Module 5: Classification

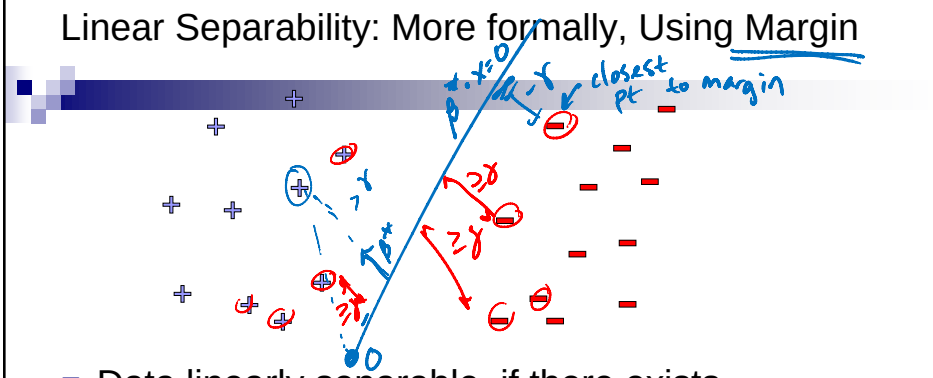
Kernelized Perceptron: Kernels, again!

STAT/BIOSTAT 527, University of Washington
Emily Fox

May 29th, 2014

©Emily Fox 2014

Linear Separability: More formally, Using Margin



- Data linearly separable, if there exists
  - a vector $\exists\, \beta^*$ $\|\beta^*\| = 1$
  - a margin $\gamma > 0$
- Such that all pts are γ far away or more from $\beta^* \cdot x = 0$

$\forall t$ if $y_t = +1$ $\beta^* \cdot x_t > \gamma$
$y_t = -1$ $\beta^* \cdot x_t < -\gamma$ $\Rightarrow$ $\boxed{y_t(\beta^* \cdot x_t) > \gamma}$

©Emily Fox 2014

Perceptron Analysis: Linearly Separable Case

- Theorem [Block, Novikoff]:
  - Given a sequence of labeled examples: $(x_1, y_1), \dots, (x_n, y_n)$

    examples need not be iid nor random
  - Each covariate vector has bounded norm:

    $\forall t \quad \|x_t\| \le R$
  - If dataset is linearly separable:

    $\exists \beta^* \quad \|\beta^*\| = 1 \quad$ s.t. $\quad y_t(\beta^* \cdot x_t) \ge \gamma \quad \forall t \quad$ for some $\gamma > 0$
- Then the number of mistakes made by the online perceptron on this sequence is bounded by

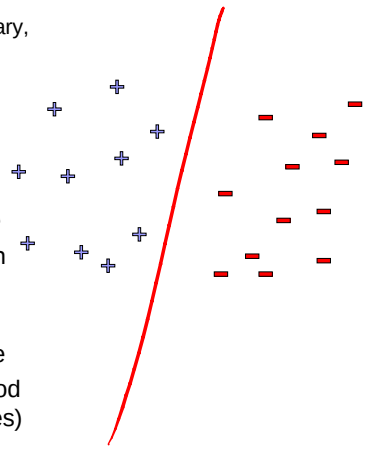
  $\left(\frac{R}{\gamma}\right)^2$ crazy!

  constant... doesn't depend on T or dim x

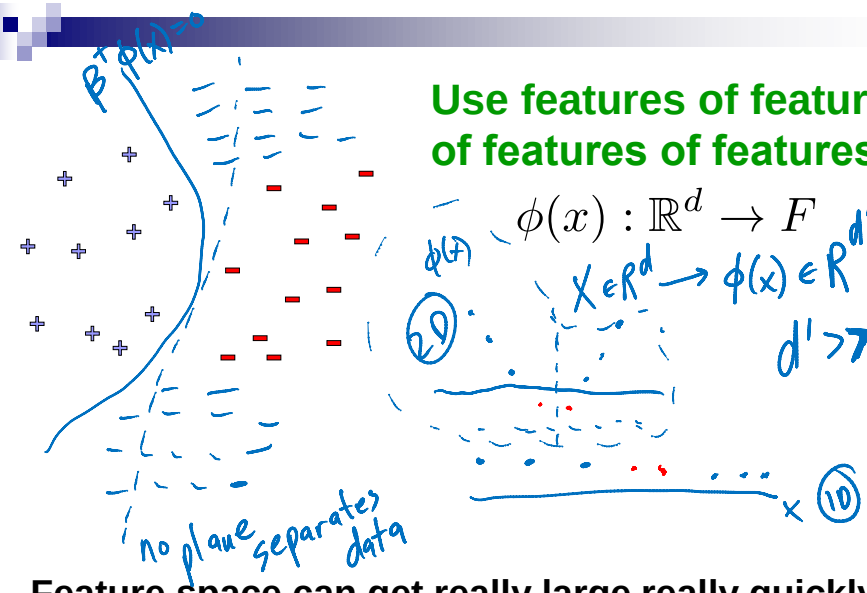
  w.r.t. time

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)



What if the data are not linearly separable?



Use features of features of features of features….

$$\phi(x) : \mathbb{R}^d \rightarrow F$$

$x \in \mathbb{R}^d \rightarrow \phi(x) \in \mathbb{R}^{d'}$

$d' >> d$

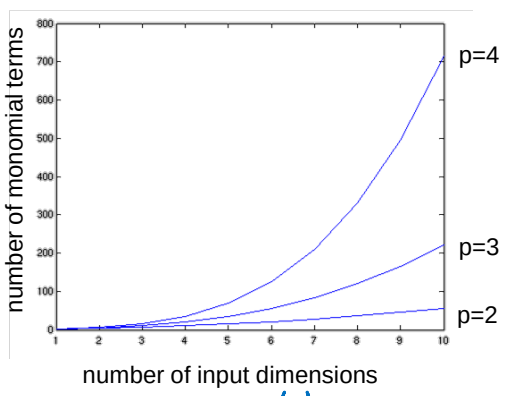
Feature space can get really large really quickly!

©Emily Fox 2014

Higher Order Polynomials (basis expansion)

$x \rightarrow \phi(x)$

$$\# \text{ terms} = \binom{p+d-1}{p} = \frac{(p+d-1)!}{p!(d-1)!}$$



d – covariate dimension
p – degree of polynomial

even though dims of $\phi(x)$ are huge, we can fit the model very quickly.

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

©Emily Fox 2014

Perceptron Revisited

- Given weight vector $\beta^{(t)}$, predict point $\mathbf{x}$ by:

$$\hat{y} = \text{sign}(\beta^{(t)} \cdot x)$$

- Mistake at time *t*: $\beta^{(t+1)} \leftarrow \beta^{(t)} + y_t x_t$
- Thus, write weight vector in terms of mistaken data points only:
  - Let $M^{(t)}$ be time steps up to *t* when mistakes were made: $(\beta^{(0)} = 0)$

$$\beta^{(t)} = \sum_{j \in M_{(t)}} y_j x_j$$

- Prediction rule now:

$$\hat{y} = \text{sign}(\beta^{(t)} \cdot x) = \text{sign}\left(\left(\sum_{j \in M_t} y_j x_j\right) \cdot x\right) = \text{sign}\left(\sum_{j \in M_t} y_j \underbrace{(x_j \cdot x)}\right)$$

- When using high dimensional features:

$$\hat{y} = \text{sign}\left(\sum_{j \in M_t} y_j \left(\underbrace{\phi(x_j) \cdot \phi(x)}\right)\right)$$

when can I compute efficiently?? classification only depends on inner product of "x"

Dot-Product of Polynomials

$\phi_{poly}(u) \cdot \phi_{poly}(v) =$ polynomials of degree exactly *p*

$p=1$: $\phi(u) \cdot \phi(v) = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = u_1 v_1 + u_2 v_2 = (u \cdot v)^1$

$p=2$: $\phi(u) \cdot \phi(v) = \begin{pmatrix} u_1^2 \\ u_1 u_2 \\ u_2 u_1 \\ u_2^2 \end{pmatrix} \cdot \begin{pmatrix} v_1^2 \\ v_1 v_2 \\ v_2 v_1 \\ v_2^2 \end{pmatrix} = u_1^2 v_1^2 + 2 u_1 u_2 v_1 v_2 + v_2^2 u_2^2$

$= (u_1 v_1 + u_2 v_2)^2$

$= (u \cdot v)^2$

By induction,

degree p: $\phi(u) \cdot \phi(v) = (u \cdot v)^p$

"kernel trick" compute efficiently!

The Kernel Trick Again: Kernelized Perceptron

- Every time you make a mistake, remember (x_t, y_t)

keep track of indices $M(t)$ where we make mistakes up to "t".

- Kernelized perceptron prediction for **x**:

$$\text{sign}(\beta^{(t)} \cdot \phi(x)) = \sum_{i \in M^{(t)}} y_i (\phi(x_i) \cdot \phi(x))$$

$(x_i \cdot x)$

$$= \sum_{i \in M^{(t)}} y_i \boxed{k(x_i, x)}$$

kernel fcn.

one choice

$(x_i \cdot x)^p$ $\quad k(u,v) = \phi(u) \cdot \phi(v)$

Going Infinite…

Change of notation:

$$h_j(x) \to \phi_j(x)$$

- Nonparametric Gaussian regression: basis fcns "K"
  Would like to let the number of "features" $M \to \infty$

what kernel can represent a "$\phi \cdot \phi^T$" ??

- *Prior:* $p(\beta \mid 0, \alpha^{-1} I_M)$

Dist. on f

- ~~*Predictions*~~: $f = \Phi\beta$

$(f(x_1), \ldots, f(x_n))^T$ $\quad (\beta_1, \ldots, \beta_n)^T$

linear comb. of Gaussians β_j

$\Rightarrow f$ Gauss.

$E[f] = \phi E[\beta] = 0$

$\text{cov}(f) = \phi E[\beta\beta^T] \phi^T = \frac{1}{\alpha} \phi\phi^T$

$P(f) = N(f \mid 0, \alpha^{-1} \underbrace{\phi\phi^T}_{(n \times n)})$

$\underset{n \times 1}{\begin{bmatrix} \\ \\ \end{bmatrix}} = \underset{n \times M}{\begin{bmatrix} \\ \\ \end{bmatrix}} \underset{M \times 1}{\begin{bmatrix} \\ \\ \end{bmatrix}}$ $\quad [\phi_1(x_i) \ldots \phi_n(x_i)]$

- Gaussian process models replace explicit basis function representation with a direct specification in terms of a ***positive definite kernel function***

Mercer Kernel Functions

- ~~Predictions~~ Dist. are of the form

$M = 1000, 10000, \ldots$

$$p(f) = N(f \mid 0, \alpha^{-1}\Phi\Phi^T) = N(f \mid 0, K)$$

$n \times M \quad M \times n$ $\quad n \times n$ matrix

where the ***Gram matrix*** *K* is defined as

$$K_{ij} = k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

kernel fcn $\quad$ dim M

- *K* is a ***Mercer kernel*** if the Gram matrix is positive definite for any *n* and any $x_1, \ldots, x_n$

Note: K is $n \times n$ regardless of M (dim of basis/features)

Example of the "kernel trick"

Polynomial kernels

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

$\phi(u) \cdot \phi(v) = (u \cdot v)^p =$ polynomials of degree exactly *p*

- How about all monomials of degree up to *p*?
  - Solution 0: $\phi(u) \cdot \phi(v) = \sum_{i=0}^{p} \binom{p}{i} (u \cdot v)^i$
  - Better solution:

$d = 2$

$$(u \cdot v)^1 + (u \cdot v)^2 + (u \cdot v)^1 + (u \cdot v)^0 = (u \cdot v + 1)^2$$

for degree p: $\phi(u) \cdot \phi(v) = K(u,v) = (u \cdot v + 1)^p$

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

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

- Sigmoid

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

MANY MANY MORE!!

radial

has a $\phi(u)\cdot\phi(v)$ decomp.

$\phi(u) \in \mathbb{R}^\infty$

inner product on an ∞-dim. space

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

Module 5: Classification

≈ perceptron + regularization

Support Vector Machines

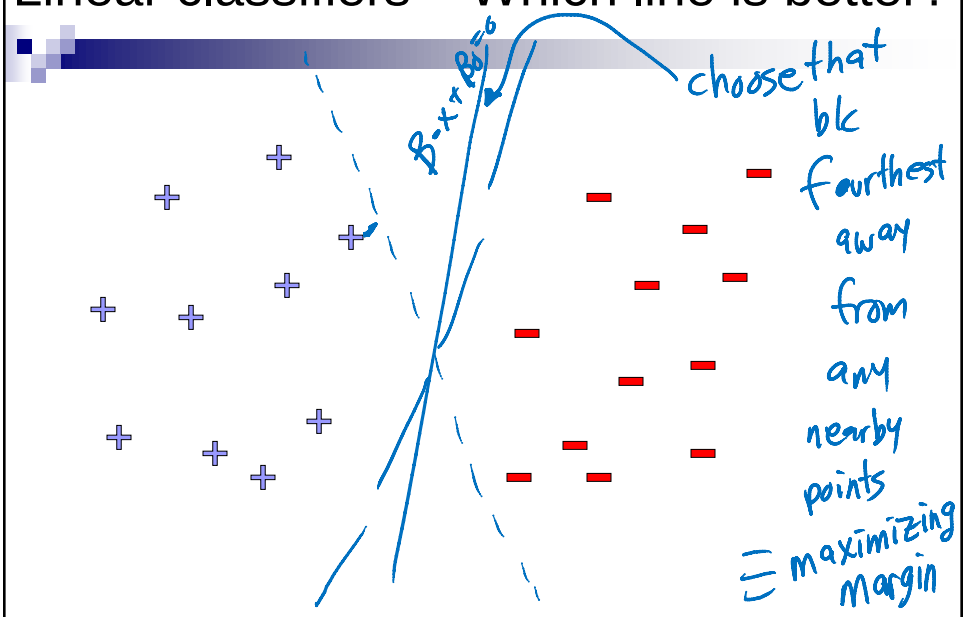
STAT/BIOSTAT 527, University of Washington

Emily Fox

May 29th, 2014

©Emily Fox 2014

Linear classifiers – Which line is better?

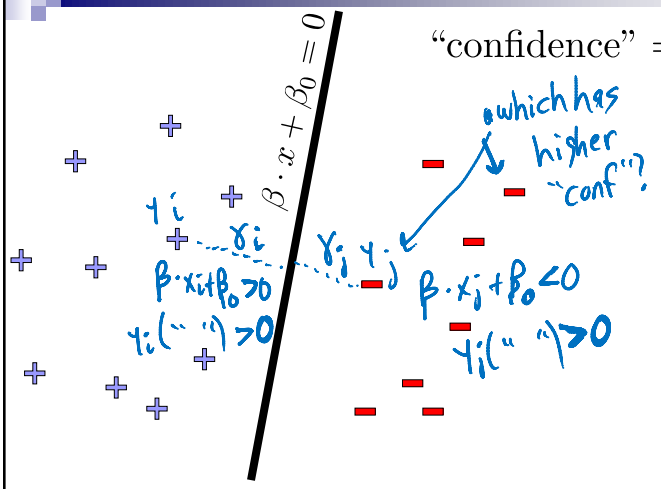


choose that b/c fourthest away from any nearby points

≡ maximizing margin

©Emily Fox 2014

Pick the one with the largest margin!



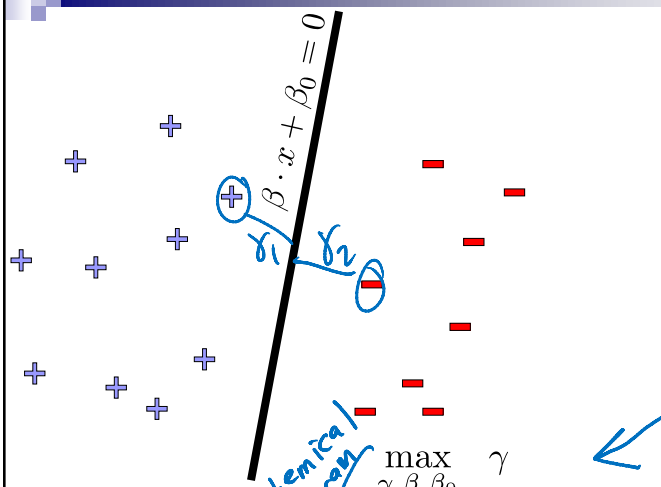
$$\text{"confidence"} = y_i(\beta \cdot x_i + \beta_0)$$

which has higher "conf"?

pos. if made no mistake

Maximize the margin

$\equiv$ maximize the "minimum" margin "γ"



$$\max_{\beta,\beta_0} \min_i \gamma_i$$

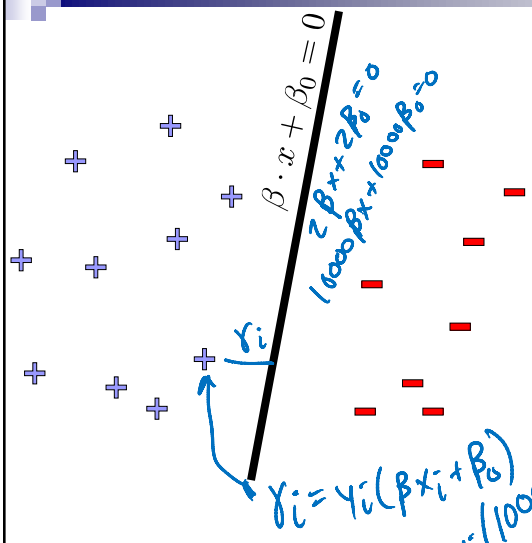
Find β, β_0 that maximizes the minimum "γ" margin.

mathematical program

$$\max_{\gamma,\beta,\beta_0} \quad \gamma$$

$$y_i(\beta \cdot x_i + \beta_0) \geq \gamma, \forall i \in \{1, \ldots, n\}$$

But there are many planes…



$\gamma_i = y_i(\beta x_i + \beta_0)$

$y_i(10000\beta x_i + 10000\beta_0) = 10000\gamma_i$

recall obj: max γ

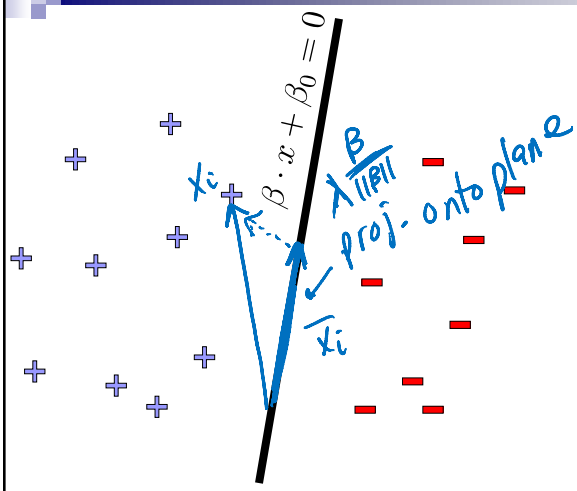
"γ" will increase in magnitude as we scale up "β", "β_0"

→ sol'n not well specified.

©Emily Fox 2014

Review: Normal to a plane

$$x_i = \bar{x}_i + \lambda \frac{\beta}{||\beta||}$$



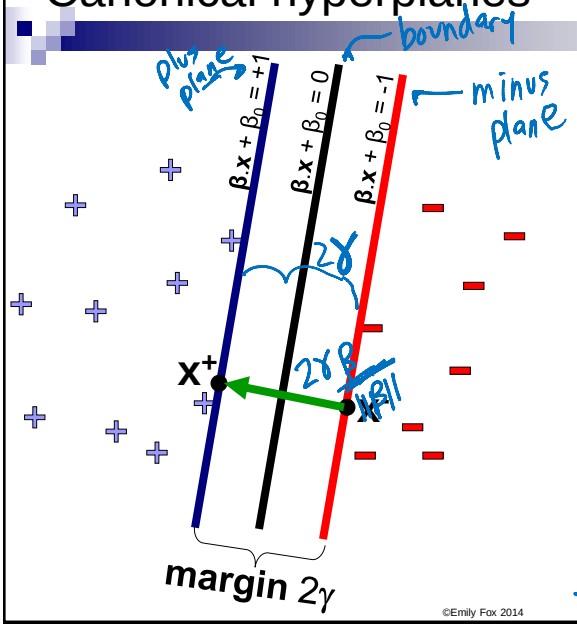
Projection

margin (or normal)

©Emily Fox 2014

A Convention: Normalized margin – Canonical hyperplanes

$$x_i = \bar{x}_i + \lambda \frac{\beta}{||\beta||}$$



pts. on plane

$\beta \cdot x^+ + \beta_0 = 1$

$\beta \cdot x^- + \beta_0 = -1$

$x^+ = x^- + 2\gamma \frac{\beta}{||\beta||}$

$\beta \cdot (x^- + 2\gamma \frac{\beta}{||\beta||}) + \beta_0 = +1$

$\underbrace{\beta \cdot x^- + \beta_0}_{-1} + 2\gamma \frac{\beta \cdot \beta}{||\beta||} = +1$

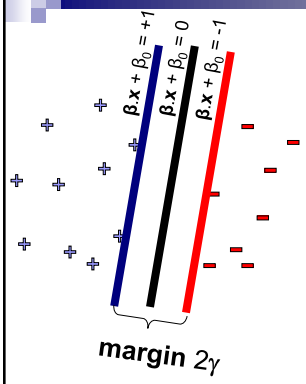
$\rightarrow -1 + 2\gamma ||\beta|| = 1$

$\gamma = \frac{1}{||\beta||}$

©Emily Fox 2014

Margin maximization using canonical hyperplanes

$\max_x \frac{1}{f(x)} \leftrightarrow \min_x f(x)$



Unnormalized problem: $\max_{\gamma,\beta,\beta_0} \gamma$ "min" margin

$$y_i(\beta \cdot x_i + \beta_0) \geq \gamma, \forall i \in \{1, \ldots, n\}$$

Normalized Problem:

if $y_i(\beta \cdot x_i + \beta_0) \geq 1$

"canonical" hyperplanes

$\max \gamma \leftrightarrow \max \frac{1}{||\beta||}$

$\leftrightarrow \min ||\beta||^2$

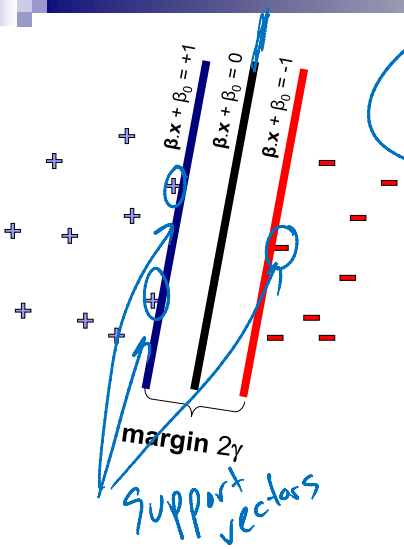
$$\min_{\beta,\beta_0} ||\beta||_2^2$$

$$y_i(\beta \cdot x_i + \beta_0) \geq 1, \forall i \in \{1, \ldots, n\}$$

standard SVM problem (derivatives are easier) (norm minimization)

©Emily Fox 2014

Support vector machines (SVMs)



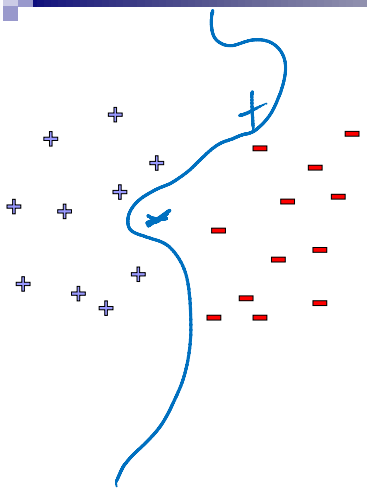
$$\min_{\beta,\beta_0} \quad ||\beta||_2^2$$

$$y_i(\beta \cdot x_i + \beta_0) \geq 1, \forall i \in \{1, \ldots, n\}$$

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

* only perturbations of these points will change the sol'n. (contrast w/ LDA, QDA)

What if the data are not linearly separable?



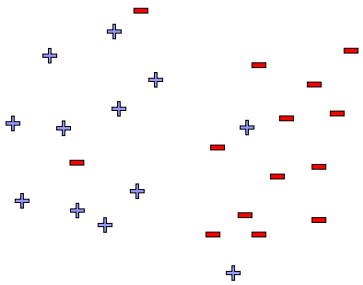
Use features of features of features of features….

SVM can be solved as usual BUT now we can apply a kernel (just as in perceptron)

What if the data are still not linearly separable?

$$\min_{\beta,\beta_0} \quad ||\beta||_2^2$$

$$y_i(\beta \cdot x_i + \beta_0) \geq 1, \forall i \in \{1, \ldots, n\}$$



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

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

- Perceptron was minimizing the hinge loss:

$$\sum_{i=1}^{n} (-y_i(\beta \cdot x_i + \beta_0))_+$$

- SVMs minimizes the regularized hinge loss!!

$$||\beta||_2^2 + C \sum_{i=1}^{n} (1 - y_i(\beta \cdot x_i + \beta_0))_+$$

Stochastic Gradient Descent for SVMs

- Perceptron minimization:

$$\sum_{i=1}^{n} (-y_i(\beta \cdot x_i + \beta_0))_+$$

- SGD for Perceptron:

$$\beta^{(t+1)} \leftarrow \beta^{(t)} + \mathbb{I}\left[y_t(\beta^{(t)} \cdot x_t) \leq 0\right] y_t x_t$$

- SVMs minimization:

$$||\beta||_2^2 + C \sum_{i=1}^{n} (1 - y_i(\beta \cdot x_i + \beta_0))_+$$

- SGD for SVMs:

©Emily Fox 2014

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

SVM:

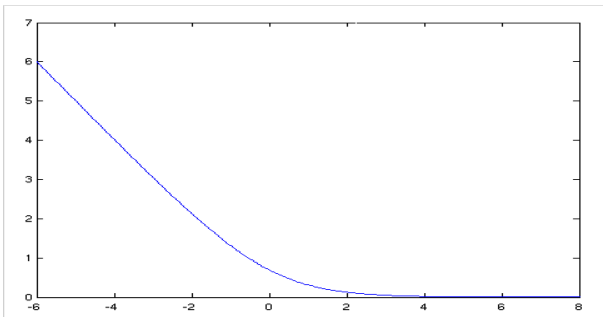
$$\min_{\beta,\beta_0} \; ||\beta||_2^2 + C \sum_{i=1}^{n} (1 - y_i(\beta \cdot x_i + \beta_0))_+$$

Logistic regression:

$$p(Y = 1 \mid x, \beta) = \frac{1}{1 + e^{-(\beta \cdot x + \beta_0)}}$$

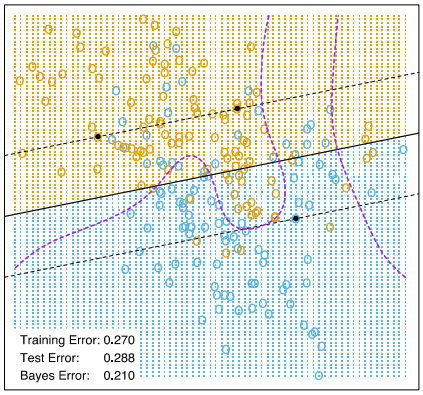
Log loss:

$$-\log p(Y = 1 \mid x, \beta) = \log\left(1 + e^{-(\beta \cdot x + \beta_0)}\right)$$

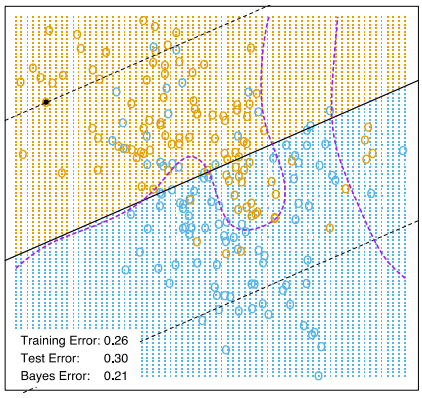


©Emily Fox 2014

Mixture Model Example



$C = 10000$



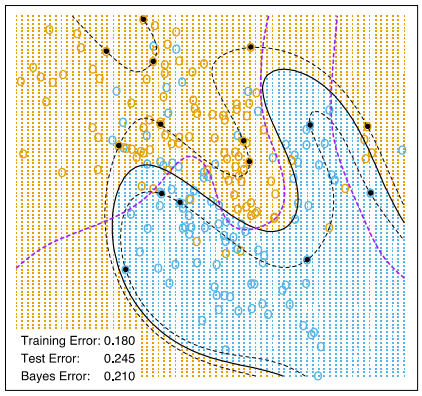
$C = 0.01$

From Hastie, Tibshirani, Friedman book

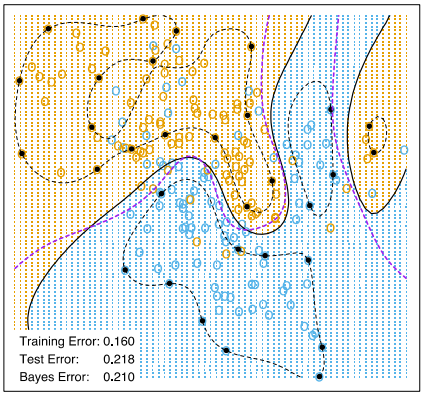
©Emily Fox 2014

Mixture Model Example - Kernels

SVM - Degree-4 Polynomial in Feature Space



SVM - Radial Kernel in Feature Space



From Hastie, Tibshirani, Friedman book

©Emily Fox 2014