

# Instance-based Learning

Machine Learning – 10701/15781  
Carlos Guestrin  
Carnegie Mellon University

October 15<sup>th</sup>, 2007

©2005-2007 Carlos Guestrin

1

## 1-Nearest Neighbor

Four things make a memory based learner:

1. A distance metric  
**Euclidian (and many more)**
2. How many nearby neighbors to look at?  
**One**
3. A weighting function (optional)  
**Unused**
4. How to fit with the local points?  
**Just predict the same output as the nearest neighbor.**



©2005-2007 Carlos Guestrin

2

# Consistency of 1-NN

- Consider an estimator  $f_n$  trained on  $n$  examples
  - e.g., 1-NN, neural nets, regression,...
- Estimator is consistent if true error goes to zero as amount of data increases

- e.g., for no noise data, consistent if:

$$\lim_{n \rightarrow \infty} \underline{MSE}(f_n) = 0$$

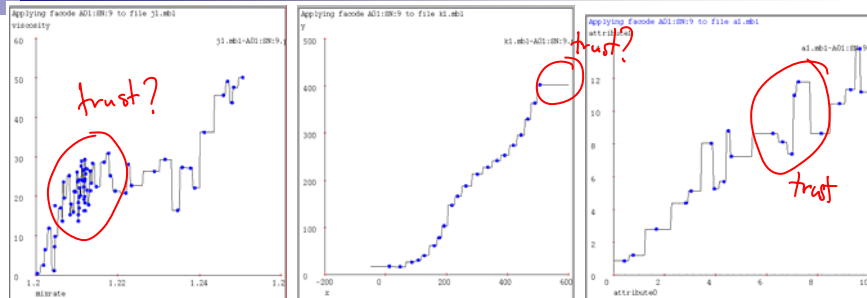
- Regression is not consistent!
  - Representation bias
- **1-NN is consistent** (under some mild fineprint)

**What about variance???**

©2005-2007 Carlos Guestrin

3

# 1-NN overfits?



©2005-2007 Carlos Guestrin

4

# k-Nearest Neighbor

Four things make a memory based learner:

1. A distance metric  
**Euclidian (and many more)**
2. How many nearby neighbors to look at?  
**k**
1. A weighting function (optional)  
**Unused**
2. How to fit with the local points?

**Just predict the average output among the k nearest neighbors.**

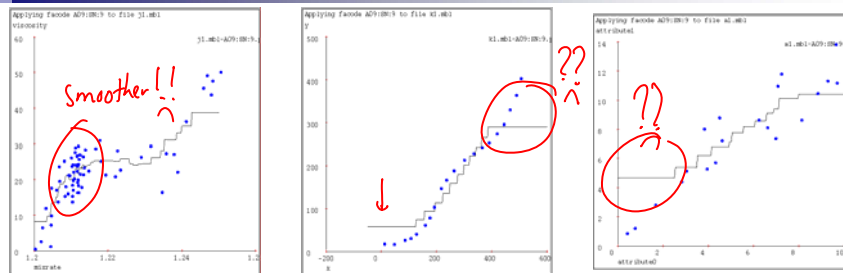
$$x_q = \text{query} \quad N(x_q)$$

$$\hat{y}_{x_q} = \frac{1}{k} \sum_{x_i \in N(x_q)} y_i$$

©2005-2007 Carlos Guestrin

5

# k-Nearest Neighbor (here k=9)



less variance, but more bias  
(still consistent, see above for details, if k fixed as  $n \rightarrow \infty$ )

**K-nearest neighbor for function fitting smoothes away noise, but there are clear deficiencies.**

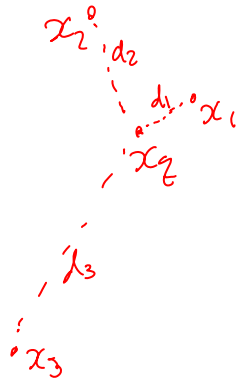
What can we do about all the discontinuities that k-NN gives us?

©2005-2007 Carlos Guestrin

6

# Weighted k-NNs

- Neighbors are not all the same (an example)



eg. 
$$\vec{y}_q = \frac{\sum_{x_i \in \mathcal{N}(x_q)} \frac{1}{d_i} y_i}{\sum_{x_i \in \mathcal{N}(x_q)} \frac{1}{d_i}}$$

normalize to get a convex combination

©2005-2007 Carlos Guestrin

7

# Kernel regression

good work for high: learn distance function (learn important features)

everything (doesn't work well in high-dim, unless lots of data)

Four things make a memory based learner:

- A distance metric  
**Euclidian (and many more)**
- How many nearby neighbors to look at?  
**All of them**
- A weighting function (optional)  
 $w_i = \exp(-D(x_i, \text{query})^2 / K_w^2)$

Nearby points to the query are weighted strongly, far points weakly. The  $K_w$  parameter is the **Kernel Width**. Very important.

- How to fit with the local points?

Predict the weighted average of the outputs:

predict =  $\sum w_i y_i / \sum w_i$

$$\vec{y}_q = \frac{\sum_{i=1}^m w_i y_i}{\sum_{i=1}^m w_i}$$



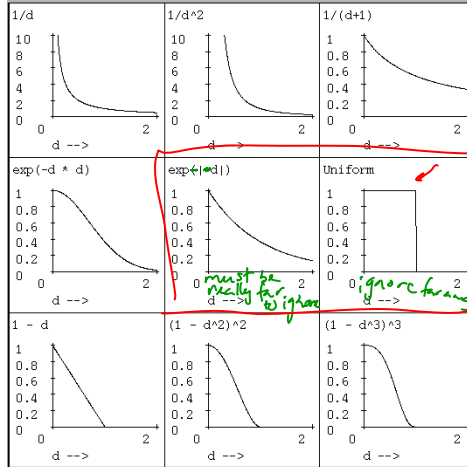
©2005-2007 Carlos Guestrin

8

# Weighting functions

$$w_i = \exp(-D(x_i, \text{query})^2 / K_w^2)$$

Squared exponential | Kernel  
Gaussian



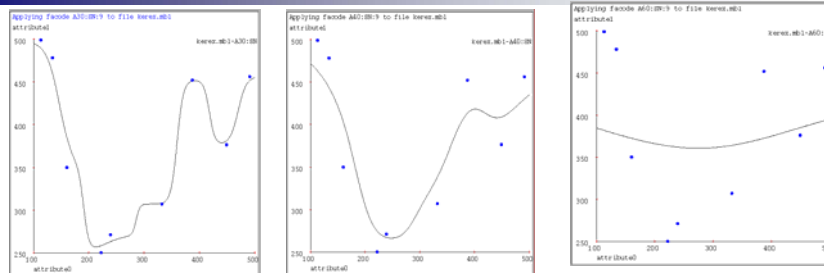
Typically optimize  $K_w$   
using gradient descent

(Our examples use Gaussian)

©2005-2007 Carlos Guestrin

9

# Kernel regression predictions



$K_w=10$

$K_w=20$

$K_w=80$

bias ↓  
var ↑

↑  
↓

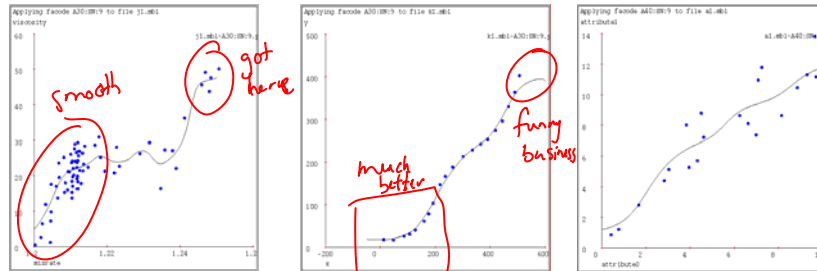
Increasing the kernel width  $K_w$  means further away points get an opportunity to influence you.

As  $K_w \rightarrow \infty$ , the prediction tends to the global average.

©2005-2007 Carlos Guestrin

10

# Kernel regression on our test cases



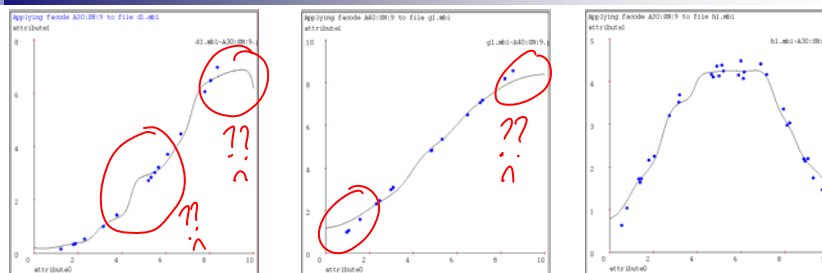
KW=1/32 of x-axis width.

KW=1/32 of x-axis width.

KW=1/16 axis width.

Choosing a good  $K_w$  is important. Not just for Kernel Regression, but for all the locally weighted learners we're about to see. *X validation*

# Kernel regression can look bad



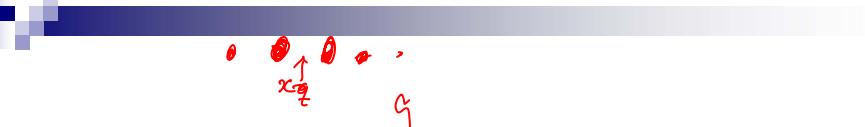
KW = Best.

KW = Best.

KW = Best.

Time to try something more powerful...

# Locally weighted regression



## Kernel regression:

Take a very very conservative function approximator called AVERAGING. Locally weight it.

## Locally weighted regression:

Take a conservative function approximator called LINEAR REGRESSION. Locally weight it.



# Locally weighted regression



- Four things make a memory based learner:

- A distance metric

Any

- How many nearby neighbors to look at?

All of them

- A weighting function (optional)

Kernels

- $w_i = \exp(-D(x_i, query)^2 / Kw^2)$

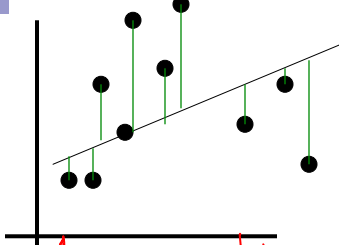
- How to fit with the local points?

## General weighted regression:

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{k=1}^N w_k^2 (y_k - \beta^T x_k)^2$$

weight ↑  $w_k$   
↑  $\sum_i \beta_i x_k^i$

# How LWR works

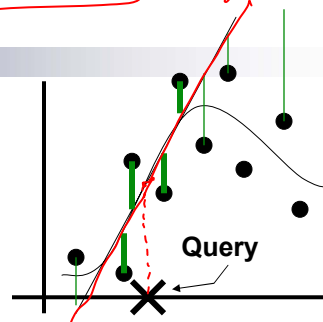


## Linear regression

- Same parameters for all queries

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

constant basis function fits average.



## Locally weighted regression

- Solve weighted linear regression for each query

$$\hat{\beta}_q = ((WX)^T WX)^{-1} (WX)^T WY$$

$$W = \begin{pmatrix} w_1 & 0 & 0 & 0 \\ 0 & w_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & w_n \end{pmatrix}$$

Kernel regression =  
LWLR, if only use  
constant basis function

©2005-2007 Carlos Guestrin

15

# Another view of LWR

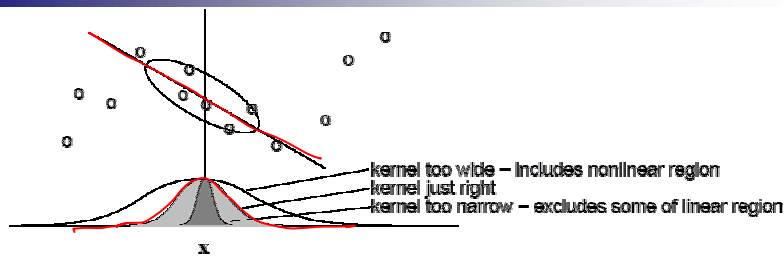
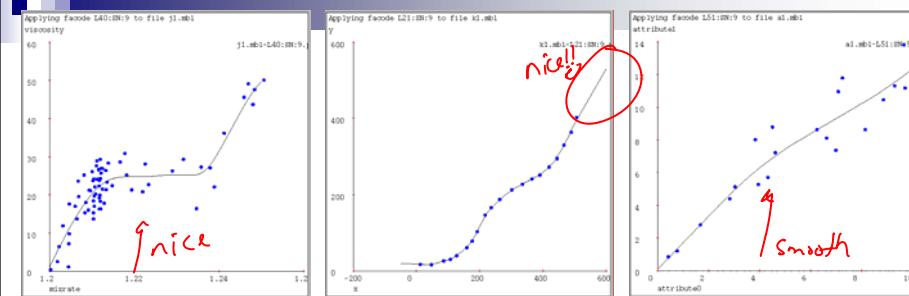


Image from Cohn, D.A., Ghahramani, Z., and Jordan, M. (1996). 'Active Learning with Statistical Models', JAIR Volume 4, pages 129-145.

# LWR on our test cases



KW = 1/16 of x-axis width.

KW = 1/32 of x-axis width.

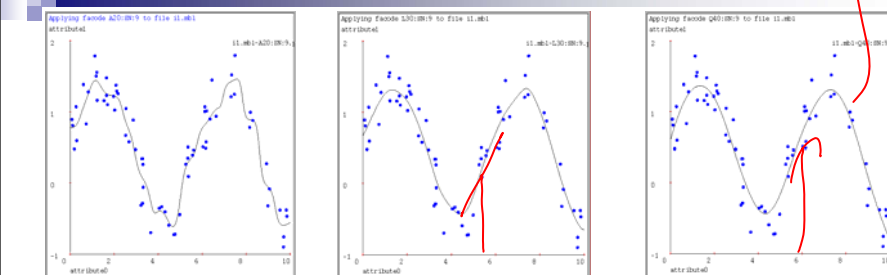
KW = 1/8 of x-axis width.

©2005-2007 Carlos Guestrin

17

# Locally weighted polynomial regression

locally  $a_2 + b_2x + c_2x^2$



Kernel Regression *constant*  
Kernel width  $K_W$  at optimal level.

KW = 1/100 x-axis

LW Linear Regression  
Kernel width  $K_W$  at optimal level.

KW = 1/40 x-axis

LW Quadratic Regression  
Kernel width  $K_W$  at optimal level.

KW = 1/15 x-axis

*(matrix)* Local quadratic regression is easy: just add quadratic terms to the  $WX$  matrix. As the regression degree increases, the kernel width can increase without introducing bias.

©2005-2007 Carlos Guestrin

18

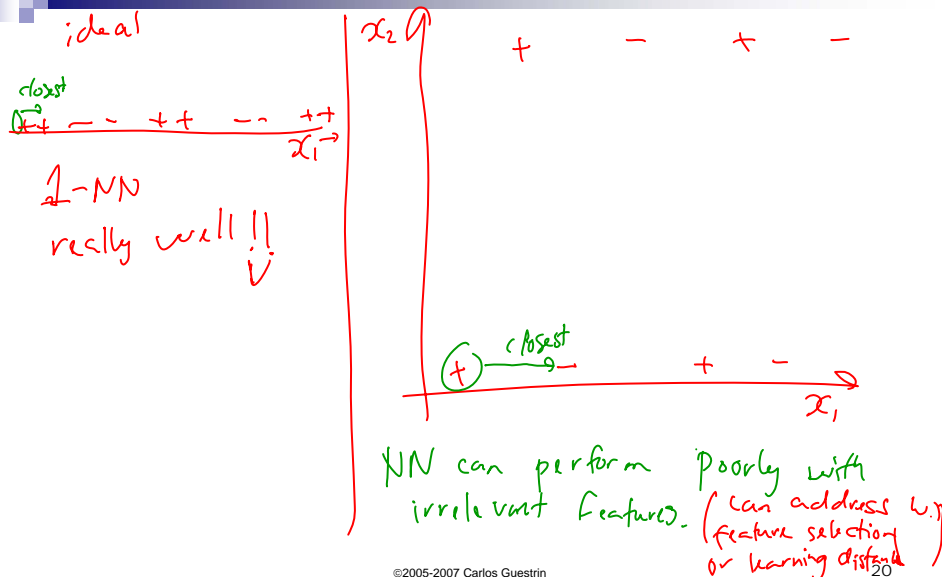
## Curse of dimensionality for instance-based learning

- Must store and retrieve all data!
  - Most real work done during testing
  - For every test sample, must search through all dataset – very slow!
  - We'll see fast methods for dealing with large datasets. → KD-trees  
??-trees
- Instance-based learning often poor with noisy or irrelevant features

©2005-2007 Carlos Guestrin

19

## Curse of the irrelevant feature



©2005-2007 Carlos Guestrin

# What you need to know about instance-based learning

*If NN perfect  
D stump unhappy*

## ■ k-NN

- Simplest learning algorithm
- With sufficient data, very hard to beat “strawman” approach
- Picking k?

## ■ Kernel regression

- Set k to n (number of data points) and optimize weights by gradient descent
- Smoother than k-NN

## ■ Locally weighted regression

- Generalizes kernel regression, not just local average

## ■ Curse of dimensionality

- Must remember (very large) dataset for prediction
- Irrelevant features often killers for instance-based approaches

# Acknowledgment

- This lecture contains some material from Andrew Moore’s excellent collection of ML tutorials:

□ <http://www.cs.cmu.edu/~awm/tutorials>

# Support Vector Machines, SVMs

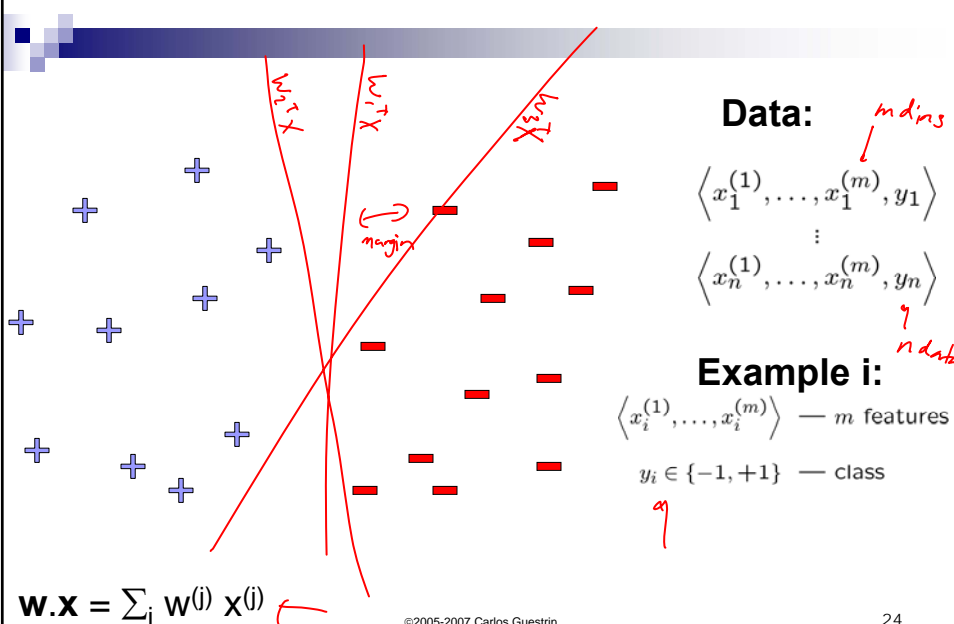
Machine Learning – 10701/15781  
 Carlos Guestrin  
 Carnegie Mellon University

October 15<sup>th</sup>, 2007

©2005-2007 Carlos Guestrin

23

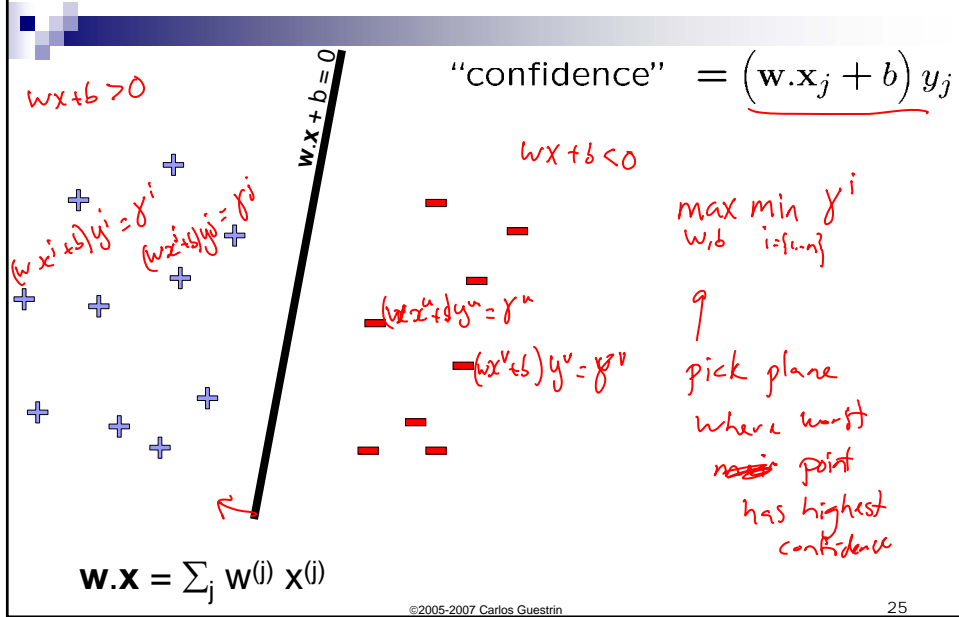
## Linear classifiers – Which line is better?



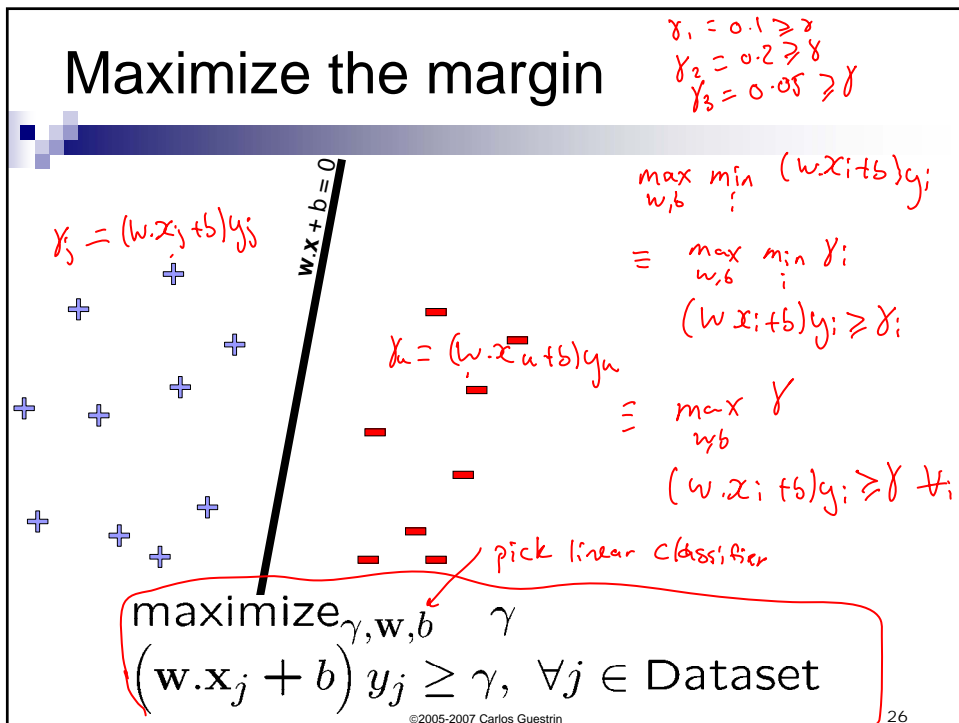
©2005-2007 Carlos Guestrin

24

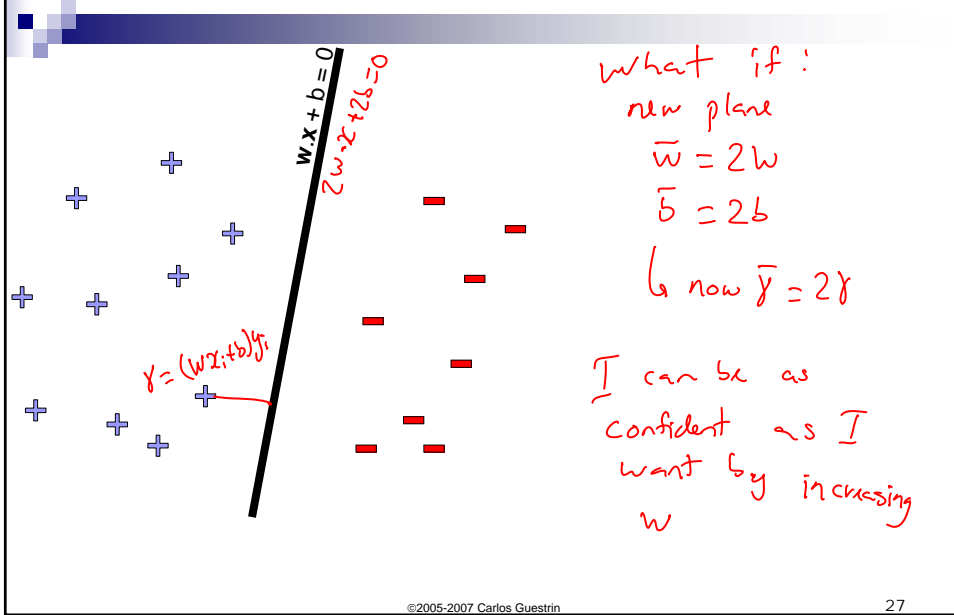
# Pick the one with the largest margin!



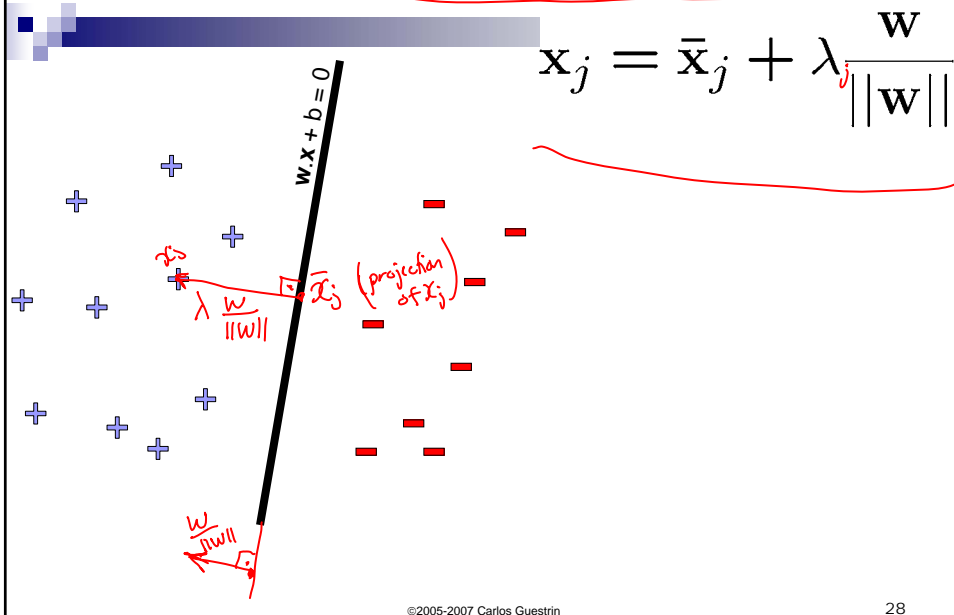
# Maximize the margin

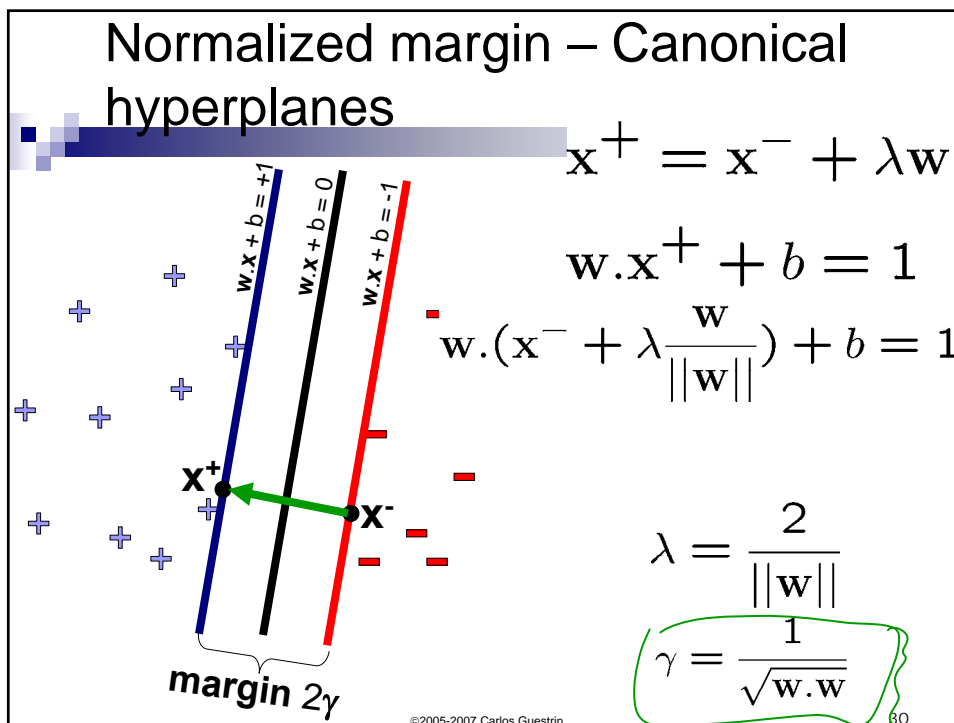
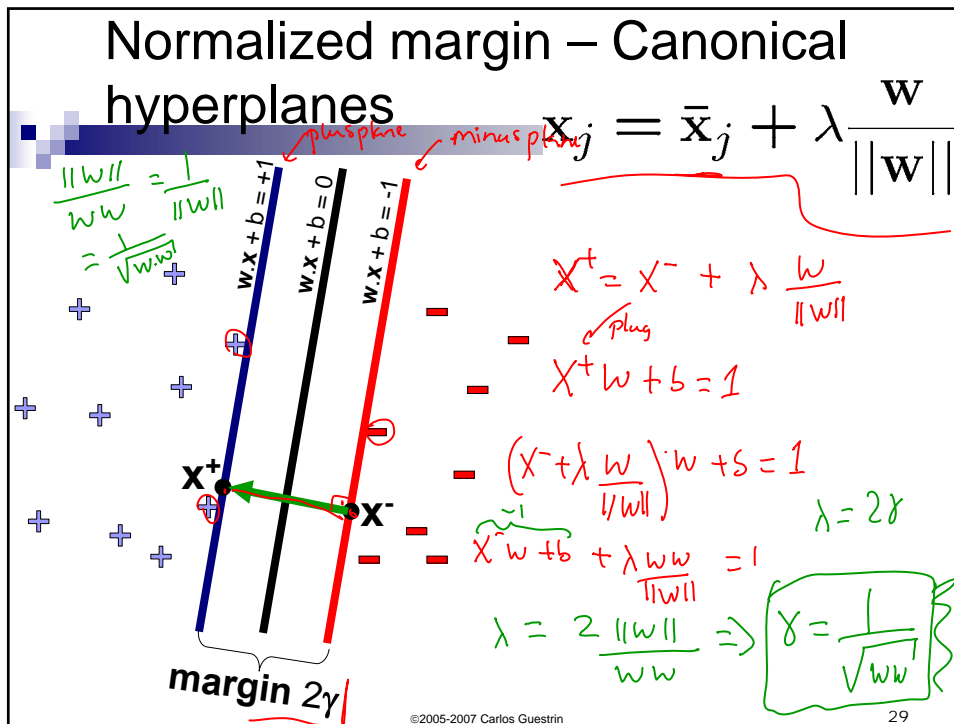


# But there are a many planes...



# Review: Normal to a plane





# Margin maximization using canonical hyperplanes

$$\gamma = \frac{1}{\sqrt{w \cdot w}}$$

$\text{maximize}_{\gamma, w, b} \gamma, \gamma \leq 1$   
 $(w \cdot x_j + b) y_j \geq \gamma, \forall j \in \text{Dataset}$

$\text{max } \frac{1}{\sqrt{w \cdot w}}$   
 $(w \cdot x_j + b) y_j \geq \gamma, \gamma \leq 1$   
 $\equiv \text{min } w \cdot w$   
 $(w \cdot x_j + b) y_j \geq \gamma; \gamma \leq 1$

$\text{minimize}_{w, b} w \cdot w$   
 $(w \cdot x_j + b) y_j \geq 1, \forall j \in \text{Dataset}$

*Handwritten notes:*  
 - maximum achieved when  $\gamma = 1$   
 - is a monotonic function  
 - margin on confidence of best for all points  
 - SUM

©2005-2007 Carlos Guestrin

31

# Support vector machines (SVMs)

$\text{minimize}_{w, b} w \cdot w$   
 $(w \cdot x_j + b) y_j \geq 1, \forall j$

- Solve efficiently by quadratic programming (QP)
  - Well-studied solution algorithms
- Hyperplane defined by support vectors

*Handwritten notes:*  
 - x no change  
 - no change  
 - change!!  
 - move here  
 - solution doesn't change  
 - points where  $(w \cdot x_j + b) y_j = 1$  (support vectors)

©2005-2007 Carlos Guestrin

32