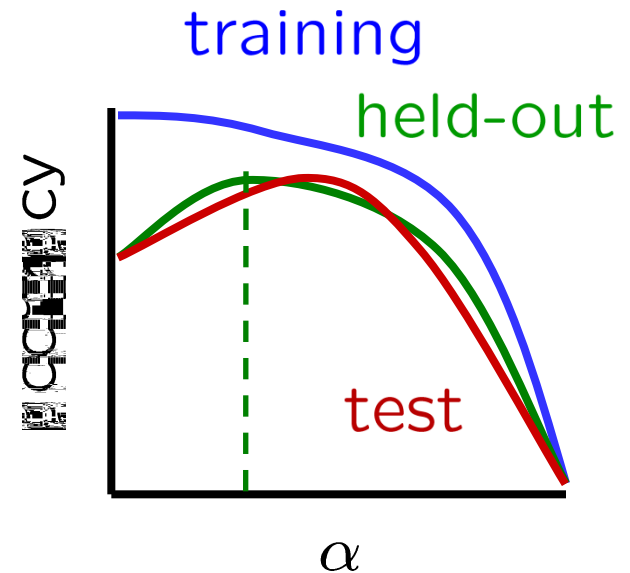
$$\frac{P(W|\text{spam})}{P(W|\text{ham})}$$

verdana	:	28.8
Credit	:	28.4
ORDER	:	27.2
	:	26.9
money	:	26.5
...		

Do these make more sense?

Tuning on Held-Out Data

- Now we've got two kinds of unknowns
 - Parameters: the probabilities $P(Y|X)$, $P(Y)$
 - Hyperparameters, like the amount of smoothing to do: k
- How to learn?
 - Learn parameters from training data
 - Must tune hyperparameters on different data
 - Why?
 - For each value of the hyperparameters, train and test on the held-out (validation) data
 - Choose the best value and do a final test on the test data



How to Learn

- **Data:** labeled instances, e.g. emails marked spam/ham
 - Training set
 - Held out (validation) set
 - Test set
- **Features:** attribute-value pairs which characterize each x
- **Experimentation cycle**
 - Learn parameters (e.g. model probabilities) on training set
 - Tune hyperparameters on held-out set
 - Compute accuracy on test set
 - Very important: never “peek” at the test set!
- **Evaluation**
 - Accuracy: fraction of instances predicted correctly
- **Overfitting and generalization**
 - Want a classifier which does well on *test* data
 - Overfitting: fitting the training data very closely, but not generalizing well to test data

Training
Data

Held-Out
Data

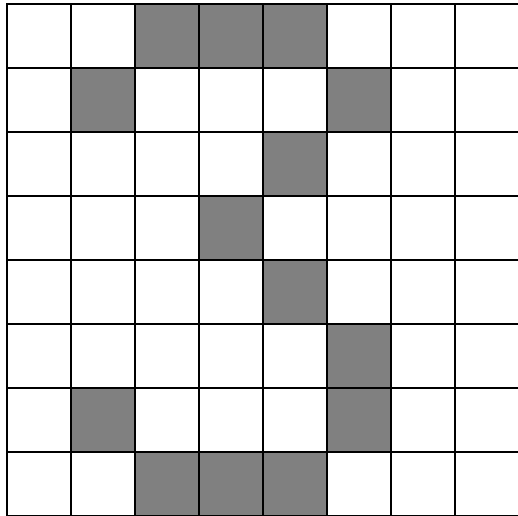
Test
Data

What to Do About Errors?

- Need more features words aren't enough!
 - Have you emailed the sender before?
 - Have 1K other people just gotten the same email?
 - Is the sending information consistent?
 - Is the email in ALL CAPS?
 - Do inline URLs point where they say they point?
 - Does the email address you by (your) name?
- Can add these information sources as new variables in the Naïve Bayes model

A Digit Recognizer

- Input: x = pixel grids



- Output: y = a digit 0-9



Example: Digit Recognition

- Input: x = images (pixel grids)
- Output: y = a digit 0-9
- Setup:
 - Get a large collection of example images, each labeled with a digit
 - Note: someone has to hand label all this data!
 - Want to learn to predict labels of new, future digit images
- Features: The attributes used to make the digit decision
 - Pixels: (6,8)=ON
 - Shape Patterns: NumComponents, AspectRatio, NumLoops
 -



0



1



2



1



??

Naïve Bayes for Digits

- Simple version:

- One feature F_{ij} for each grid position $\langle i,j \rangle$
- Boolean features
- Each input maps to a feature vector, e.g.

1 $\rightarrow \langle F_{0,0} = 0 \ F_{0,1} = 0 \ F_{0,2} = 1 \ F_{0,3} = 1 \ F_{0,4} = 0 \ \dots F_{15,15} = 0 \rangle$

- Here: lots of features, each is binary valued

- Naïve Bayes model:

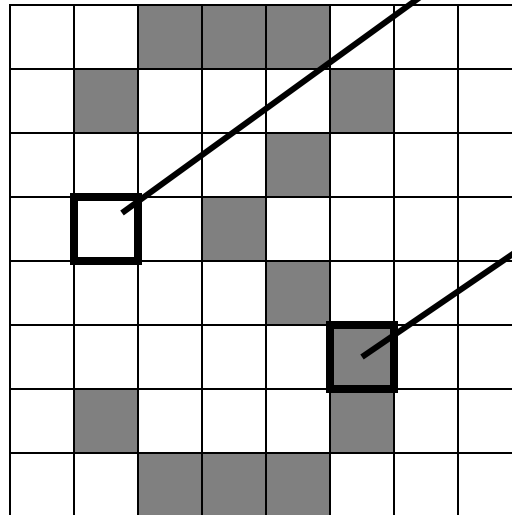
$$P(Y|F_{0,0} \dots F_{15,15}) \propto P(Y) \prod_{i,j} P(F_{i,j}|Y)$$

Learning Model Parameters

$$P(Y, F_1 \dots F_n) = P(Y) \prod_i P(F_i | Y)$$

$P(Y)$

1	0.1
2	0.1
3	0.1
4	0.1
5	0.1
6	0.1
7	0.1
8	0.1
9	0.1
0	0.1



$P(F_{3,1} = \text{on} | Y)$

$P(F_{5,5} = \text{on} | Y)$

1	0.01
2	0.05
3	0.05
4	0.30
5	0.80
6	0.90
7	0.05
8	0.60
9	0.50
0	0.80

1	0.05
2	0.01
3	0.90
4	0.80
5	0.90
6	0.90
7	0.25
8	0.85
9	0.60
0	0.80

Problem: Overfitting

$P(\text{features}, C = 2)$

$$P(C = 2) = 0.1$$

$$P(\text{on}|C = 2) = 0.8$$

$$P(\text{on}|C = 2) = 0.1$$

$$P(\text{off}|C = 2) = 0.1$$

$$P(\text{on}|C = 2) = 0.01$$

$P(\text{features}, C = 3)$

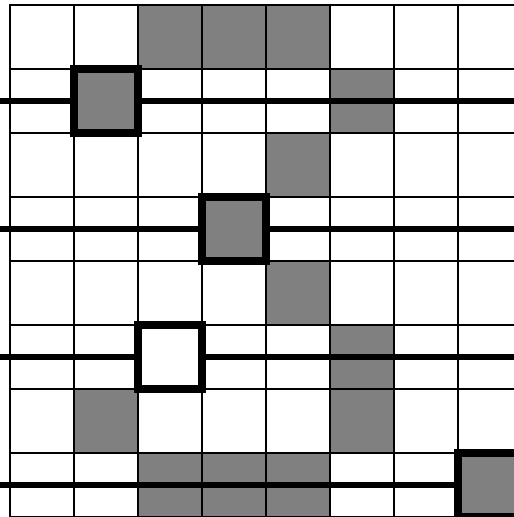
$$P(C = 3) = 0.1$$

$$P(\text{on}|C = 3) = 0.8$$

$$P(\text{on}|C = 3) = 0.9$$

$$P(\text{off}|C = 3) = 0.7$$

$$P(\text{on}|C = 3) = 0.0$$



2 wins!!



Outline

- Machine Learning
- Classification (Naïve Bayes)
- Regression (Linear, Smoothing)
- Linear Separation (Perceptron, SVMs)
- Non-parametric classification (KNN)

Regression

- Start with very simple example
 - Linear regression
- What you learned in high school math
 - From a new perspective
- Linear model
 - $y = m x + b$
 - $h_{\mathbf{w}}(x) = y = w_1 x + w_0$
- Find best values for parameters
 - “maximize goodness of fit”
 - “maximize probability” or “minimize loss”

Regression: Minimizing Loss

- Assume true function f is given by

$$y = f(x) = m x + b + \text{noise}$$

where noise is normally distributed

- Then most probable values of parameters found by minimizing squared-error loss:

$$\text{Loss}(h_{\mathbf{w}}) = \sum_j (y_j - h_{\mathbf{w}}(x_j))^2$$

Regression: Minimizing Loss

For what w is

$$\prod_{i=1}^n P(y_i | w, x_i) \text{ maximized?}$$

For what w is

$$\prod_{i=1}^n \exp\left(-\frac{1}{2}\left(\frac{y_i - wx_i}{\sigma}\right)^2\right) \text{ maximized?}$$

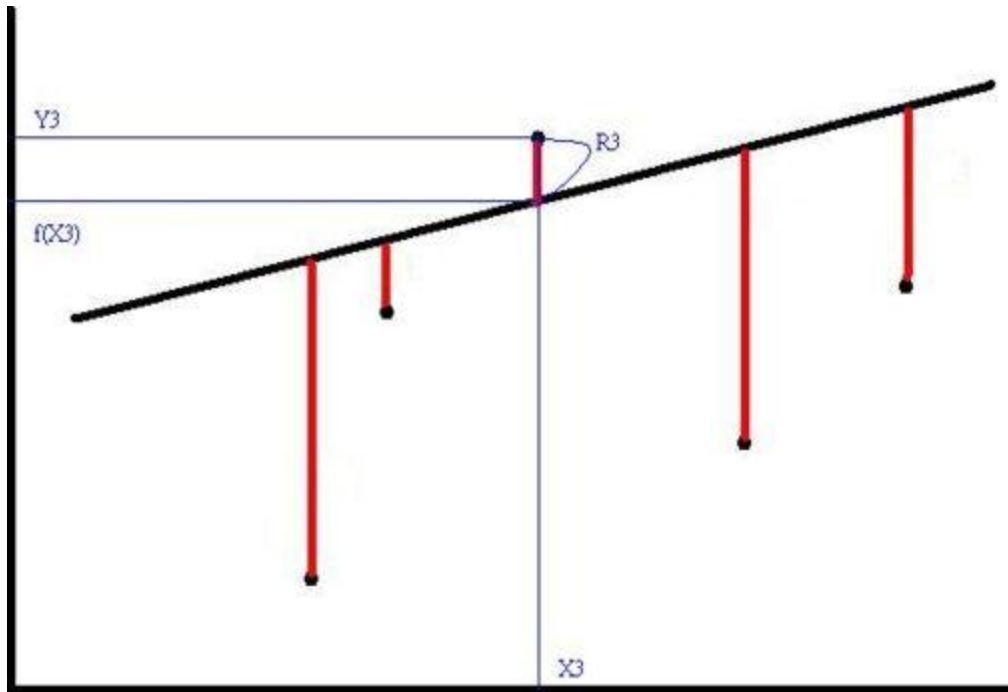
For what w is

$$\sum_{i=1}^n -\frac{1}{2}\left(\frac{y_i - wx_i}{\sigma}\right)^2 \text{ maximized?}$$

For what w is

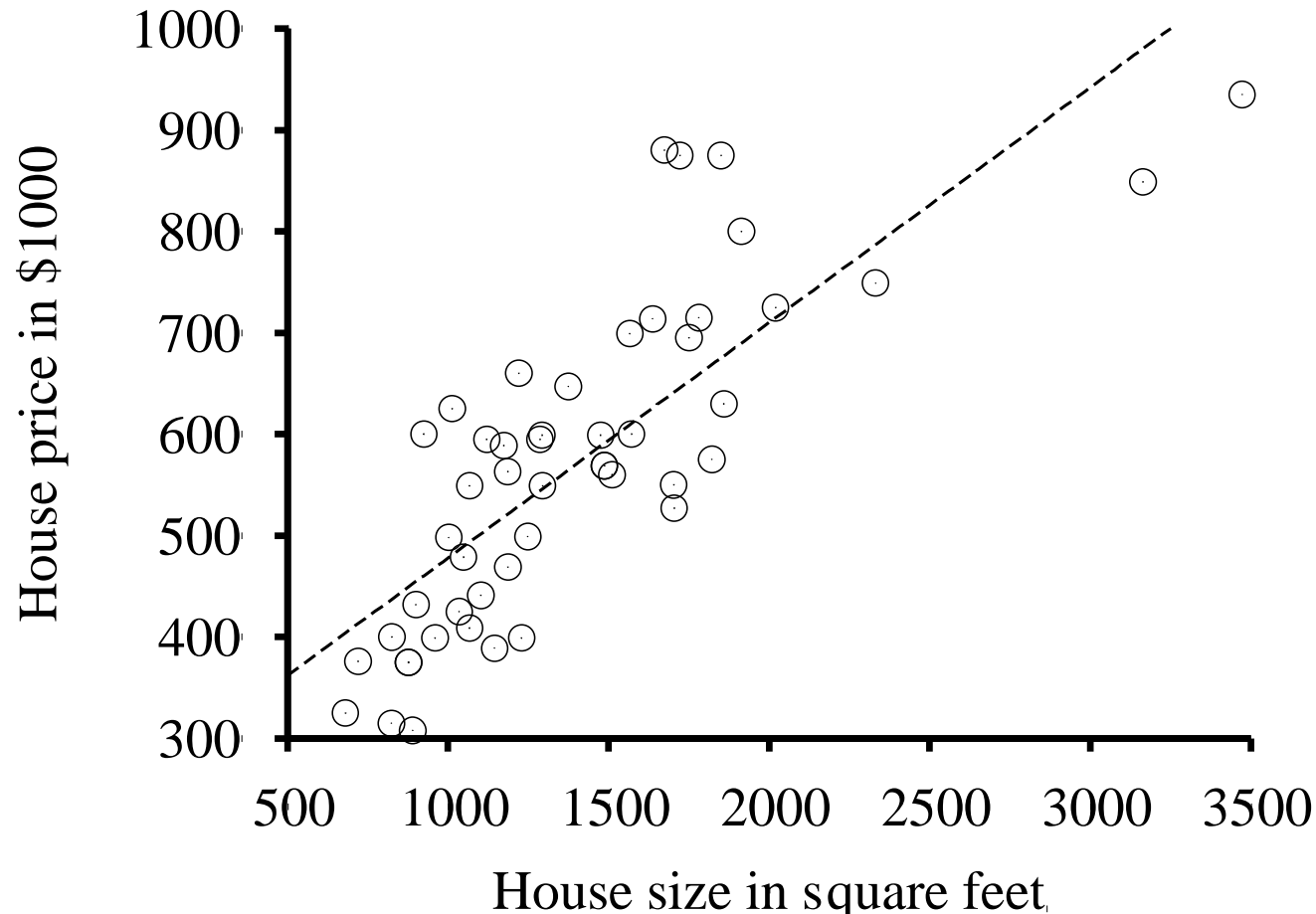
$$\sum_{i=1}^n (y_i - wx_i)^2 \text{ minimized?}$$

Regression: Minimizing Loss

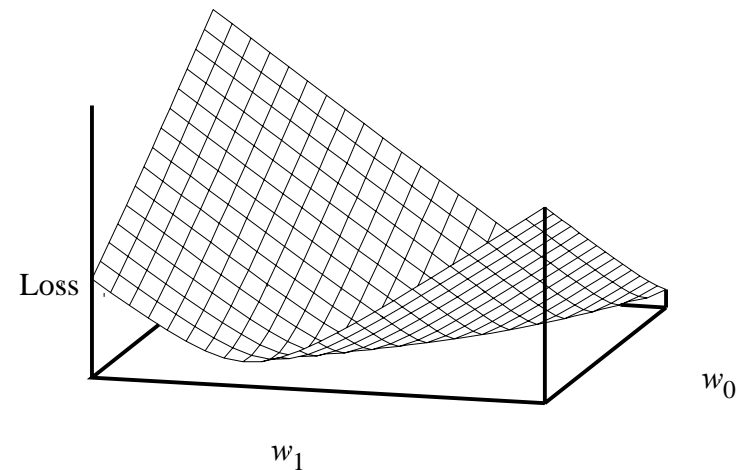
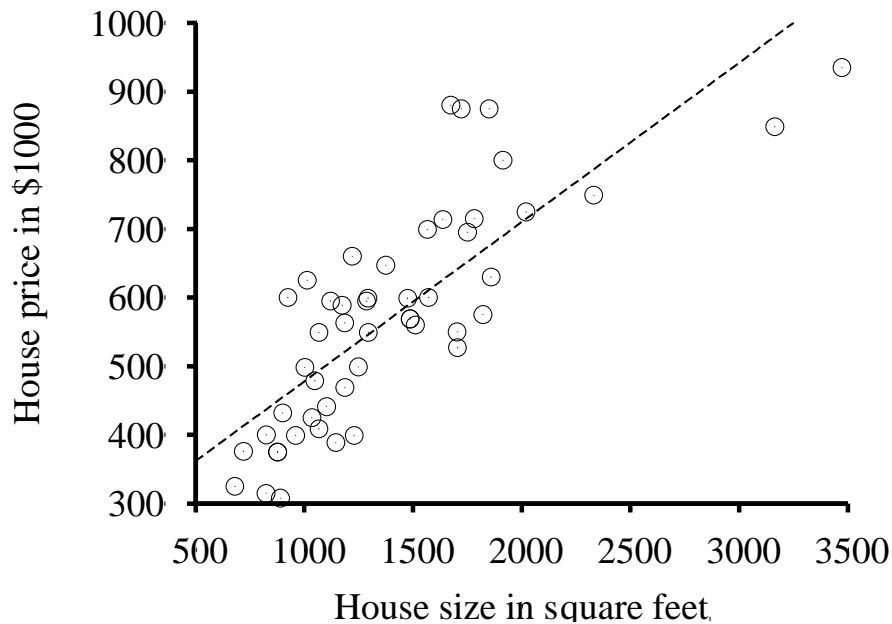


Choose weights to minimize
sum of squared errors

Regression: Minimizing Loss



Regression: Minimizing Loss



$$y = w_1 x + w_0$$

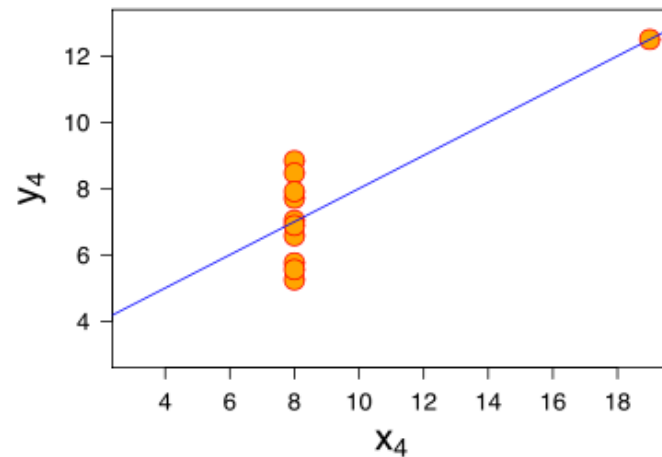
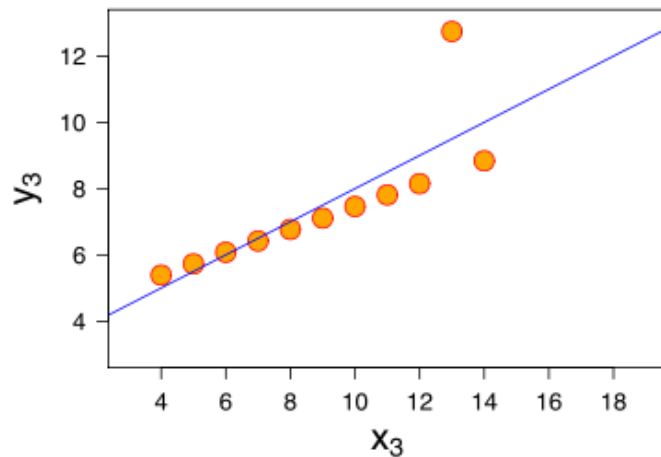
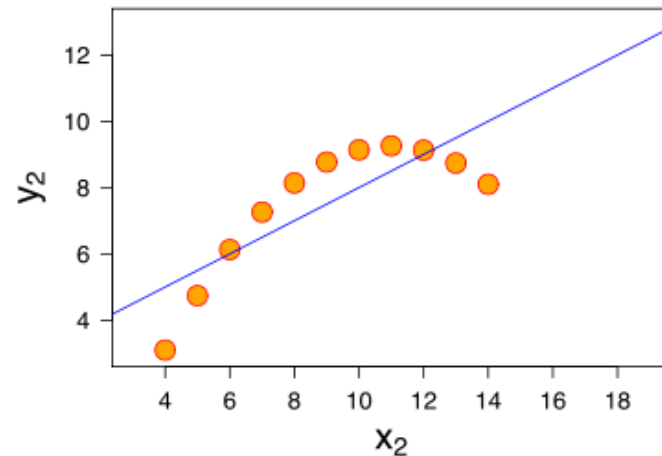
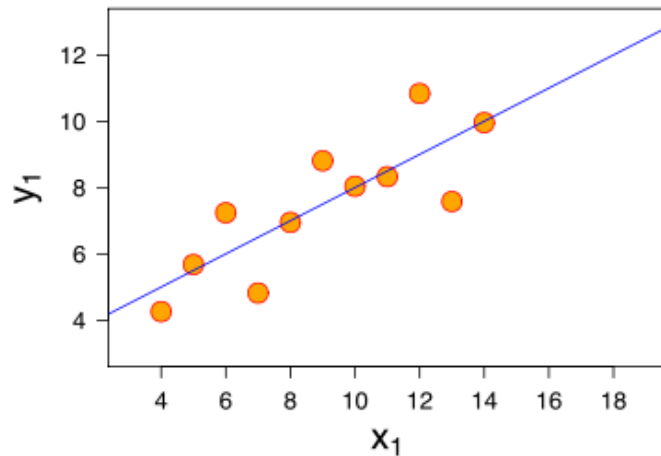
Linear algebra gives an exact solution to the minimization problem

Linear Algebra Solution

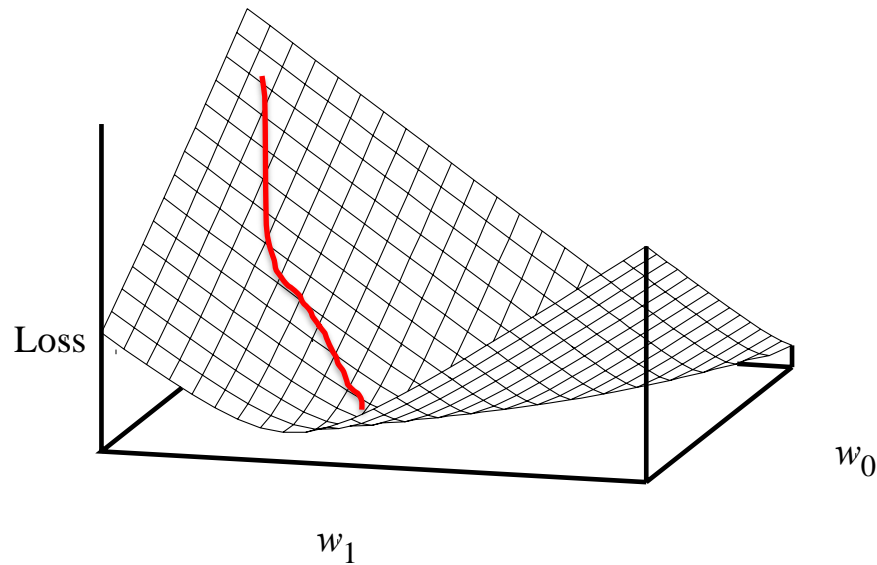
$$w_1 = \frac{M \bar{a} x_i y_i - \bar{a} x_i \bar{a} y_i}{M \bar{a} x_i^2 - (\bar{a} x_i)^2}$$

$$w_0 = \frac{1}{M} \bar{a} y_i - \frac{w_1}{M} \bar{a} x_i$$

Don't Always Trust Linear Models



Regression by Gradient Descent



\mathbf{w} = any point

loop until convergence do:

for each w_i in \mathbf{w} do:

$$w_i = w_i - \alpha \frac{\partial}{\partial w_i} Loss(\mathbf{w})$$

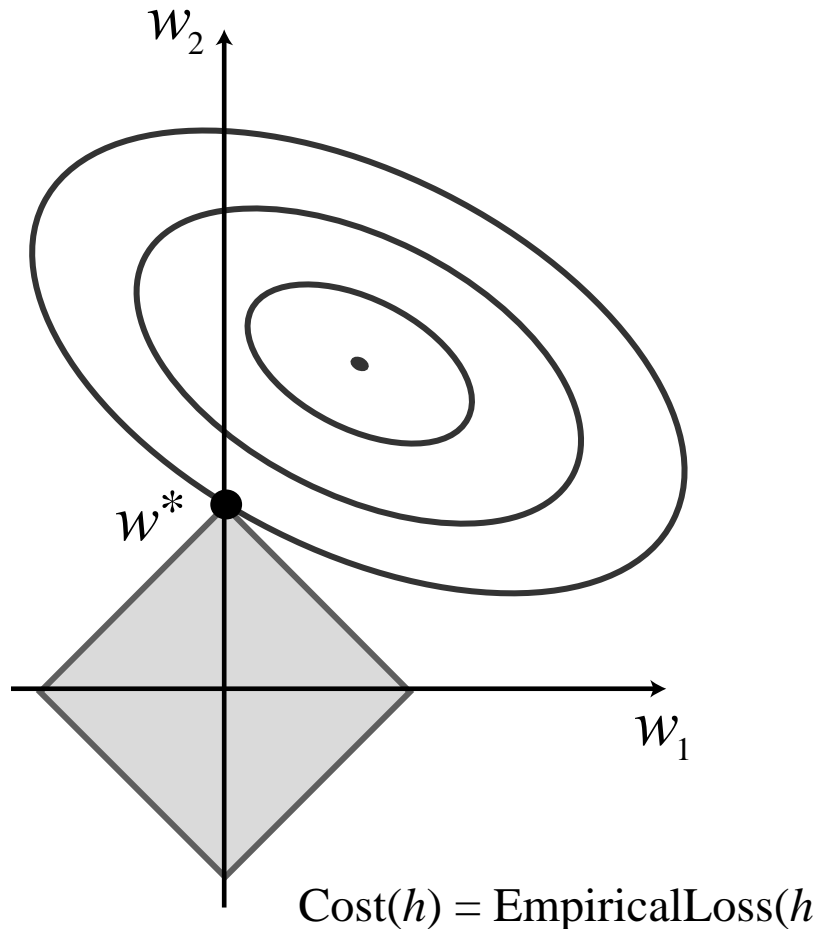
Multivariate Regression

- You learned this in math class too
 - $h_{\mathbf{w}}(\mathbf{x}) = \sum_i w_i x_i = \mathbf{w}^T \mathbf{x}$
- The most probable set of weights, \mathbf{w}^* (minimizing squared error):
 - $\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$

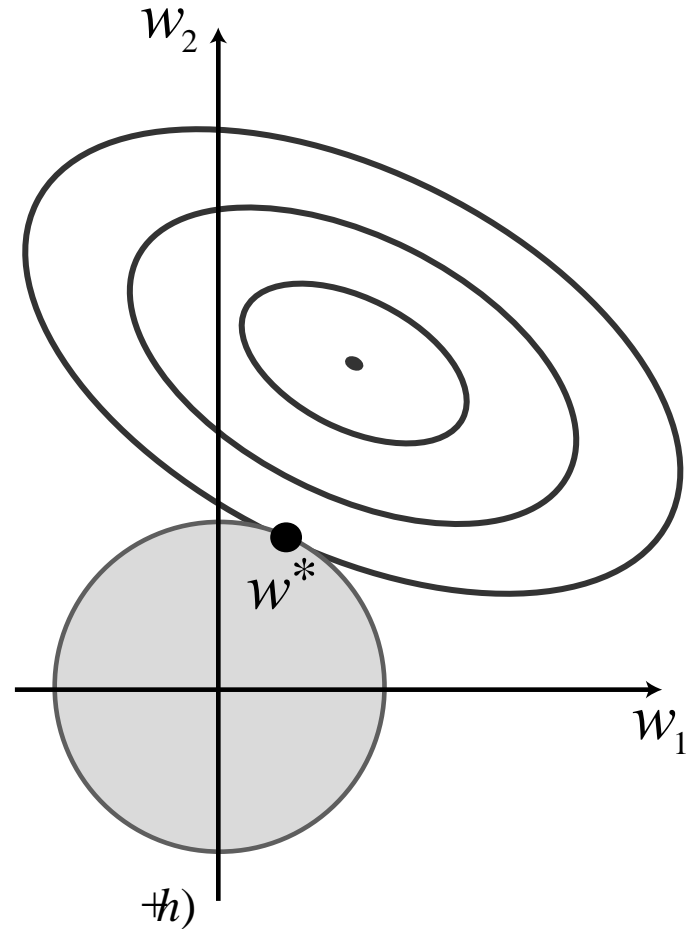
Overfitting

- To avoid overfitting, don't just minimize loss
- Maximize probability, including prior over \mathbf{w}
- Can be stated as minimization:
 - $\text{Cost}(h) = \text{EmpiricalLoss}(h) + \text{Complexity}(h)$
- For linear models, consider
 - $\text{Complexity}(h_{\mathbf{w}}) = L_q(\mathbf{w}) = \sum_i |w_i|^q$
 - L_1 regularization minimizes sum of abs. values
 - L_2 regularization minimizes sum of squares

Regularization and Sparsity



L_1 regularization

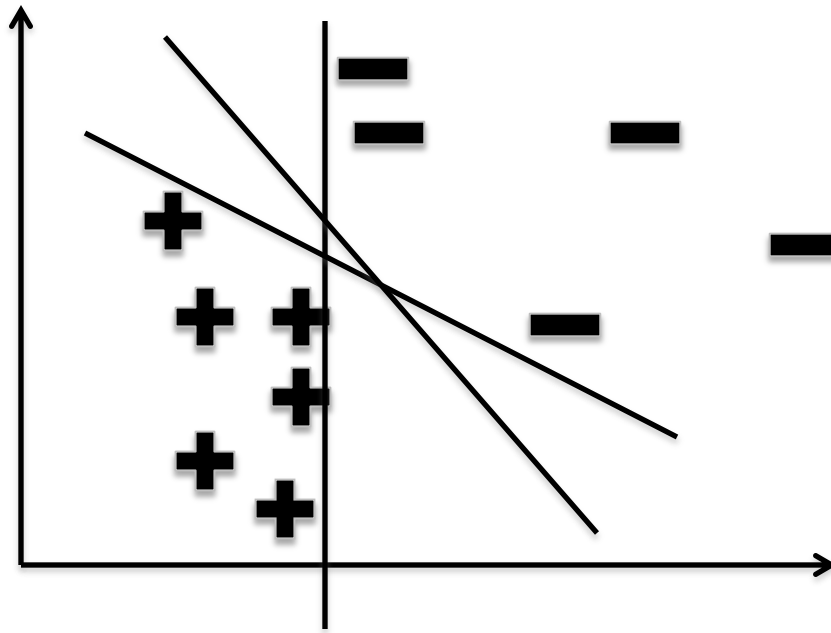


L_2 regularization

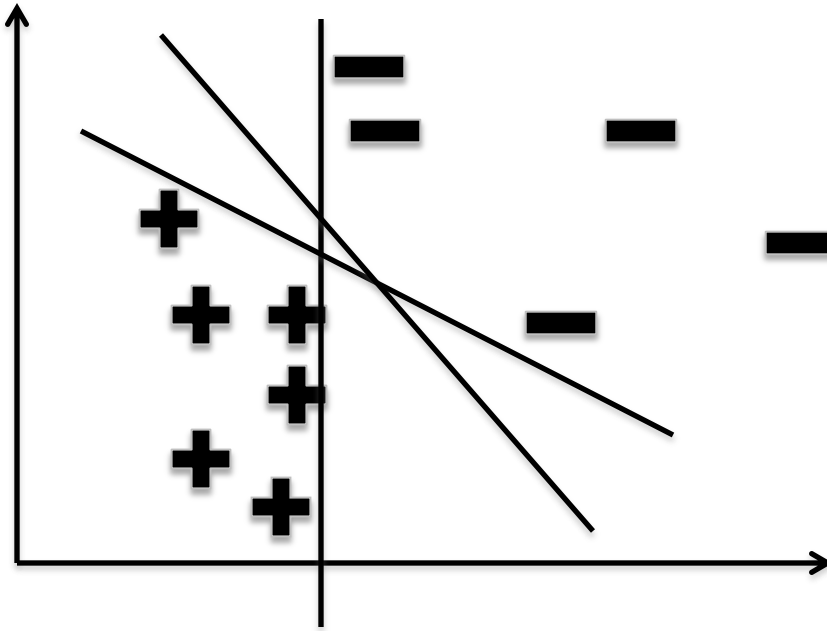
Outline

- Machine Learning
- Classification (Naïve Bayes)
- Regression (Linear, Smoothing)
- Linear Separation (Perceptron, SVMs)
- Non-parametric classification (KNN)

Linear Separator



Perceptron

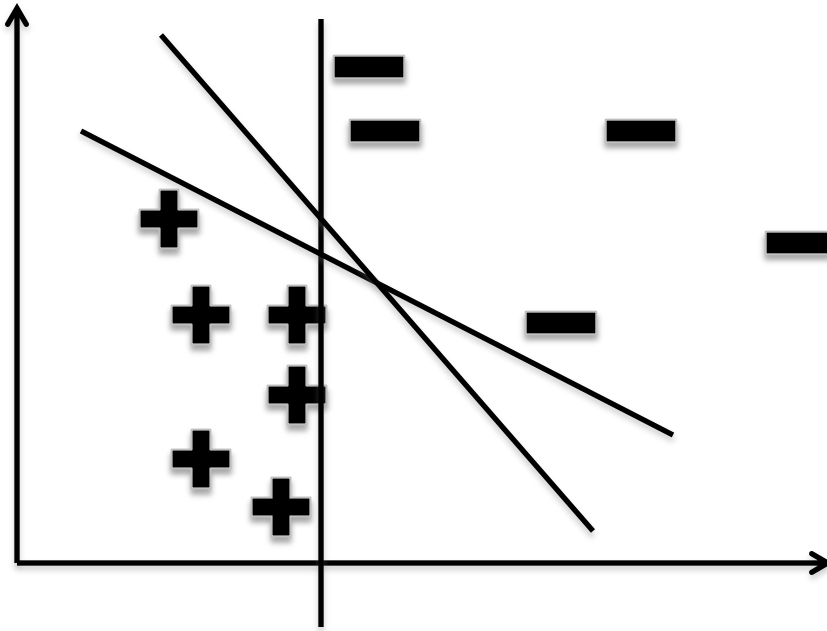


$$f(x) = \begin{cases} 1 & \text{if } w_1 x + w_0 \geq 0 \\ 0 & \text{if } w_1 x + w_0 < 0 \end{cases}$$

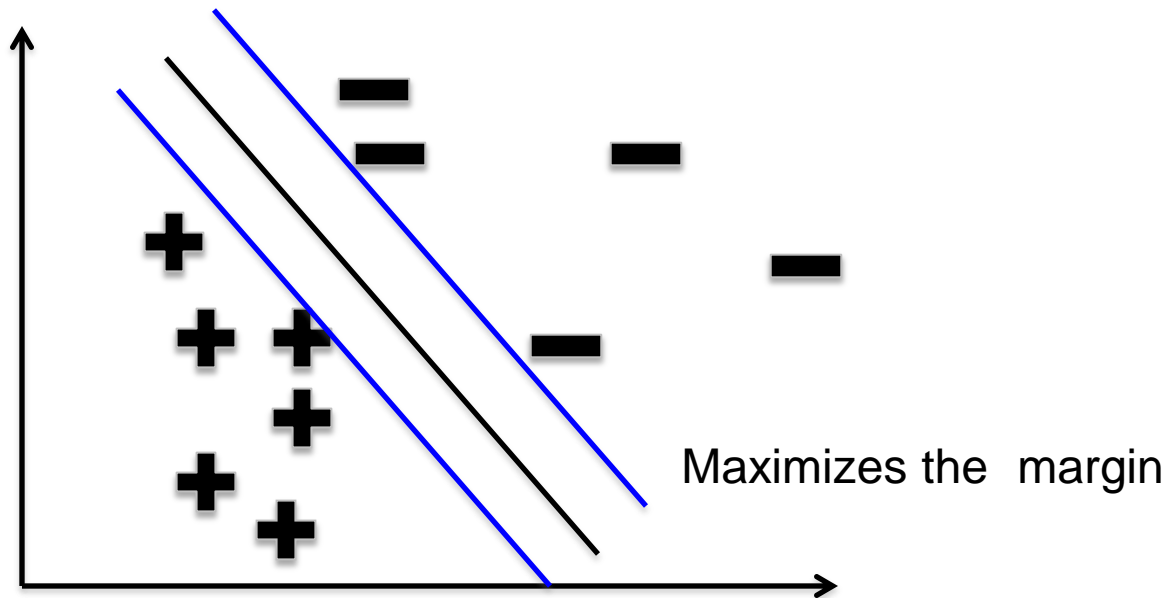
Perceptron Algorithm

- Start with random w_0, w_1
- Pick training example $\langle x, y \rangle$
- If $f(x) \neq y$
 - $w_1 \leftarrow w_1 - yx$
 - $w_0 \leftarrow w_0 - y$
- Converges to linear separator (if exists)
- Picks a linear separator (a good one?)

What Linear Separator to Pick?

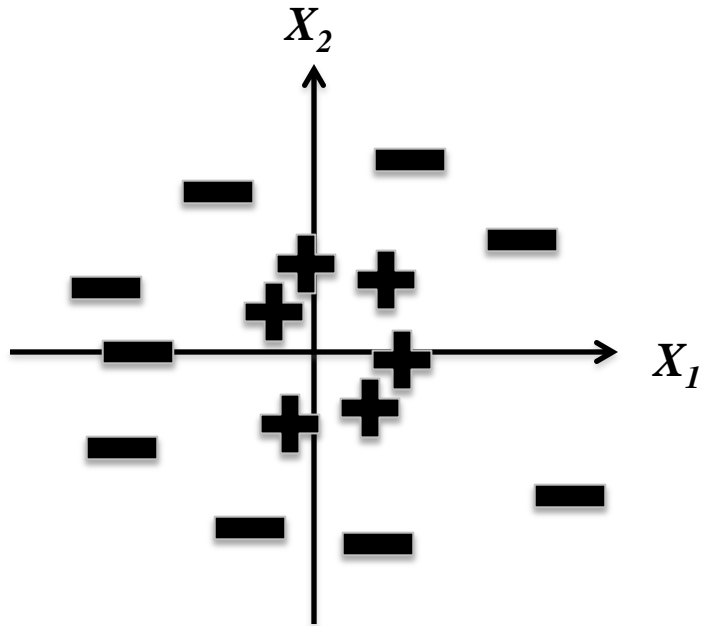


What Linear Separator to Pick?



Support Vector Machines

Non-Separable Data?



- Not linearly separable for x_1, x_2
- What if we add a feature?
- $x_3 = x_1^2 + x_2^2$
- See: Kernel Trick



Outline

- Machine Learning
- Classification (Naïve Bayes)
- Regression (Linear, Smoothing)
- Linear Separation (Perceptron, SVMs)
- Non-parametric classification (KNN)

Nonparametric Models

- If the process of learning good values for parameters is prone to overfitting, can we do without parameters?

Nearest-Neighbor Classification

- Nearest neighbor for digits:
 - Take new image
 - Compare to all training images
 - Assign based on closest example



- Encoding: image is vector of intensities:

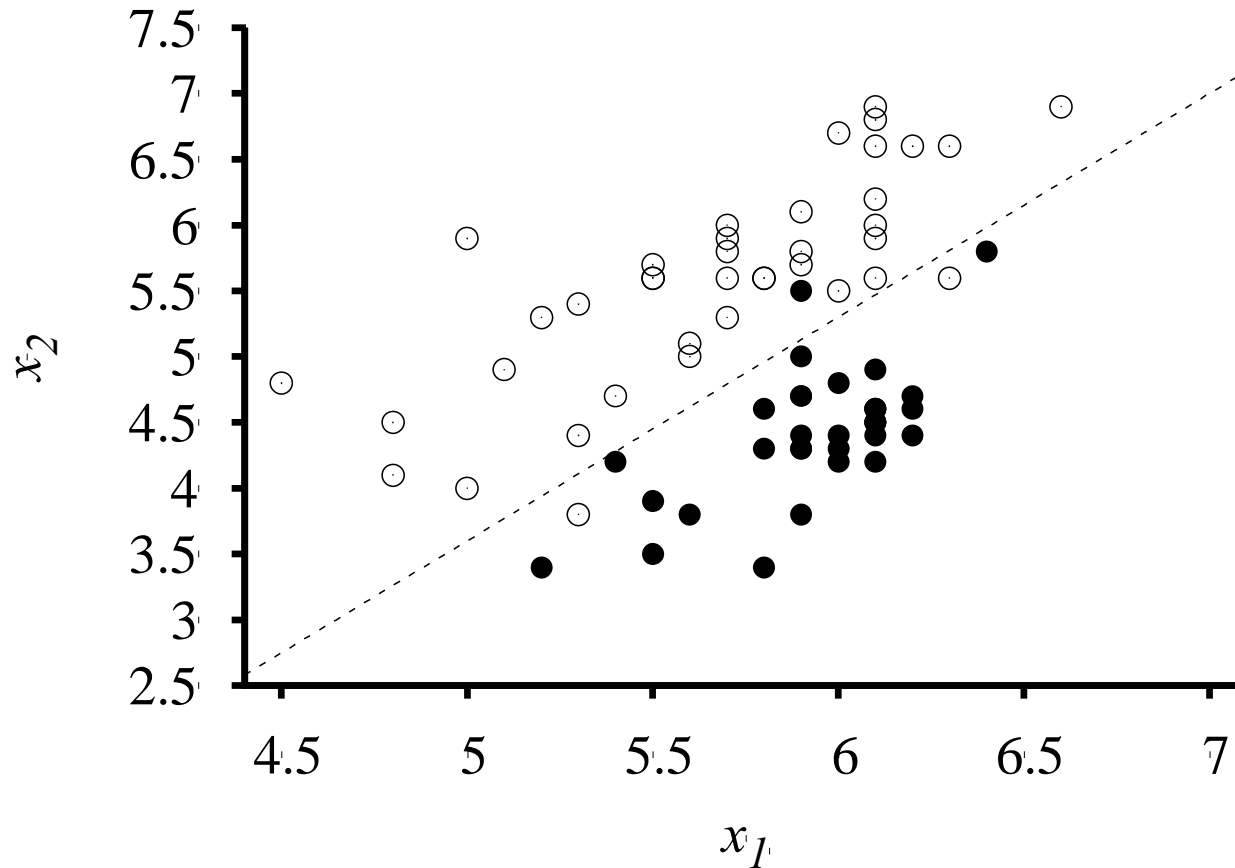
$$1 = \langle 0.0 \ 0.0 \ 0.3 \ 0.8 \ 0.7 \ 0.1 \dots 0.0 \rangle$$

- What's the similarity function?
 - Dot product of two images vectors?

$$\text{sim}(x, y) = x \cdot y = \sum_i x_i y_i$$

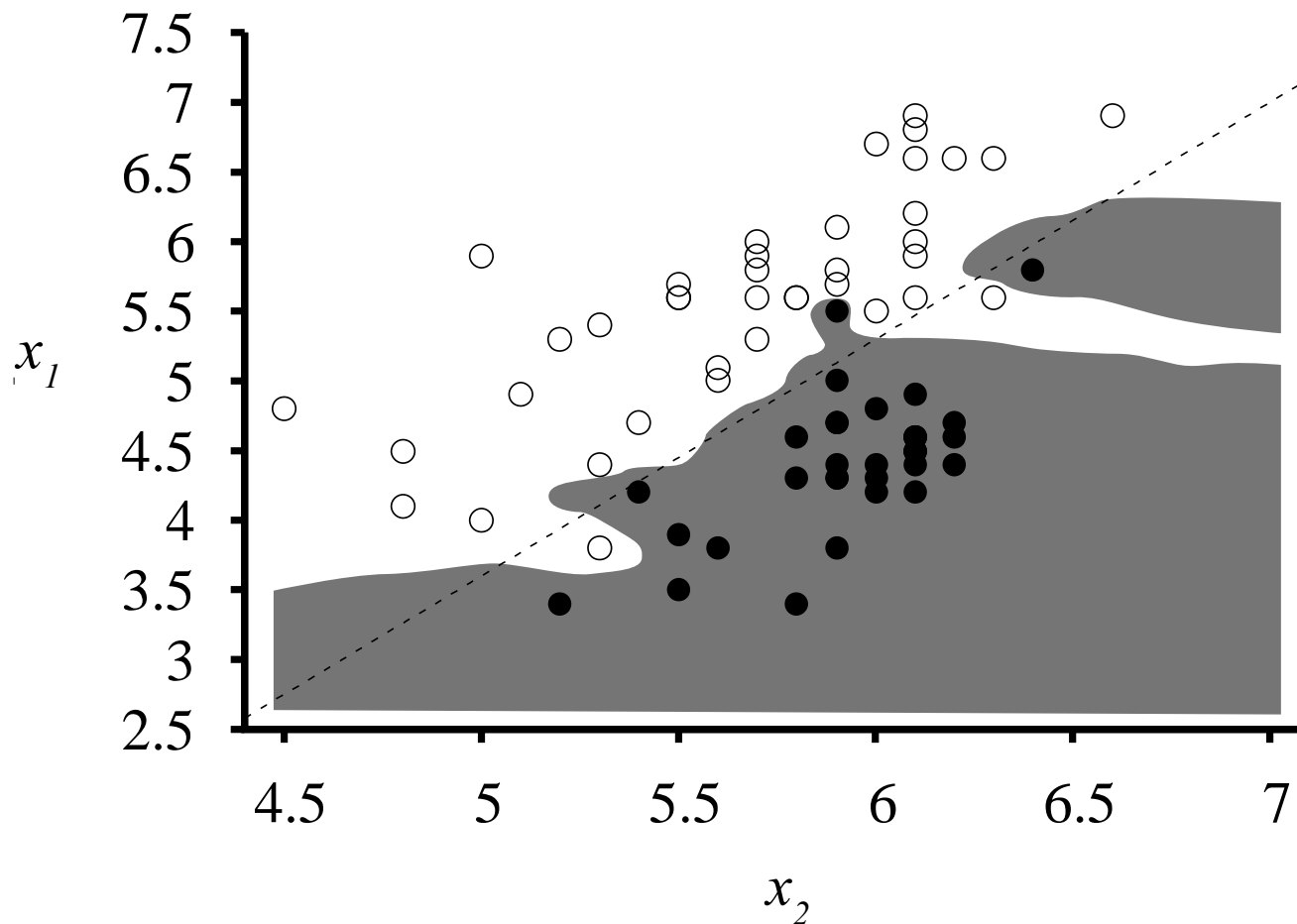
- Usually normalize vectors so $\|x\| = 1$
- min = 0 (when?), max = 1 (when?)

Earthquakes and Explosions



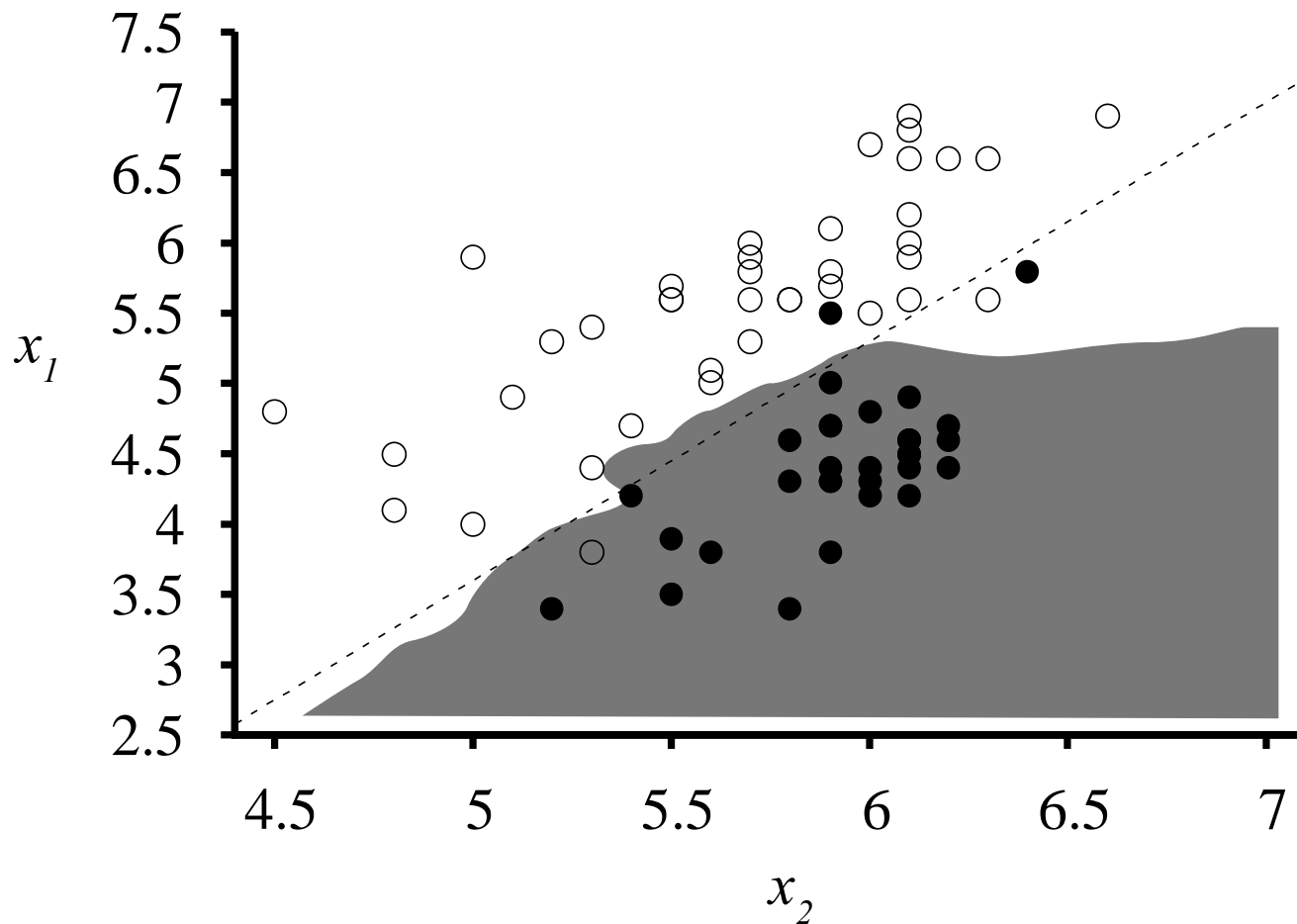
Using logistic regression (similar to linear regression) to do linear classification

$K=1$ Nearest Neighbors



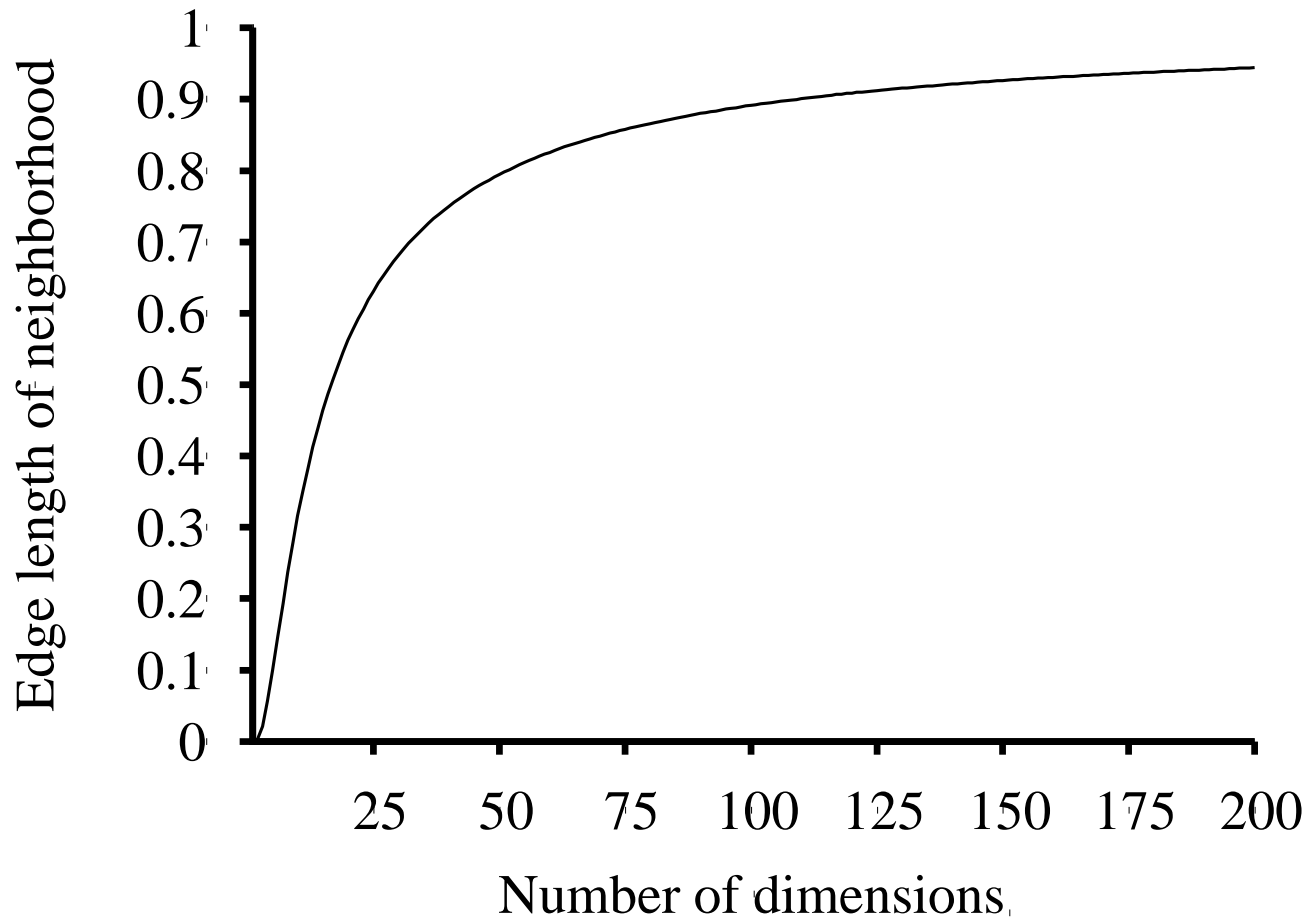
Using nearest neighbors to do classification

$K=5$ Nearest Neighbors



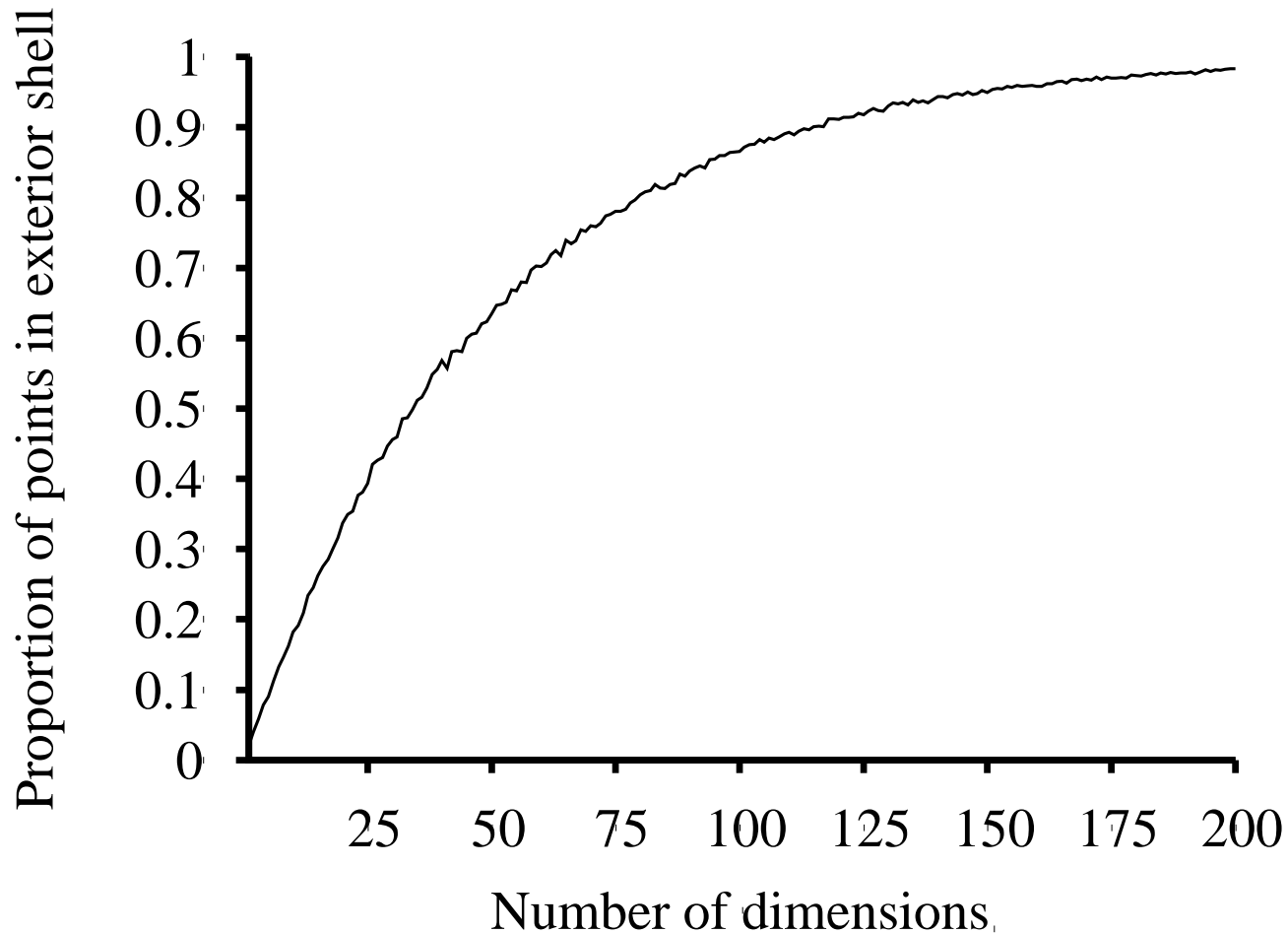
Even with no parameters, you still have hyperparameters!

Curse of Dimensionality



Average neighborhood size for 10-nearest neighbors, n dimensions, 1M uniform points

Curse of Dimensionality



Proportion of points that are within the outer shell, 1% of thickness of the hypercube

Outline

- Machine Learning
- Classification (Naïve Bayes)
- Regression (Linear, Smoothing)
- Linear Separation (Perceptron, SVMs)
- Non-parametric classification (KNN)

Machine Learning Lingo

What?	Parameters	Structure	Hidden concepts	
What from?	Supervised	Unsupervised	Reinforcement	Self-supervised
What for?	Prediction	Diagnosis	Compression	Discovery
How?	Passive	Active	Online	Offline
Output?	Classification	Regression	Clustering	
Details??	Generative	Discriminative	Smoothing	