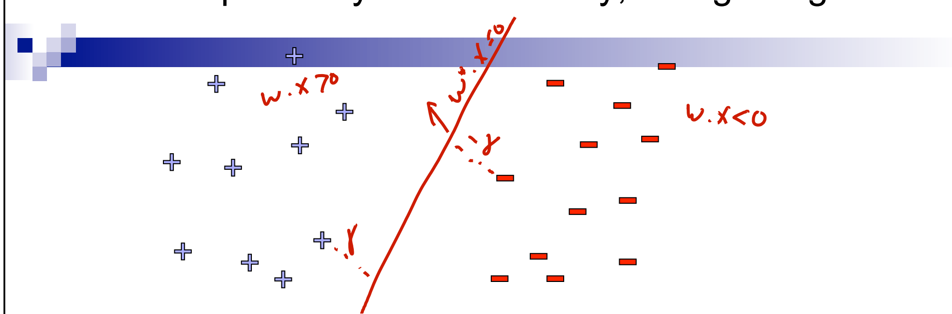
Kernels

Machine Learning – CSE446
Carlos Guestrin
University of Washington

October 28, 2013

Linear Separability: More formally, Using Margin



- Data linearly separable, if there exists
  - a vector $\exists\, w^*,\ \|w^*\| = 1$
  - a margin $\gamma > 0$
- Such that all points are at least γ away from $w.x = 0$

$\forall t$: if $y^{(t)} = +1,\ w^*.x^{(t)} \geq \gamma$; Linearly separable:
$y^{(t)} = -1,\ w^*.x^{(t)} \leq -\gamma$; $y^{(t)}\, w^*.x^{(t)} \geq \gamma$

Perceptron Analysis: Linearly Separable Case

- Theorem [Block, Novikoff]:
  - Given a sequence of labeled examples: $(x^{(1)}, y^{(1)}) \ldots (x^{(T)}, y^{(T)})$ — not iid
  - Each feature vector has bounded norm: $\forall t \; \|x^{(t)}\| \leq R$
  - If dataset is linearly separable:

$$\exists w^*, \|w^*\| = 1, \; \forall t \quad y^{(t)} w^* \cdot x^{(t)} \geq \gamma \quad \text{for } \gamma > 0$$

- Then the number of mistakes made by the online perceptron on any such sequence is bounded by

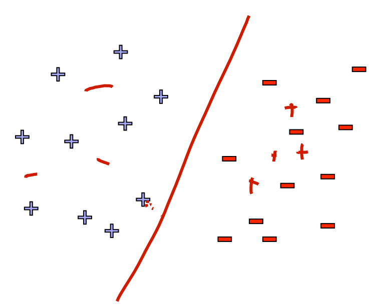
$$\left(\frac{R}{\gamma}\right)^2$$

Wow!! :)

constant, doesn't grow with T !! :)

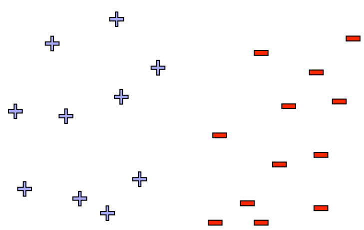
Beyond Linearly Separable Case

- Perceptron algorithm is super cool!
  - No assumption about data distribution!
    - Could be generated by an oblivious adversary, no need to be iid
  - Makes a fixed number of mistakes, and it's done for ever!
    - Even if you see infinite data
- However, real world not linearly separable
  - Can't expect never to make mistakes again
  - Analysis extends to non-linearly separable case
  - Very similar bound, see Freund & Schapire
  - Converges, but ultimately may not give good accuracy (make many many many mistakes)



& if data is very very non-linearly separable

What if the data is not linearly separable?



Use features of features of features of features….

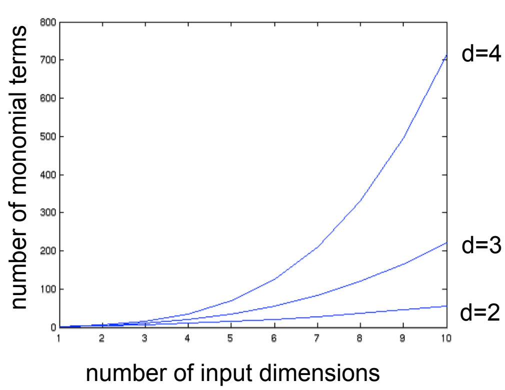
$$\Phi(\mathbf{x}) : R^m \mapsto F$$

Feature space can get really large really quickly!

Higher order polynomials

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

m – input features
d – degree of polynomial



grows fast!
d = 6, m = 100
about 1.6 billion terms

Perceptron Revisited

- Given weight vector $w^{(t)}$, predict point **x** by:
- Mistake at time *t*: $w^{(t+1)} \leftarrow w^{(t)} + y^{(t)} x^{(t)}$
- Thus, write weight vector in terms of mistaken data points only:
  - Let $M^{(t)}$ be time steps up to *t* when mistakes were made:
- Prediction rule now:
- When using high dimensional features:

Dot-product of polynomials

$\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = \text{polynomials of degree exactly } d$

Finally the Kernel Trick!!! (Kernelized Perceptron

- Every time you make a mistake, remember $(x^{(t)},y^{(t)})$

- Kernelized Perceptron prediction for **x**:

$$sign(\mathbf{w}^{(t)} \cdot \phi(\mathbf{x})) = \sum_{j \in M^{(t)}} y^{(j)} \phi(\mathbf{x}^{(j)}) \cdot \phi(\mathbf{x})$$

$$= \sum_{j \in M^{(t)}} y^{(j)} k(\mathbf{x}^{(j)}, \mathbf{x})$$

Polynomial kernels

- All monomials of degree d in O(d) operations:

$$\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = (\mathbf{u} \cdot \mathbf{v})^d = \text{polynomials of degree exactly d}$$

- How about all monomials of degree up to d?
  - Solution 0:
  - Better solution:

Common kernels

- Polynomials of degree exactly d

$$K(\mathbf{u},\mathbf{v}) = (\mathbf{u}\cdot\mathbf{v})^d$$

- Polynomials of degree up to d

$$K(\mathbf{u},\mathbf{v}) = (\mathbf{u}\cdot\mathbf{v}+1)^d$$

- Gaussian (squared exponential) kernel

$$K(\mathbf{u},\mathbf{v}) = \exp\left(-\frac{||\mathbf{u}-\mathbf{v}||^2}{2\sigma^2}\right)$$

- Sigmoid

$$K(\mathbf{u},\mathbf{v}) = \tanh(\eta\mathbf{u}\cdot\mathbf{v}+\nu)$$

What you need to know

- Notion of online learning
- Perceptron algorithm
- Mistake bounds and proofs
- The kernel trick
- Kernelized Perceptron
- Derive polynomial kernel
- Common kernels
- In online learning, report averaged weights at the end

Your Midterm…

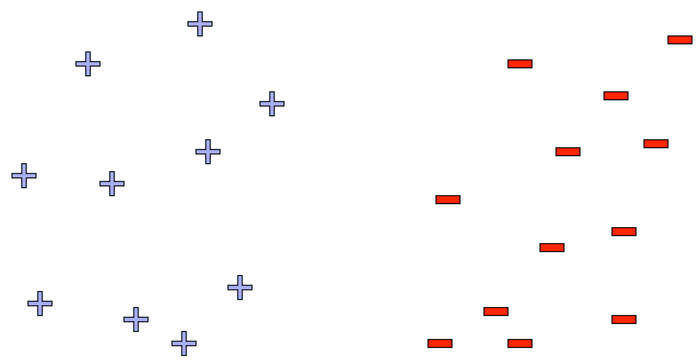
- Content: Everything up to last Wednesday (Perceptron)…
- Only 80mins, so arrive early and settle down quickly, we'll start and end on time
- "Open book"
  - Textbook, Books, Course notes, Personal notes
- Bring a calculator that can do log ☺
- No:
  - Computers, tablets, phones, other materials, internet devices, wireless telepathy or wandering eyes…
- The exam:
  - Covers key concepts and ideas, work on understanding the big picture, and differences between methods

Support Vector Machines

Machine Learning – CSE446
Carlos Guestrin
University of Washington

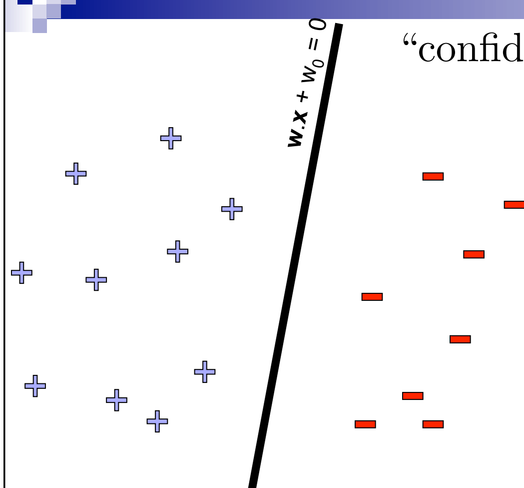
October 28, 2013

Linear classifiers – Which line is better?

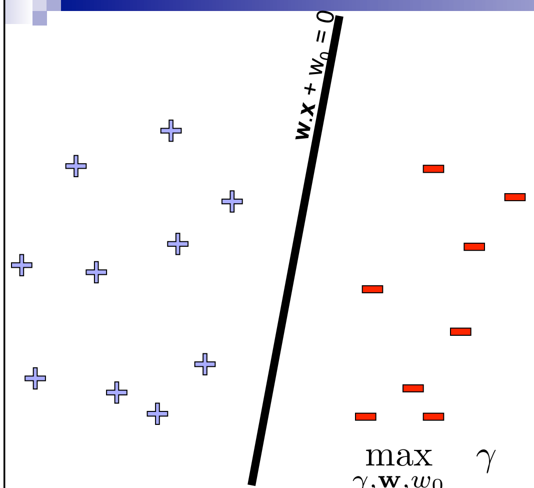


Pick the one with the largest margin!

$$\text{“confidence”} = y^j(\mathbf{w} \cdot \mathbf{x}^j + w_0)$$



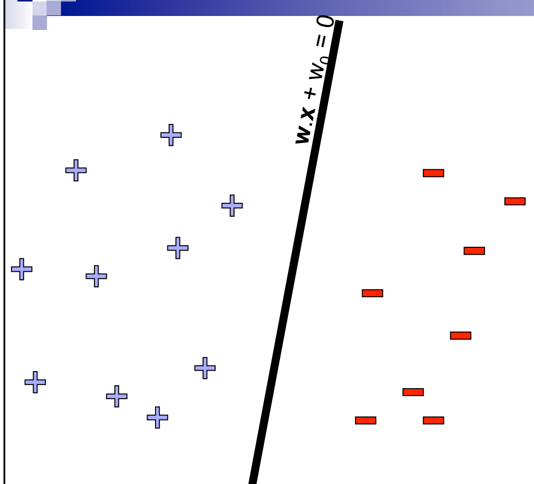
Maximize the margin



$$\max_{\gamma,\mathbf{w},w_0} \gamma$$

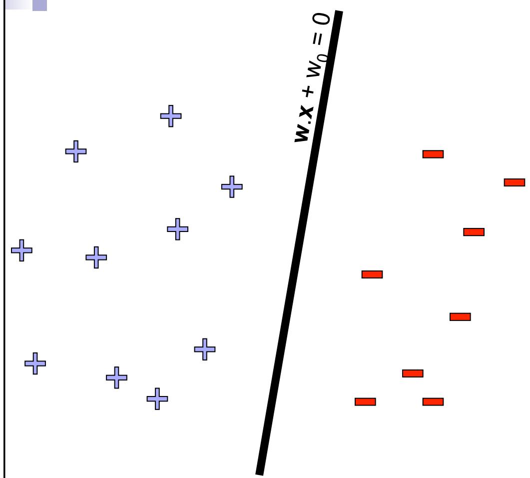
$$y^j(\mathbf{w} \cdot \mathbf{x}^j + w_0) \geq \gamma, \forall j \in \{1, \ldots, N\}$$

But there are many planes…



Review: Normal to a plane

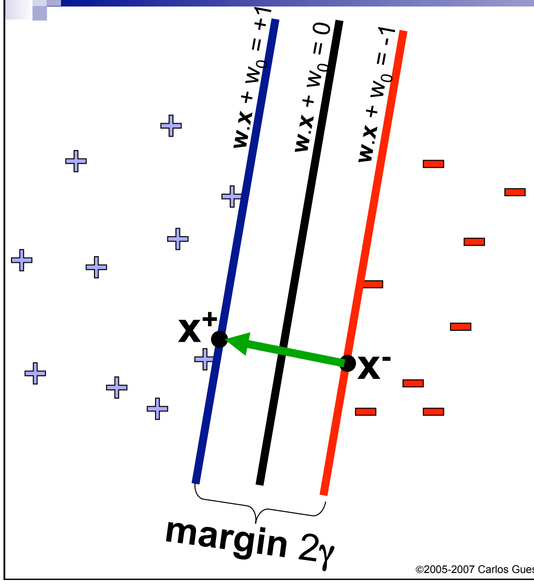
$$\mathbf{x}^j = \bar{\mathbf{x}}^j + \alpha \frac{\mathbf{w}}{||\mathbf{w}||}$$



 19

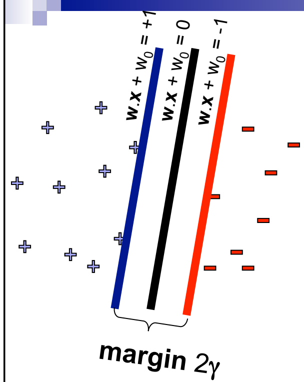
A Convention: Normalized margin – Canonical hyperplanes

$$\mathbf{x}^j = \bar{\mathbf{x}}^j + \alpha \frac{\mathbf{w}}{||\mathbf{w}||}$$



 20

Margin maximization using canonical hyperplanes



Unnormalized problem:

$$\max_{\gamma,\mathbf{w},w_0} \quad \gamma$$

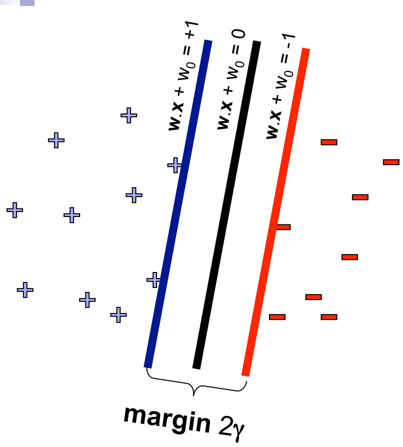
$$y^j(\mathbf{w}\cdot\mathbf{x}^j + w_0) \geq \gamma, \forall j \in \{1,\ldots,N\}$$

Normalized Problem:

$$\min_{\mathbf{w},w_0} \quad ||w||_2^2$$

$$y^j(\mathbf{w}\cdot\mathbf{x}^j + w_0) \geq 1, \forall j \in \{1,\ldots,N\}$$

Support vector machines (SVMs)



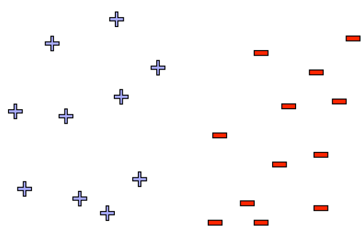
$$\min_{\mathbf{w},w_0} \quad ||w||_2^2$$

$$y^j(\mathbf{w}\cdot\mathbf{x}^j + w_0) \geq 1, \forall j \in \{1,\ldots,N\}$$

- Solve efficiently by many methods, e.g.,
  - quadratic programming (QP)
    - Well-studied solution algorithms
  - Stochastic gradient descent
- Hyperplane defined by support vectors

What if the data is not linearly separable?

Use features of features of features of features….

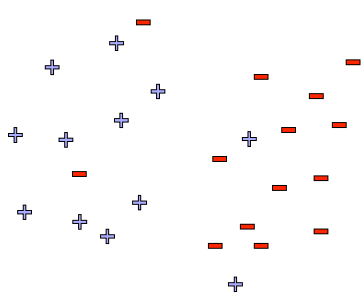


 23

What if the data is still not linearly separable?

$$\min_{\mathbf{w},w_0} \quad ||w||_2^2$$

$$y^j(\mathbf{w}\cdot\mathbf{x}^j + w_0) \geq 1 \qquad ,\forall j$$



- If data is not linearly separable, some points don't satisfy margin constraint:
- How bad is the violation?
- Tradeoff margin violation with **||w||**:

 24

SVMs for Non-Linearly Separable meet my friend the Perceptron…

- Perceptron was minimizing the hinge loss:

$$\sum_{j=1}^{N} \left(-y^j(\mathbf{w} \cdot \mathbf{x}^j + w_0)\right)_+$$

- SVMs minimizes the regularized hinge loss!!

$$||\mathbf{w}||_2^2 + C\sum_{j=1}^{N} \left(1 - y^j(\mathbf{w} \cdot \mathbf{x}^j + w_0)\right)_+$$

Stochastic Gradient Descent for SVMs

- Perceptron minimization:

$$\sum_{j=1}^{N} \left(-y^j(\mathbf{w} \cdot \mathbf{x}^j + w_0)\right)_+$$

- SGD for Perceptron:

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + \mathbb{1}\left[y^{(t)}(\mathbf{w}^{(t)} \cdot \mathbf{x}^{(t)}) \leq 0\right] y^{(t)}\mathbf{x}^{(t)}$$

- SVMs minimization:

$$||\mathbf{w}||_2^2 + C\sum_{j=1}^{N} \left(1 - y^j(\mathbf{w} \cdot \mathbf{x}^j + w_0)\right)_+$$

- SGD for SVMs:

What you need to know

- Maximizing margin
- Derivation of SVM formulation
- Non-linearly separable case
  - Hinge loss
  - A.K.A. adding slack variables
- SVMs = Perceptron + L2 regularization
- Can also use kernels with SVMs
- Can optimize SVMs with SGD
  - Many other approaches possible