# Kernels and Support Vector Machines

Machine Learning – CSE446
Sham Kakade
University of Washington

November 1, 2016



## Announcements:

- Project Milestones coming up
- HW2
  - You've implemented GD, SGD, etc…
- HW3 posted this week.
  - Let's get state of the art on MNIST!
  - It'll be collaborative
- Today:
  - Review: the perceptron, margins, and separability
  - Kernels & SVMs

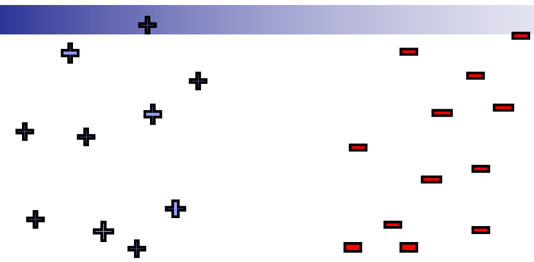
## Support Vector Machines (Two Ideas Mixed up)

- 1) An attempt to better optimize the classification loss?
  - Questionable?
  - Latent SVMs are interesting.
- 2) Kernels
  - Warp the feature space
  - This idea is actually more general

- The success of SVMS?

## Linear Separability: More formally, Using Margin



- Data linearly separable, if there exists
  - a vector
  - a margin
- Such that

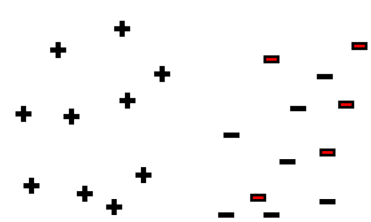
## Perceptron Analysis: Linearly Separable Case

- Theorem [Block, Novikoff]:
  - Given a sequence of labeled examples:
  - Each feature vector has bounded norm:
  - If dataset is linearly separable:
- Then the number of mistakes made by the online perceptron on any such sequence is bounded by

## 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



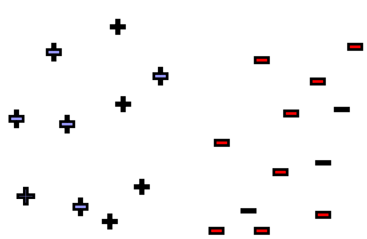
# Kernels

Machine Learning – CSE446
Sham Kakade
University of Washington

November 1, 2016




## What if the data is not linearly separable?



**Use features of features of features of features….**

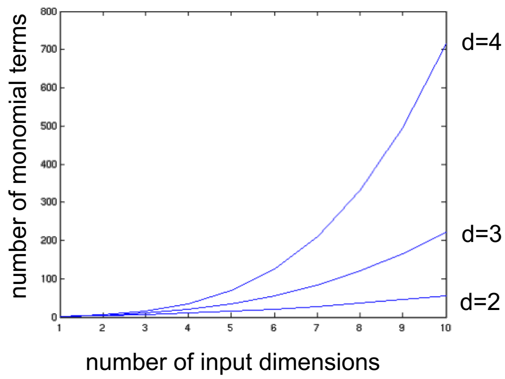
$$\Phi(\mathbf{x}) : R^m \mapsto F$$

**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

$\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) =$ 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

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

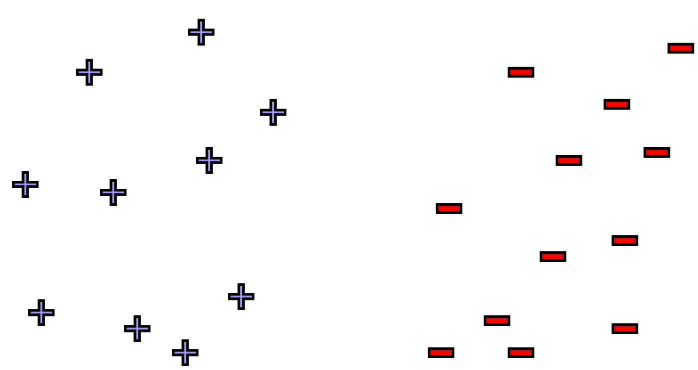
# Support Vector Machines

Machine Learning – CSE446
Sham Kakade
University of Washington

November 1, 2016

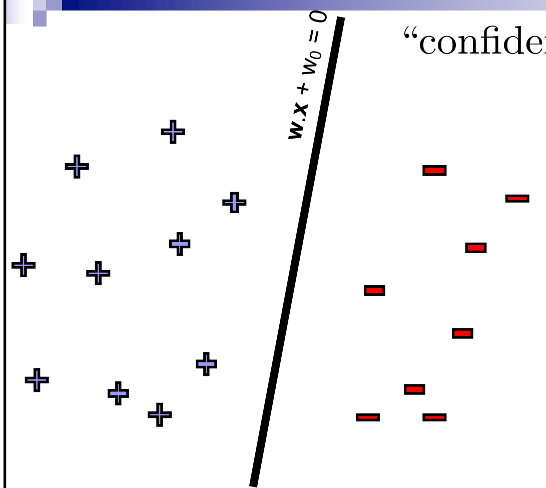



## Linear classifiers – Which line is better?



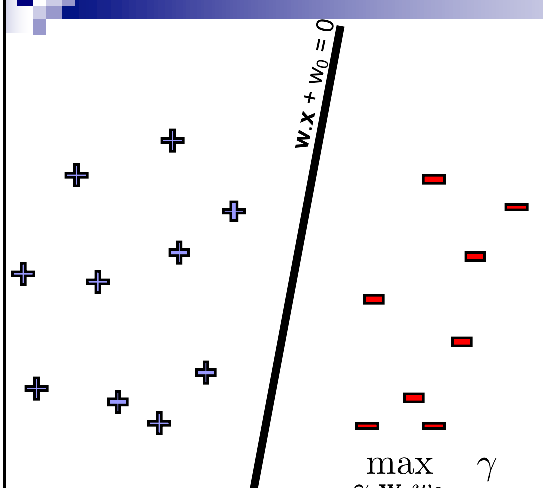
# Pick the one with the largest margin!

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





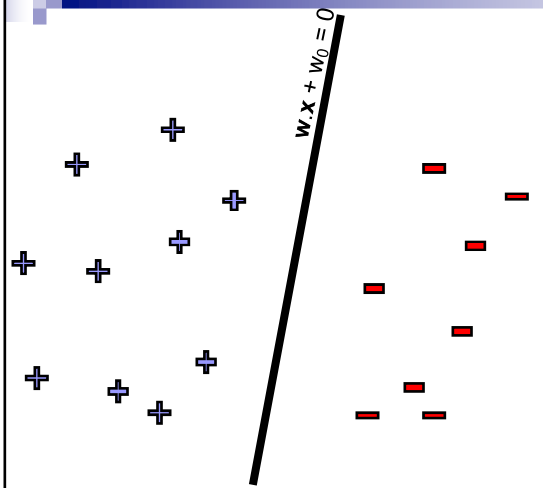

# Maximize the margin




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

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

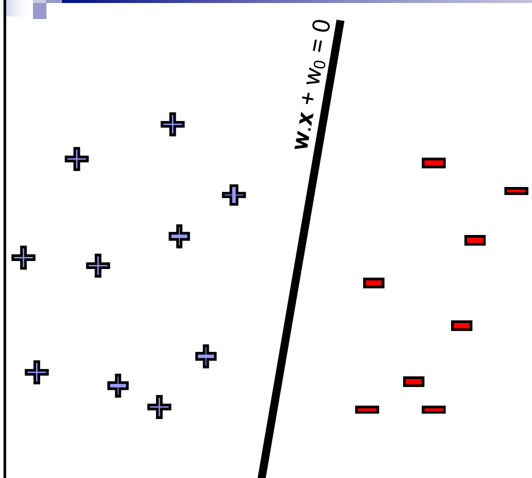
## But there are many planes…




©2016 Sham Kakade 19

## *Review*: Normal to a plane

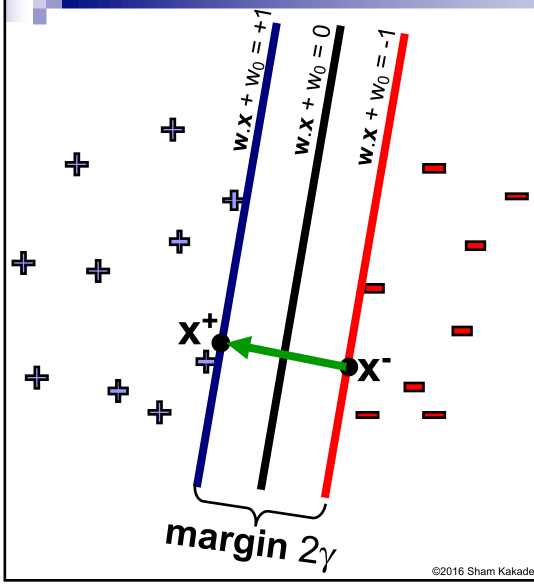
$$\mathbf{x}^j = \bar{\mathbf{x}}^j + \alpha \frac{\mathbf{w}}{||\mathbf{w}||}$$




©2016 Sham Kakade 20

# A Convention: Normalized margin – Canonical hyperplanes

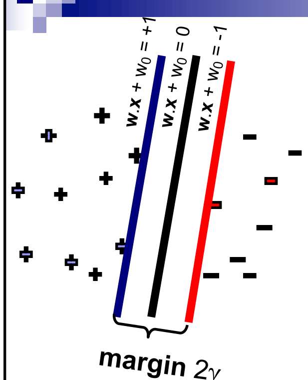
$$\mathbf{x}^j = \bar{\mathbf{x}}^j + \alpha \frac{\mathbf{w}}{||\mathbf{w}||}$$








# Margin maximization using canonical hyperplanes



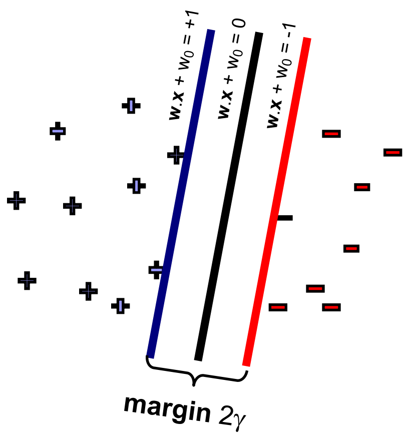

Unnormalized problem:

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

**Normalized Problem:**

$$\min_{\mathbf{w},w_0} ||w||_2^2$$
$$y^j(\mathbf{w}\cdot\mathbf{x}^j + w_0) \geq 1, \forall j \in \{1,\ldots,N\}$$

# Support vector machines (SVMs)



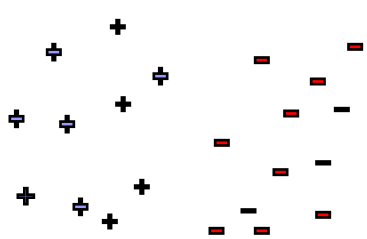

$$\min_{\mathbf{w},w_0} \quad ||w||_2^2$$

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

- 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?

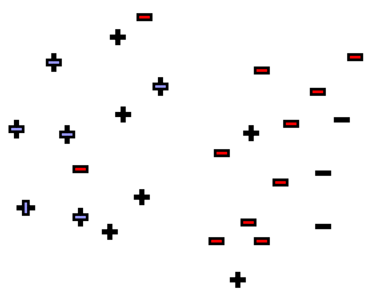


**Use features of features of features of features….**

# 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||**:



# 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:



## SVMs vs logistic regression

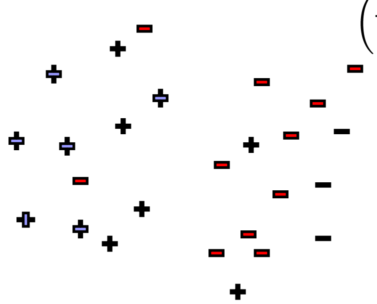
- We often want probabilities/confidences (logistic wins here)
- For classification loss, they are comparable
- Multiclass setting:
  - Softmax naturally generalizes logistic regression
  - SVMs have
- What about good old least squares?

# Multiple Classes

- One can generalize the hinge loss
  - If no error (by some margin) -> no loss
  - If error, penalize what you said against the best
- SVMs vs logistic regression
  - We often want probabilities/confidences (logistic wins here)
  - For classification loss, they are
- Latent SVMs
  - When you have many classes it's difficult to do logistic regression
- 2) Kernels
  - Warp the feature space

# Slack variables – Hinge loss

$$\text{minimize}_{\mathbf{w},b} \quad \mathbf{w}.\mathbf{w}$$
$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 \qquad ,\forall j$$



- If margin ¸ 1, don't care
- If margin < 1, pay linear penalty



# *Side note*: What's the difference between SVMs and logistic regression?

**SVM:**

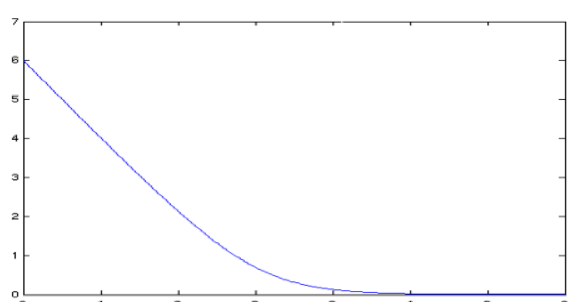
$$\text{minimize}_{\mathbf{w},b} \quad \mathbf{w}.\mathbf{w} + C \sum_j \xi_j$$
$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j, \ \forall j$$
$$\xi_j \geq 0, \ \forall j$$

**Logistic regression:**

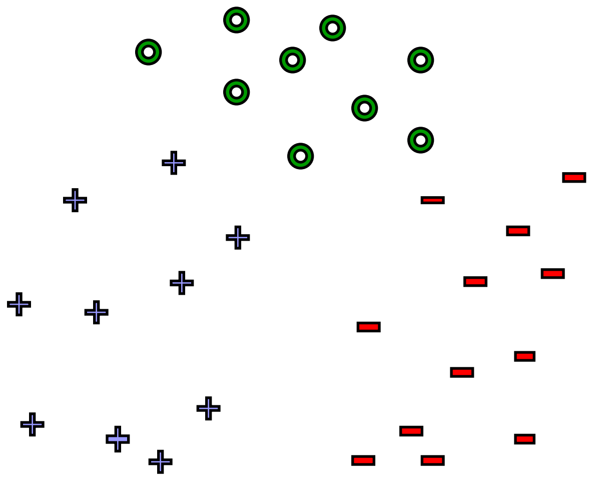
$$P(Y = 1 \mid x, \mathbf{w}) = \frac{1}{1 + e^{-(\mathbf{w}.\mathbf{x}+b)}}$$

**Log loss:**

$$-\ln P(Y = 1 \mid x, \mathbf{w}) = \ln\left(1 + e^{-(\mathbf{w}.\mathbf{x}+b)}\right)$$

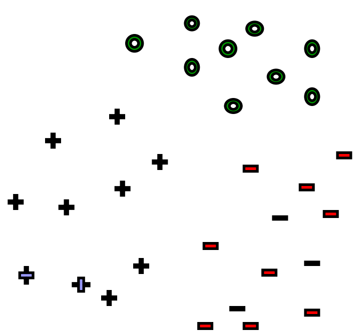


# What about multiple classes?



# One against All

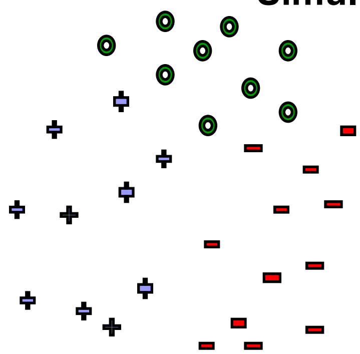


**Learn 3 classifiers:**

## Learn 1 classifier: Multiclass SVM

**Simultaneously learn 3 sets of weights**



$$\mathbf{w}^{(y_j)}.\mathbf{x}_j + b^{(y_j)} \geq \mathbf{w}^{(y')}.\mathbf{x}_j + b^{(y')} + 1, \ \forall y' \neq y_j, \ \forall j$$




## Learn 1 classifier: Multiclass SVM

$$\text{minimize}_{\mathbf{w},b} \quad \sum_y \mathbf{w}^{(y)}.\mathbf{w}^{(y)} + C\sum_j \xi_j$$
$$\mathbf{w}^{(y_j)}.\mathbf{x}_j + b^{(y_j)} \geq \mathbf{w}^{(y')}.\mathbf{x}_j + b^{(y')} + 1 - \xi_j, \ \forall y' \neq y_j, \ \forall j$$
$$\xi_j \geq 0, \ \forall j$$