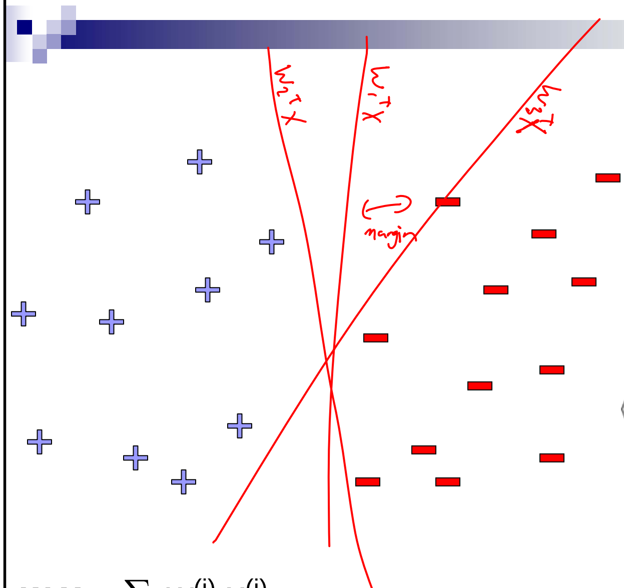
Support Vector, cont.
Machines, SVMs

Machine Learning – 10701/15781

Carlos Guestrin
Carnegie Mellon University

October 17th, 2007

Linear classifiers – Which line is better?



Data: m dims

$$\left\langle x_1^{(1)}, \ldots, x_1^{(m)}, y_1 \right\rangle$$
$$\vdots$$
$$\left\langle x_n^{(1)}, \ldots, x_n^{(m)}, y_n \right\rangle$$

n data

Example i:

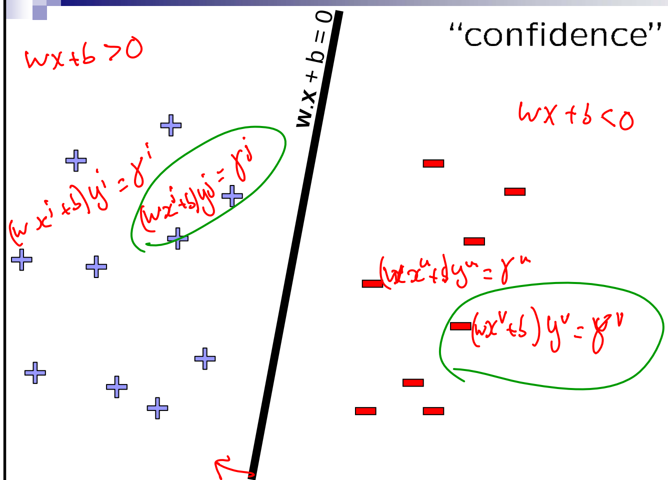
$\left\langle x_i^{(1)}, \ldots, x_i^{(m)} \right\rangle$ — m features

$y_i \in \{-1, +1\}$ — class

w.x = $\sum_j w^{(j)} x^{(j)}$

Pick the one with the largest margin!

“confidence” $= \left(\mathbf{w}.\mathbf{x}_j + b\right) y_j$



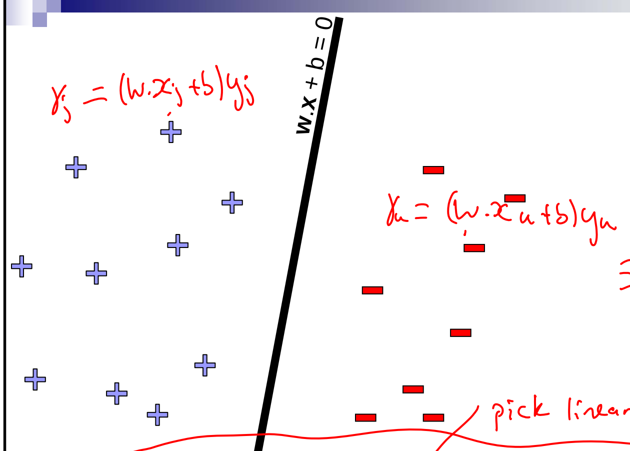
$\max_{w,b} \min_{i=\{1..n\}} \gamma^i$

pick plane where worst ~~nearer~~ point has highest confidence

w.x = $\sum_j$ w$^{(j)}$ x$^{(j)}$

Maximize the margin

$\gamma_1 = 0.1 \geq \gamma$
$\gamma_2 = 0.2 \geq \gamma$
$\gamma_3 = 0.05 \geq \gamma$



$\max_{w,b} \min_i (w.x_i+b)y_i$

$\equiv \max_{w,b} \min_i \gamma_i$

$(w x_i+b)y_i \geq \gamma_i$

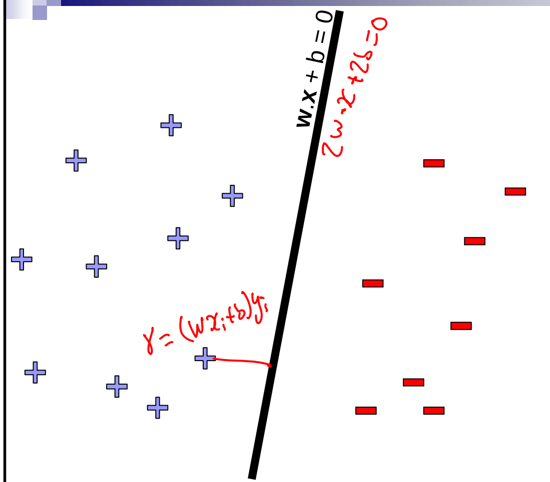
$\equiv \max_{w,b} \gamma$

$(w.x_i+b)y_i \geq \gamma \;\forall_i$

pick linear classifier

$$\text{maximize}_{\gamma,\mathbf{w},b} \quad \gamma$$
$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq \gamma, \;\; \forall j \in \text{Dataset}$$

But there are a many planes…



what if: new plane

$\bar{w} = 2w$

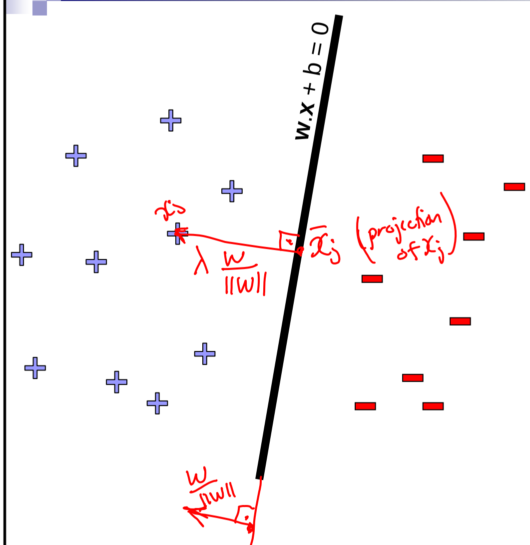
$\bar{b} = 2b$

↳ now $\bar{\gamma} = 2\gamma$

I can be as confident as I want by increasing w

Review: Normal to a plane

$$\mathbf{x}_j = \bar{\mathbf{x}}_j + \lambda_j \frac{\mathbf{w}}{||\mathbf{w}||}$$



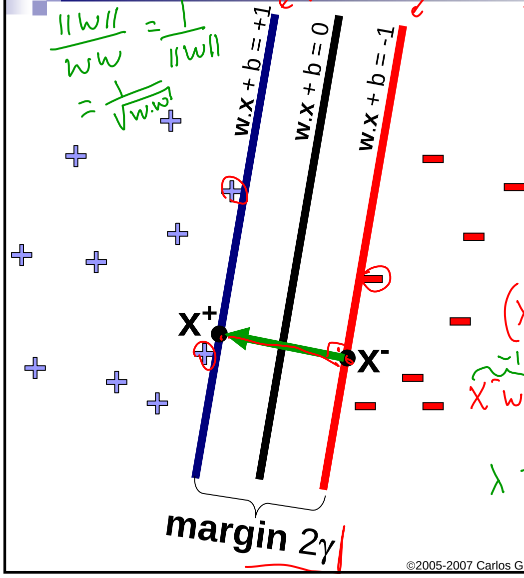
Normalized margin – Canonical hyperplanes

$$\mathbf{x}_j = \bar{\mathbf{x}}_j + \lambda \frac{\mathbf{w}}{||\mathbf{w}||}$$

plus plane

minus plane

$\frac{||w||}{ww} = \frac{1}{||w||} = \frac{1}{\sqrt{w.w}}$



$x^+ = x^- + \lambda \frac{w}{||w||}$

plug

$x^+ w + b = 1$

$\left(x^- + \lambda \frac{w}{||w||}\right) \cdot w + b = 1$

$\lambda = 2\gamma$

$\underbrace{x^- w + b}_{-1} + \lambda \frac{ww}{||w||} = 1$

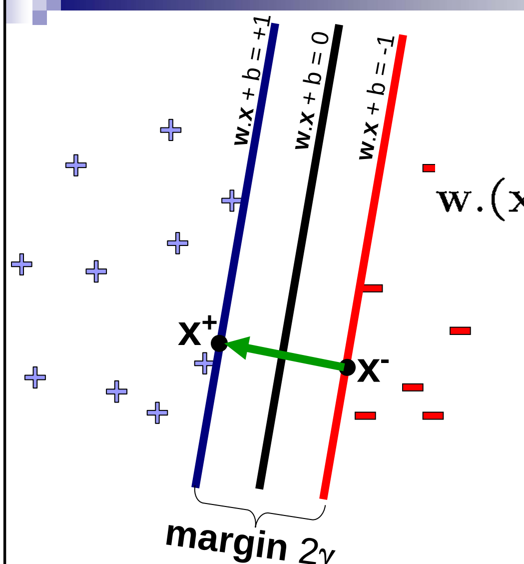
$\lambda = 2 \frac{||w||}{ww} \Rightarrow \boxed{\gamma = \frac{1}{\sqrt{ww}}}$

Normalized margin – Canonical hyperplanes

$$\mathbf{x}^+ = \mathbf{x}^- + \lambda \mathbf{w}$$

$$\mathbf{w}.\mathbf{x}^+ + b = 1$$

$$\mathbf{w}.(\mathbf{x}^- + \lambda \frac{\mathbf{w}}{||\mathbf{w}||}) + b = 1$$

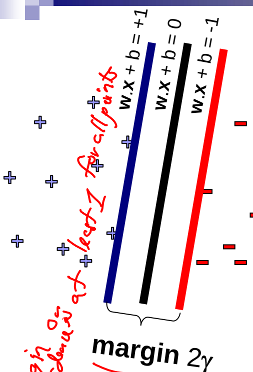


$$\lambda = \frac{2}{||\mathbf{w}||}$$

$$\boxed{\gamma = \frac{1}{\sqrt{\mathbf{w}.\mathbf{w}}}}$$

Margin maximization using canonical hyperplanes

$$\gamma = \frac{1}{\sqrt{\mathbf{w}.\mathbf{w}}}$$



margin on "canonical" hyperplane at least 1 for all points

$$\text{maximize}_{\gamma,\mathbf{w},b} \quad \gamma$$
$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq \gamma, \ \forall j \in \text{Dataset} \quad , \gamma \leq 1$$

$\max \frac{1}{\sqrt{w.w}}$

$(w.x_j+b)\, y_j \geq \gamma \quad , \gamma \leq 1$

$\equiv \min w.w$

$(w.x_j+b)\, y_j \geq \gamma \ ; \ \gamma \leq 1$

$\max \frac{1}{\sqrt{w.w}}$

$\equiv \min \sqrt{w.w}$

$\equiv \min w.w$

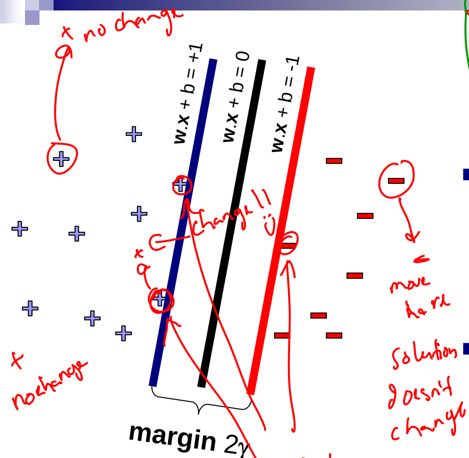
$\sqrt{\ }$ is a monotonic function

maximum achieved when $\gamma = 1$

SVM

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

Support vector machines (SVMs)



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

margin of at least 1

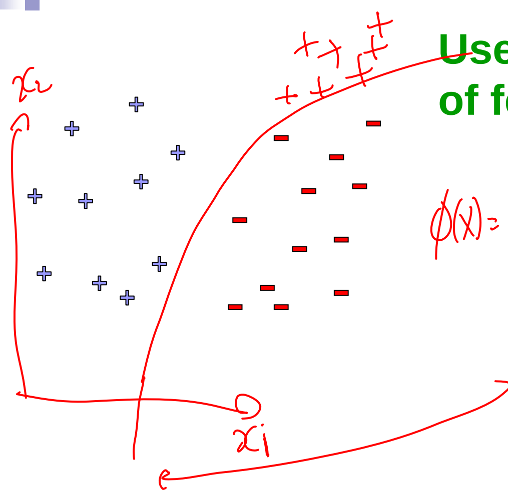
- Solve efficiently by quadratic programming (QP)
  - Well-studied solution algorithms

solution doesn't change

- Hyperplane defined by support vectors

points where $(w.x_j+b)\, y_j = 1$ (support vectors)

What if the data is not linearly separable?

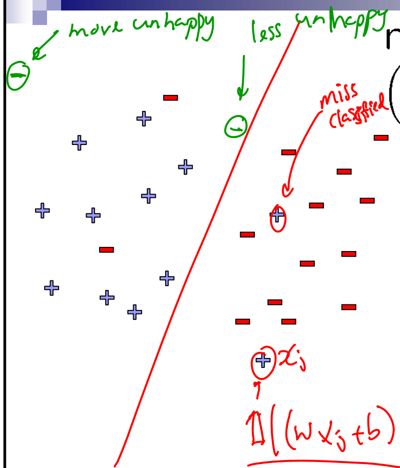


Use features of features of features of features….

$$\phi(x) = \begin{pmatrix} x_1 \\ x_2 \\ x_1 x_2 \\ x_1^2 \\ x_2^2 \end{pmatrix}$$

w. no label noise, ∃ space where points are linearly separable

What if the data is still not linearly separable?



penalty

$$\text{minimize}_{\mathbf{w},b} \quad \mathbf{w}.\mathbf{w} + C \, \#(\text{mistakes})$$

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

$\mathbb{1}[(w x_j + b) y_j < 0]$ ∈ {0,1}

- Minimize **w.w** and number of training mistakes
  - Tradeoff two criteria?

want unhappyness to depend on margin !!

- Tradeoff #(mistakes) and **w.w**
  - 0/1 loss
  - Slack penalty *C*
  - Not QP anymore
  - Also doesn't distinguish near misses and really bad mistakes

relaxed constraints

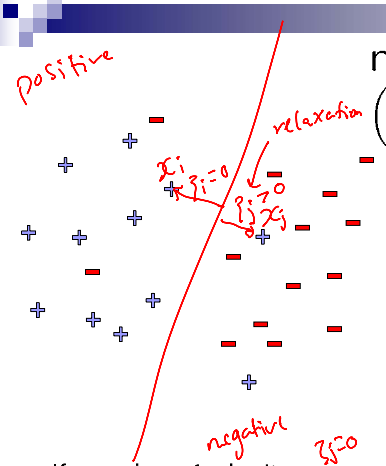
Slack variables – Hinge loss

"margin" regularized

$$\text{minimize}_{\mathbf{w},b,\xi} \quad \mathbf{w}.\mathbf{w} + C\sum_{j=1}^{n}\xi_j$$

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

$\xi_j \geq 0$



$C \to \infty \quad \to \quad$ set $\xi_j = 0$, $w =$ usual SVM (not relaxed)

not linearly separable "no solution"

- If margin $\geq$ 1, don't care $\xi_j = 0$
- If margin < 1, pay linear penalty ξ_j

$C \to 0 \longrightarrow w = 0$

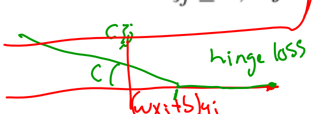
Side note: What's the difference between SVMs and logistic regression?

SVM:

$$\text{minimize}_{\mathbf{w},b} \quad \mathbf{w}.\mathbf{w} + C\sum_j \xi_j$$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j, \ \forall j$$

$$\xi_j \geq 0, \ \forall j$$



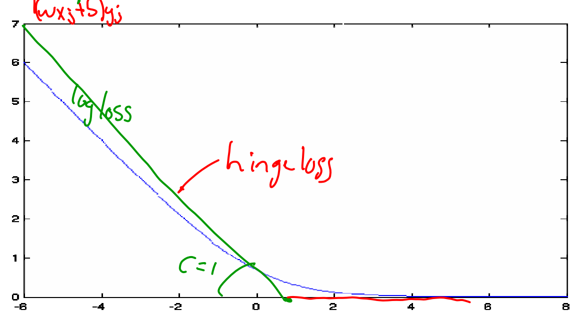
Regularized w^2

Logistic regression:

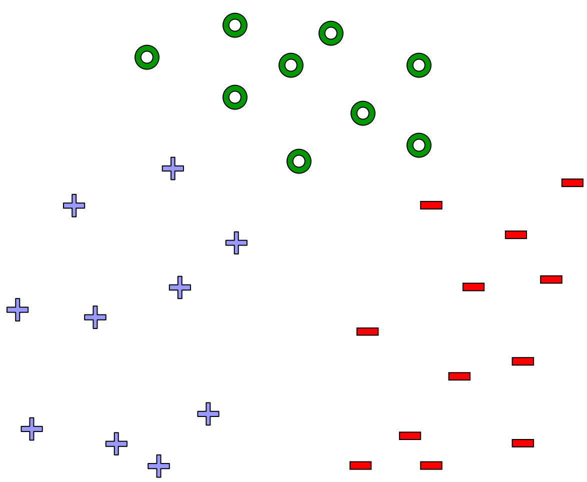
$$P(Y=1 \mid x, \mathbf{w}) = \frac{1}{1+e^{-(\mathbf{w}.\mathbf{x}+b)}}$$

Log loss:

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

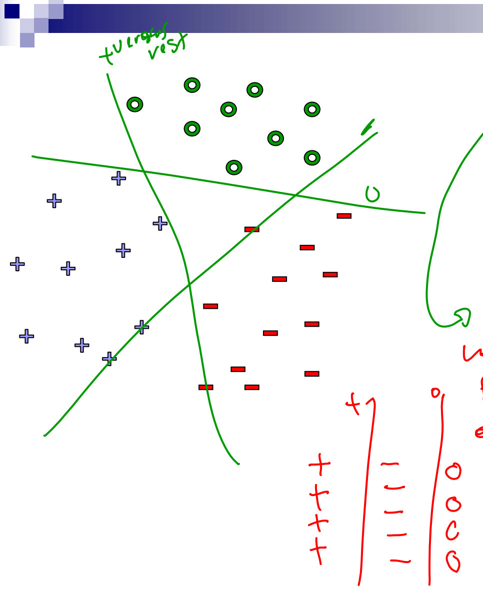


What about multiple classes?



©2005-2007 Carlos Guestrin

One against All



Learn 3 classifiers:

+ versus rest {0, −} w^+

− versus rest {0, +} w^-

0 versus rest {+, −} w^0

use SVM implementations

+ positive, {0, −} negative

winner : class with highest $w^+ x^j + b^+$, $w^- x^j + b^-$, $w^0 x^j + b^0$ for x^j

← no − plane separates − from {+, 0}

©2005-2007 Carlos Guestrin

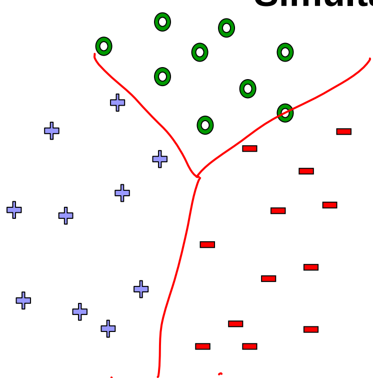
Learn 1 classifier: Multiclass SVM

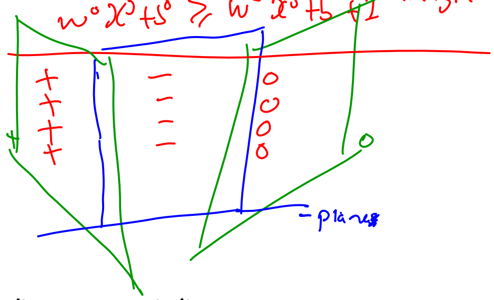
Simultaneously learn 3 sets of weights

if train point x^j is of class o

$w^o x^j + b^o \geq w^+ x^j + b^+ + 1$ margin

$w^o x^j + b^o \geq w^- x^j + b^- + 1$





class of point x^j

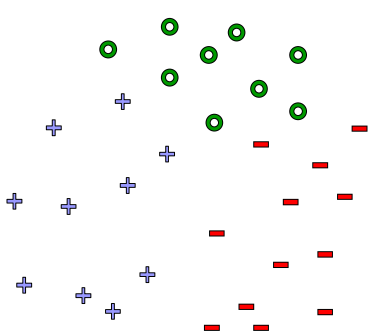
$$\mathbf{w}^{(y_j)}.\mathbf{x}_j + b^{(y_j)} \geq \mathbf{w}^{(y')}.\mathbf{x}_j + b^{(y')} + 1, \ \forall y' \neq y_j, \ \forall j$$

Learn 1 classifier: Multiclass SVM

$$\text{minimize}_{\mathbf{w},b} \quad \sum_y \mathbf{w}^{(y)}.\mathbf{w}^{(y)} + C\sum_j \xi_j$$

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

$$\xi_j \geq 0, \ \forall j$$



What you need to know

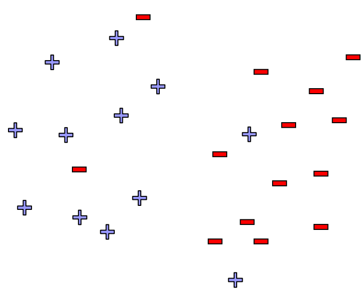
- Maximizing margin
- Derivation of SVM formulation
- Slack variables and hinge loss
- Relationship between SVMs and logistic regression
  - 0/1 loss
  - Hinge loss
  - Log loss
- Tackling multiple class
  - One against All
  - Multiclass SVMs

SVMs, Duality and the Kernel Trick

Machine Learning – 10701/15781
Carlos Guestrin
Carnegie Mellon University

October 17th, 2007

SVMs reminder



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

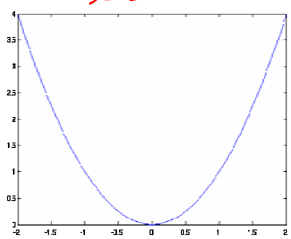
Today's lecture

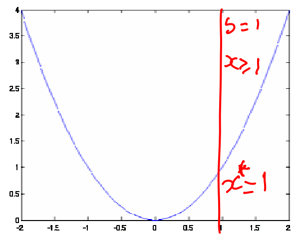
- Learn one of the most interesting and exciting recent advancements in machine learning
  - The "kernel trick"
  - High dimensional feature spaces at no extra cost!
- But first, a detour
  - Constrained optimization!

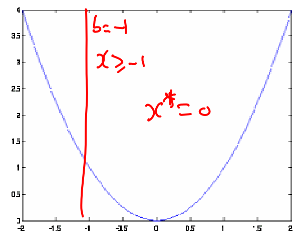
Constrained optimization

$$\min_x \; x^2$$
$$\text{s.t.} \quad x \geq b$$

no constraint
$x^* = 0$



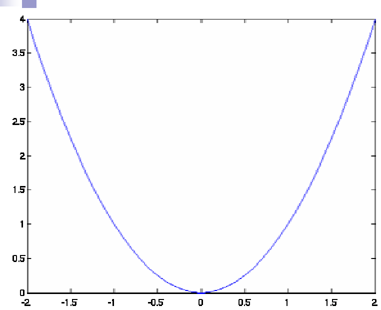




constraint important
$x^* = b$

constraint unimportant
$x^* > b$

Lagrange multipliers – Dual variables



$$\min_x \; x^2$$
$$\text{s.t.} \quad x \geq b$$

Moving the constraint to objective function
Lagrangian:

lagrange multiplier

$$L(x, \alpha) = x^2 - \alpha(x - b)$$
$$\text{s.t.} \quad \alpha \geq 0$$

$x > b \Rightarrow \alpha = 0$

$x = b \Rightarrow \alpha > 0$

Solve:

$$\min_x \max_\alpha \; L(x, \alpha)$$
$$\text{s.t.} \quad \alpha \geq 0$$

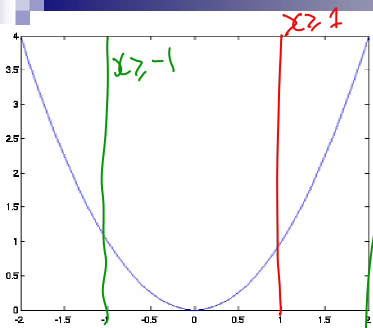
suppose, I propose α player

$x > b \Rightarrow x - b > 0 \Rightarrow \alpha = 0$

$x < b \Rightarrow x - b < 0 \Rightarrow \alpha = \infty$

x player - very unhappy

Lagrange multipliers – Dual variables



Solving: $\min_x \max_\alpha \; x^2 - \alpha(x-b)$

s.t. $\alpha \geq 0$

$L =$

$$\frac{\partial L}{\partial x} = 2x - \alpha = 0 \Rightarrow \alpha = 2x$$

$$\frac{\partial L}{\partial \alpha} = -(x-b)$$

$x = 1 \Rightarrow \frac{\partial L}{\partial \alpha} = 0$

$\alpha = 2 > 0$

constraint relevant

if I set $x = -1$ $\Downarrow$ $\frac{\partial L}{\partial \alpha} = 0$, but $\alpha = -2 < 0$

$x = 0$

no way I can set $\frac{\partial L}{\partial \alpha} = 0$

$\Rightarrow$ ignore $\alpha = 0$

constraint irrelevant

Dual SVM derivation (1) – the linearly separable case

$$\text{minimize}_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}.\mathbf{w}$$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1, \; \forall j$$

$\leftarrow$ one α_j for each point

$$L(w,b,\alpha) = \frac{1}{2} w.w - \sum_j \alpha_j \left((w x_j + b) y_j - 1 \right)$$

Dual SVM derivation (2) – the linearly separable case

$m = \#$ features

$$L(\mathbf{w},\alpha) = \tfrac{1}{2}\mathbf{w}.\mathbf{w} - \textstyle\sum_j \alpha_j \left[\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j - 1\right]$$

$\alpha_j \geq 0, \ \forall j$

m×1 vector   m×1 vector

$$\frac{\partial L}{\partial w} = w - \sum_j \alpha_j x_j y_j = 0$$

$$\Rightarrow w = \sum_{j=1}^{n} \alpha_j x_j y_j$$

b: suppose datapoint x_j is a support vector

$(w x_j + b) y_j = 1$

$\alpha_j > 0$

solve for b

$$\mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$$

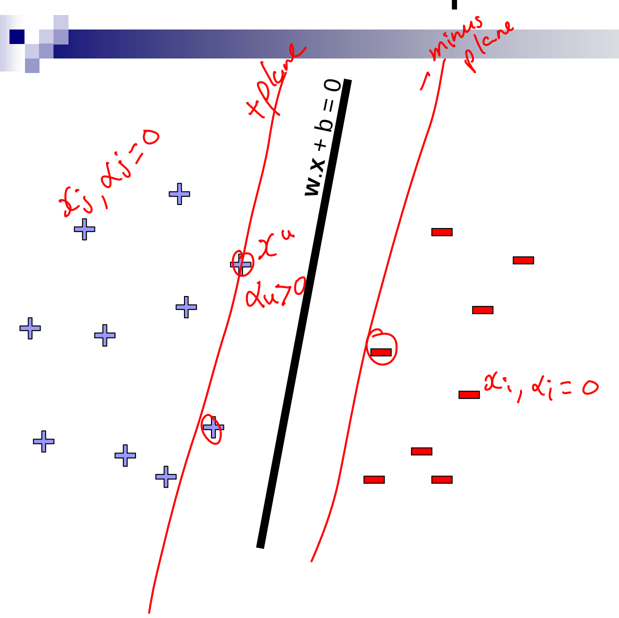
$\text{minimize}_{\mathbf{w}} \quad \frac{1}{2}\mathbf{w}.\mathbf{w}$

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

$$b = y_k - \mathbf{w}.\mathbf{x}_k$$

for any k where $\alpha_k > 0$

Dual SVM interpretation



$$\mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$$

Support vectors x_j

$\Leftrightarrow \alpha_j > 0$

Dual SVM formulation – the linearly separable case

$x_i \cdot x_j$ dot prod
$= x_i \bullet x_j$
compute dot products extremely fast in high dims.

free variables

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$

Plug into lagrangian

constraints
$$\sum_i \alpha_i y_i = 0$$
$$\alpha_i \geq 0$$

came from $\frac{\partial L}{\partial b} = 0$

$$\boxed{\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i}$$

$$b = y_k - \mathbf{w}.\mathbf{x}_k$$

for any k where $\alpha_k > 0$

dual problem

Quadratic Program just like before

Dual SVM derivation – the non-separable case

$$\text{minimize}_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_j \xi_j$$
$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j, \ \forall j \quad \leftarrow \alpha_j \text{ multipliers}$$
$$\xi_j \geq 0, \ \forall j \quad \leftarrow \mu_j \text{ multiplers}$$

whether or not point violates margin

$\mu_j = 0$, $\leftarrow \xi_j > 0$

$\mu_j > 0$ doesn't violate margin $\xi_j = 0$

Dual SVM formulation – the non-separable case

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$

$$\sum_i \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$

only difference

α_j can't be too big

α_j very large "when you really constraint problem"

targets

$\frac{\partial L}{\partial w}$

derivation of dual for problem with slack penalty similar

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

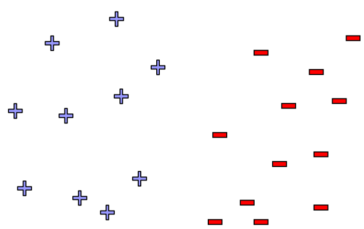
$$b = y_k - \mathbf{w}.\mathbf{x}_k$$

for any k where $C > \alpha_k > 0$

Why did we learn about the dual SVM?

- There are some quadratic programming algorithms that can solve the dual faster than the primal
- But, more importantly, the "**kernel trick**"!!!
  - Another little detour…

Reminder from last time: What if the data is not linearly separable?



Use features of features of features of features….

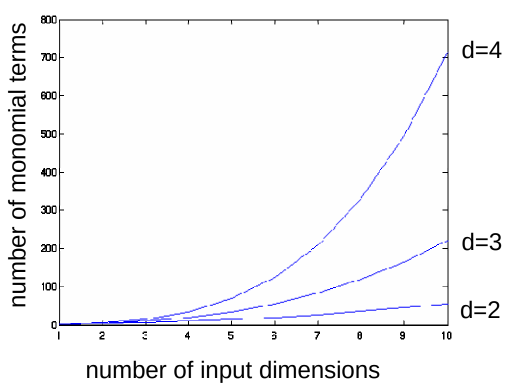
$$\Phi(\mathbf{x}) : R^m \mapsto F$$

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

Dual formulation only depends on dot-products, not on **w**!

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$
$$\sum_i \alpha_i y_i = 0$$
$$C \geq \alpha_i \geq 0$$

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$
$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$
$$\sum_i \alpha_i y_i = 0$$
$$C \geq \alpha_i \geq 0$$

Dot-product of polynomials

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

Finally: the "kernel trick"!

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

$$\sum_i \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$

$$\boxed{\begin{array}{l} \mathbf{w} = \sum_i \alpha_i y_i \Phi(\mathbf{x}_i) \\ b = y_k - \mathbf{w}.\Phi(\mathbf{x}_k) \\ \text{for any } k \text{ where } C > \alpha_k > 0 \end{array}}$$

- Never represent features explicitly
  - Compute dot products in closed form
- Constant-time high-dimensional dot-products for many classes of features
- Very interesting theory – Reproducing Kernel Hilbert Spaces
  - Not covered in detail in 10701/15781, more in 10702

Polynomial kernels

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

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

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

Common kernels

- Polynomials of degree d

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

- Polynomials of degree up to d

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

- Gaus

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

- Sigmoid

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

Overfitting?

- Huge feature space with kernels, what about overfitting???
  - Maximizing margin leads to sparse set of support vectors
  - Some interesting theory says that SVMs search for simple hypothesis with large margin
  - Often robust to overfitting

What about at classification time

- For a new input **x**, if we need to represent Φ(**x**), we are in trouble!
- Recall classifier: sign(**w**.Φ(**x**)+b)
- Using kernels we are cool!

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

$$\mathbf{w} = \sum_i \alpha_i y_i \Phi(\mathbf{x}_i)$$

$$b = y_k - \mathbf{w}.\Phi(\mathbf{x}_k)$$

for any k where $C > \alpha_k > 0$

SVMs with kernels

- Choose a set of features and kernel function
- Solve dual problem to obtain support vectors α_i
- At classification time, compute:

$$\mathbf{w} \cdot \Phi(\mathbf{x}) = \sum_i \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i)$$

$$b = y_k - \sum_i \alpha_i y_i K(\mathbf{x}_k, \mathbf{x}_i)$$

for any k where $C > \alpha_k > 0$

Classify as

$$sign\left(\mathbf{w} \cdot \Phi(\mathbf{x}) + b\right)$$

What's the difference between SVMs and Logistic Regression?

	SVMs	Logistic Regression
Loss function		
High dimensional features with kernels		

Kernels in logistic regression

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

- Define weights in terms of support vectors:

$$\mathbf{w} = \sum_i \alpha_i \Phi(\mathbf{x}_i)$$

$$\begin{aligned} P(Y=1 \mid x, \mathbf{w}) &= \frac{1}{1+e^{-(\sum_i \alpha_i \Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x})+b)}} \\ &= \frac{1}{1+e^{-(\sum_i \alpha_i K(\mathbf{x},\mathbf{x}_i)+b)}} \end{aligned}$$

- Derive simple gradient descent rule on α_i

What's the difference between SVMs and Logistic Regression? (Revisited)

	SVMs	Logistic Regression
Loss function	Hinge loss	Log-loss
High dimensional features with kernels	Yes!	Yes!

What you need to know

- Dual SVM formulation
  - How it's derived
- The kernel trick
- Derive polynomial kernel
- Common kernels
- Kernelized logistic regression
- Differences between SVMs and logistic regression