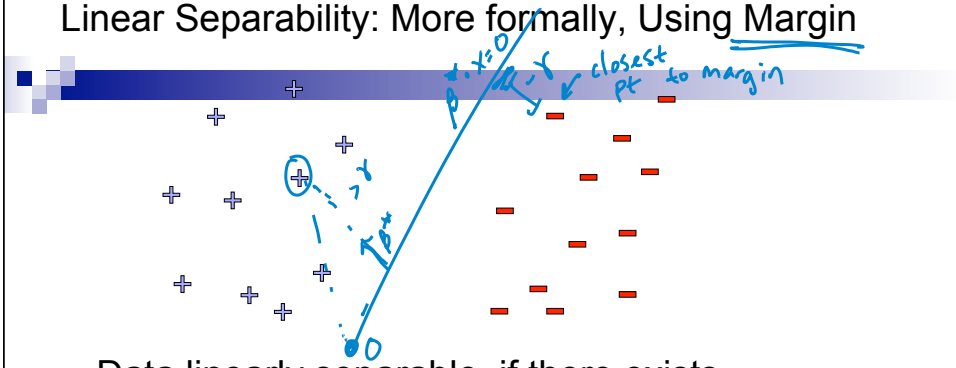
Module 5: Classification

Kernelized Perceptron: Kernels, again!

STAT/BIOSTAT 527, University of Washington
Emily Fox
June 4th, 2013

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 $y_t = +1$ $\beta^* \cdot x_t > \gamma$

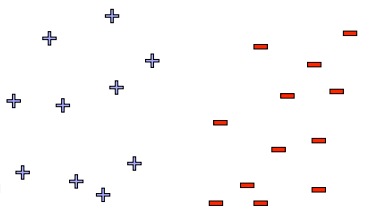
$y_t = -1$ $\beta^* \cdot x_t < -\gamma$ $\Rightarrow$ $\boxed{y_t(\beta^* \cdot x_t) > \gamma}$

Perceptron Analysis: Linearly Separable Case

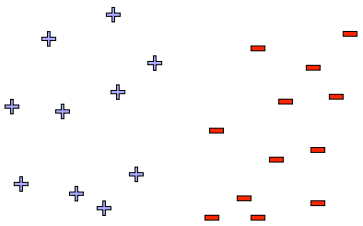
- Theorem [Block, Novikoff]:
  - Given a sequence of labeled examples: $(x_1, y_1), \ldots, (x_n, y_n)$
    examples need not be iid nor random
  - Each covariate vector has bounded norm:
    $\forall t \quad \|x_t\| \leq R$
  - If dataset is linearly separable:
    $\exists \beta^* \quad \|\beta^*\| = 1 \quad \text{s.t.} \quad y_t(\beta^* \cdot x_t) \geq \gamma \quad \forall t \quad \text{for some } \gamma > 0$
- 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

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….

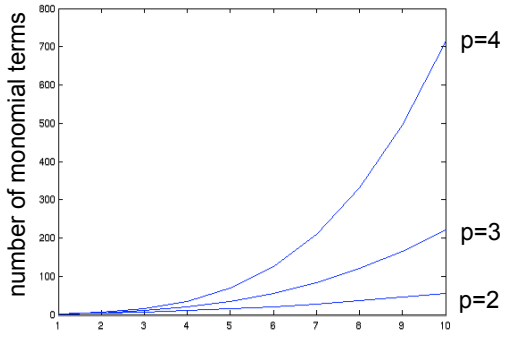
$$\phi(x) : \mathbb{R}^d \rightarrow F$$

Feature space can get really large really quickly!

Higher Order Polynomials

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

d – covariate dimension
p – degree of polynomial



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

Perceptron Revisited

- Given weight vector $\beta^{(t)}$, predict point **x** by:
- 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:
- Prediction rule now:
- When using high dimensional features:

Dot-Product of Polynomials

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

The Kernel Trick Again: Kernelized Perceptron

- Every time you make a mistake, remember (x_t, y_t)
- Kernelized perceptron prediction for **x**:

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

Going Infinite…

Change of notation: $h_j(x) \to \phi_j(x)$

- Nonparametric Gaussian regression: Would like to let the number of "features" *M* → ∞ (basis fcns)
- *Prior:* $p(\beta \mid 0, \alpha^{-1} I_M)$
- ~~*Predictions:*~~ $f = \Phi\beta$ (Dist. on f; $(f(x_1), \dots, f(x_n))^T$; $(\beta_1, \dots, \beta_n)^T$; linear comb. of Gaussians β_j; ⇒ f Gauss.; $E[f] = \Phi E[\beta] = 0$; $\text{cov}(f) = \Phi E[\beta\beta^T]\Phi^T = \frac{1}{\alpha}\Phi\Phi^T$; $p(f) = N(f \mid 0, \alpha^{-1}\Phi\Phi^T)$; $\begin{bmatrix} \\ \end{bmatrix}_{n\times 1} = \begin{bmatrix} \\ \end{bmatrix}_{n\times M} \begin{bmatrix} \\ \end{bmatrix}_{M\times 1}$; $[\phi_1(x_i) \dots \phi_M(x_i)]$)
- Gaussian process models replace explicit basis function representation with a direct specification in terms of a ***positive definite kernel function***

Mercer Kernel Functions

- ~~Predictions~~ Distributions are of the form

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

nxn matrix

nxM MxN

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

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

kernel fcn

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 nxn 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:
  - 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

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

- 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

Module 5: Classification

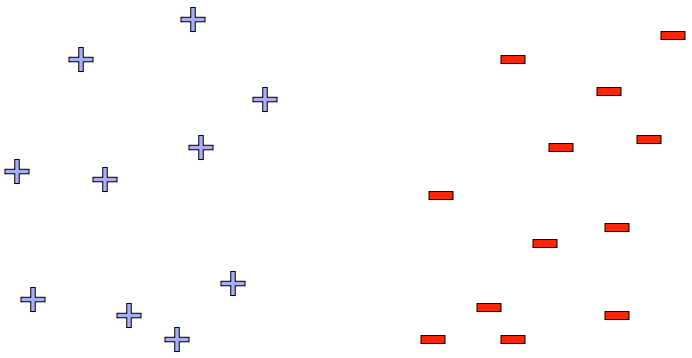
Support Vector Machines

STAT/BIOSTAT 527, University of Washington

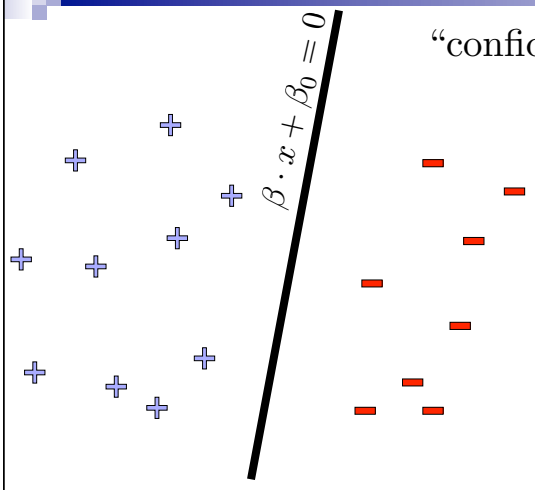
Emily Fox

June 4th, 2013

Linear classifiers – Which line is better?

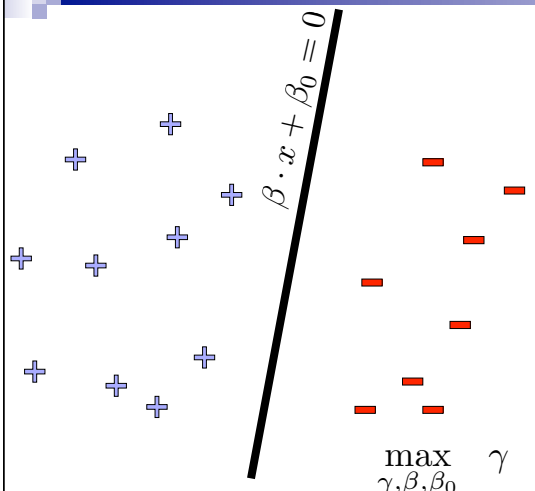


Pick the one with the largest margin!



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

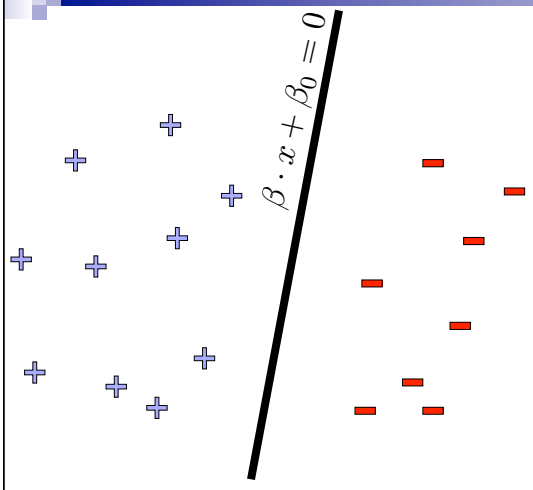
Maximize the margin



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

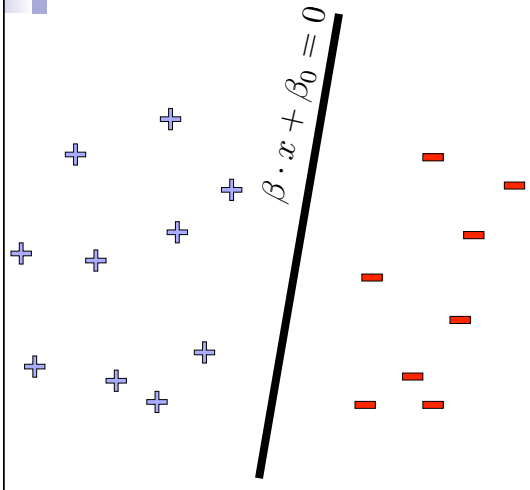
$$y_i(\beta \cdot x_i + \beta_0) \geq \gamma, \forall i \in \{1, \ldots, n\}$$

But there are many planes…



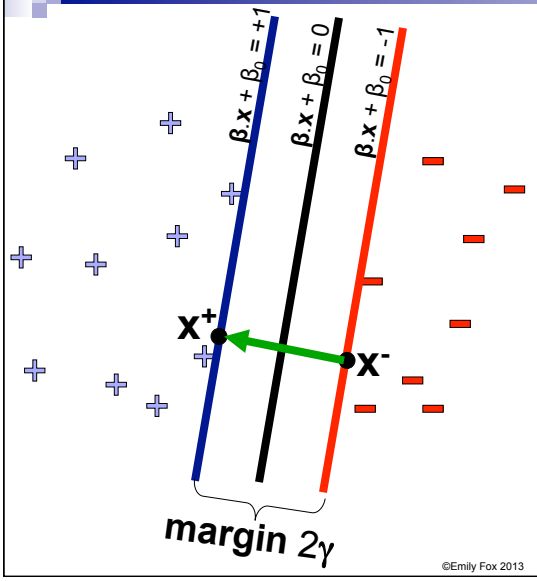
Review: Normal to a plane

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

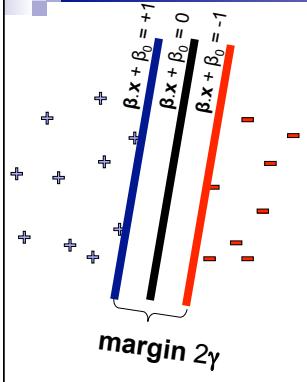


A Convention: Normalized margin – Canonical hyperplanes

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



Margin maximization using canonical hyperplanes



Unnormalized problem:

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

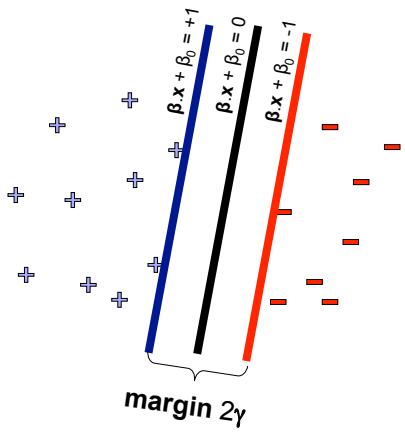
$$y_i(\beta \cdot x_i + \beta_0) \geq \gamma, \forall i \in \{1, \ldots, n\}$$

Normalized Problem:

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

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

Support vector machines (SVMs)

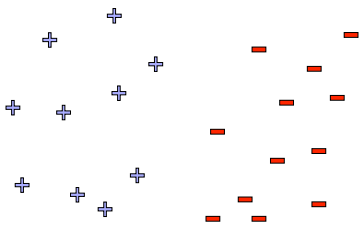


$$\min_{\beta,\beta_0} \quad ||\beta||_2^2$$

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

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

What if the data are not linearly separable?

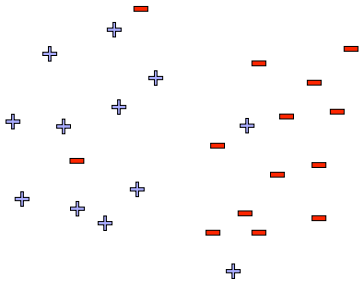


Use features of features of features of features….

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:

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

SVM:

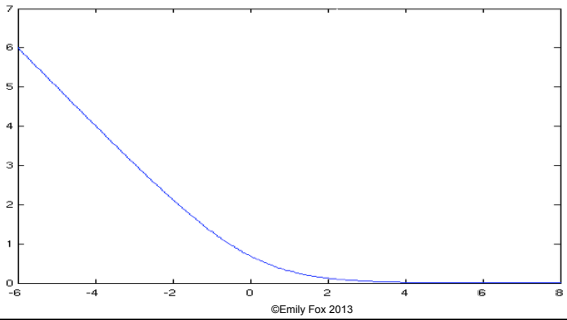
$$\min_{\beta,\beta_0} ||\beta||_2^2 + C \sum_{i=1}^{n} (1 - y_i(\beta \cdot x_i + \beta_0))_+$$

Logistic regression:

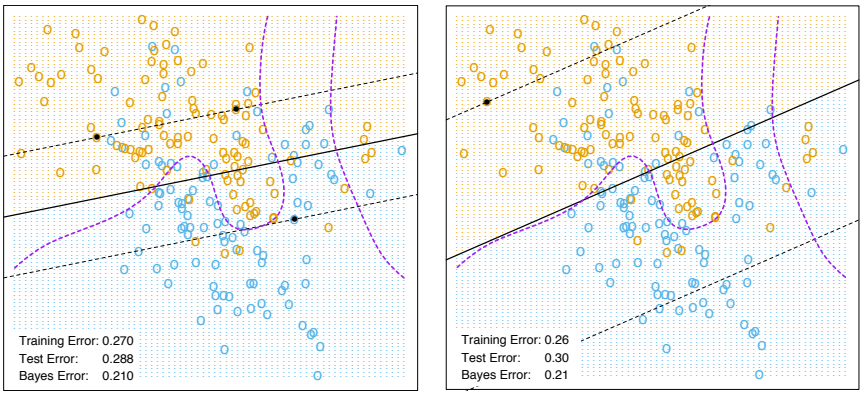
$$p(Y = 1 \mid x, \beta) = \frac{1}{1 + e^{-(\beta \cdot x + \beta_0)}}$$

Log loss:

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



Mixture Model Example



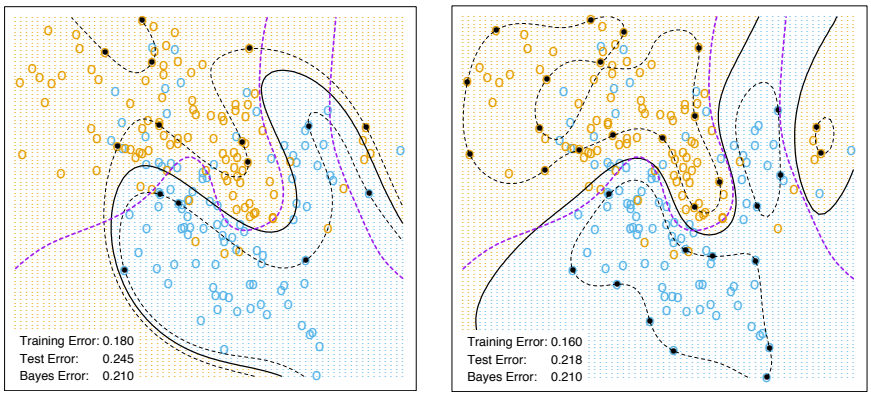
$C = 10000$ $C = 0.01$

From Hastie, Tibshirani, Friedman book

Mixture Model Example - Kernels

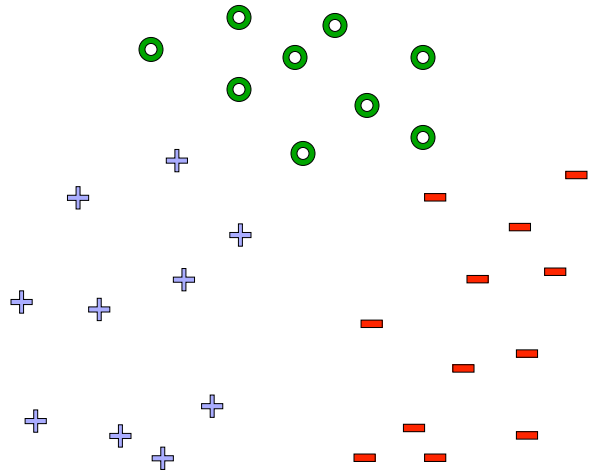
SVM - Degree-4 Polynomial in Feature Space

SVM - Radial Kernel in Feature Space

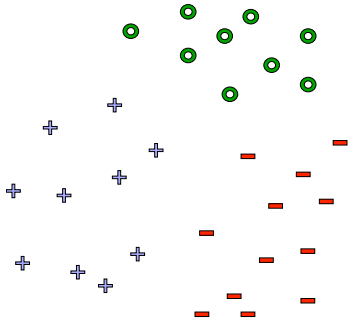


From Hastie, Tibshirani, Friedman book

What about multiple classes?



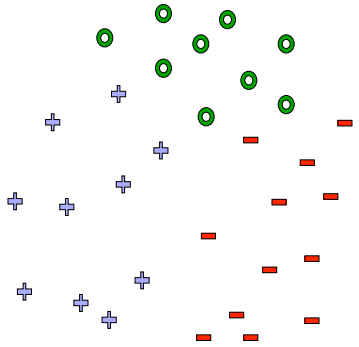
One against All



Learn 3 classifiers:

Learn 1 classifier: Multiclass SVM

Simultaneously learn 3 sets of weights

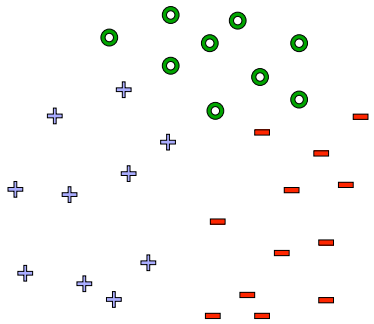


$$\beta^{(y_i)} \cdot x_i + \beta_0^{(y_i)} \geq \beta^{(y')} \cdot x_i + \beta_0^{(y')} + 1, \forall y' \neq y_i, \forall i$$

Learn 1 classifier: Multiclass SVM

$$\beta^{(y_i)} \cdot x_i + \beta_0^{(y_i)} \geq \beta^{(y')} \cdot x_i + \beta_0^{(y')} + 1, \forall y' \neq y_i, \forall i$$

$$\min_{\beta, \beta_0} \sum_y \beta^{(y)} \cdot \beta^{(y)} + C \sum_i \xi_i$$



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 optimize SVMs with SGD
  - Many other approaches possible
- Handling multiple classes

Reading

- Hastie, Tibshirani, Friedman – 12.1-12.3