Two SVM tutorials linked in class website (please, read both):

- High-level presentation with applications (Hearst 1998)
- Detailed tutorial (Burges 1998)

SVMs, Duality and the Kernel Trick

Machine Learning – 10701/15781

Carlos Guestrin

Carnegie Mellon University

February 27th, 2006

Announcements

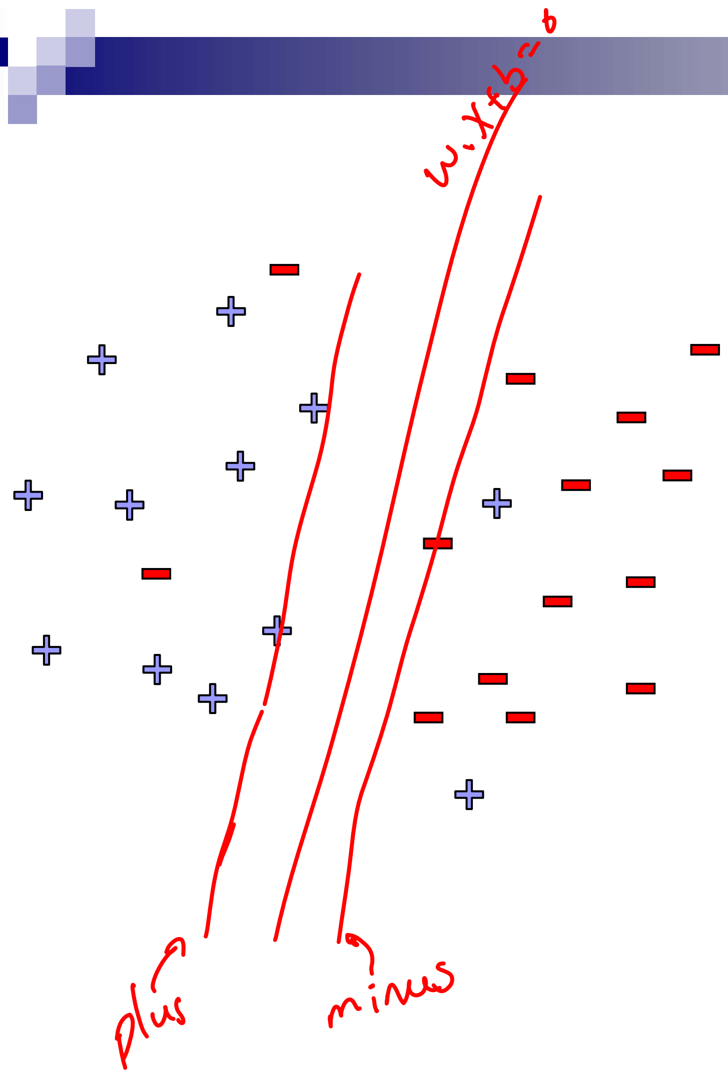
- Third homework
  - is out
  - Due March 1st

start early!! :)

- Final assigned by registrar:
  - May 12, 1-4p.m Friday
  - Location TBD

- Midterm
  - March 8th, a week from Wednesday
  - Open book, notes, papers, etc. No computers

SVMs reminder



slack penalty

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

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

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

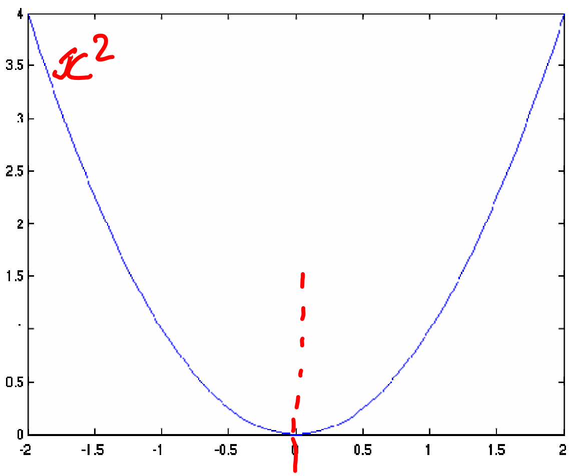
Today's lecture

- 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

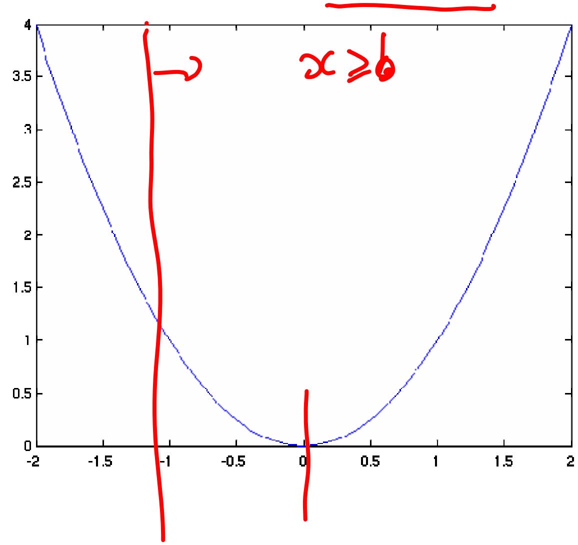
$$\min_x \ x^2$$
$$\text{s.t.} \quad \underline{x \geq b}$$

$\min_x x^2$



min

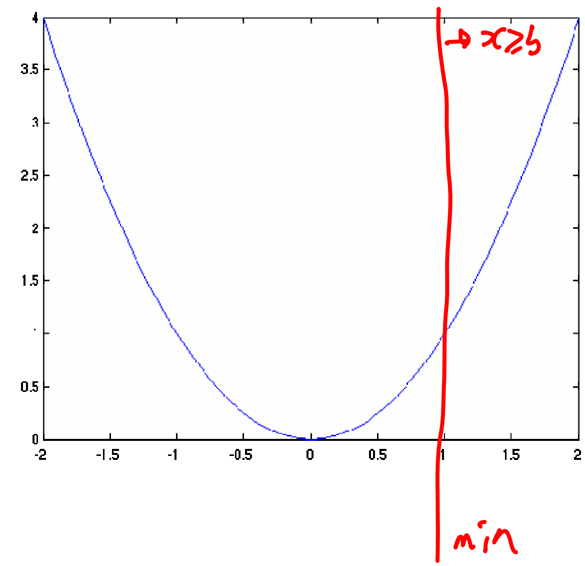
unconstrained



b

min

constraint doesn't matter



min

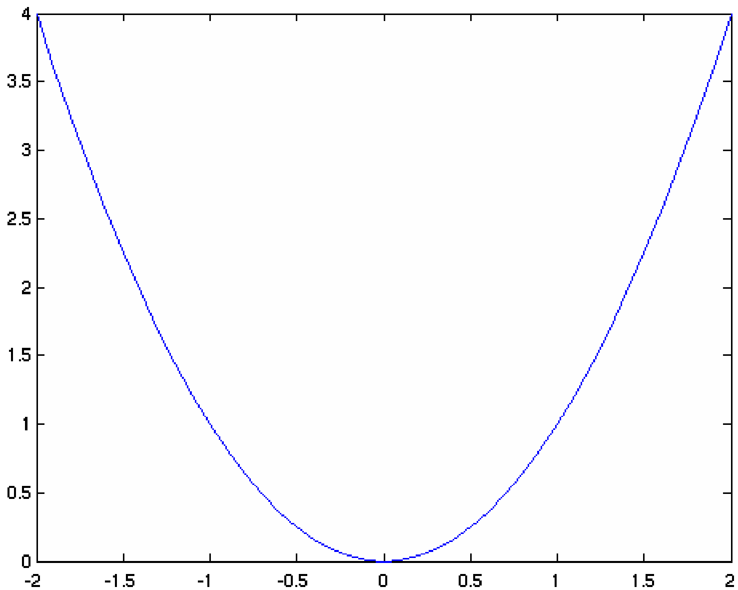
b

stopped by constraint

Lagrange multipliers – Dual variables

$$\min_x \; x^2$$

$$\text{s.t.} \quad x \geq b \; : \; x - b \geq 0$$



Moving the constraint to objective function

Lagrangian:

$$L(x, \alpha) = x^2 - \alpha(x - b)$$

$$\text{s.t.} \quad \alpha \geq 0$$

what if $x > b$: $x - b > 0$
constraint sat.
$\Downarrow$
$-\alpha(x-b) \rightarrow$ negative $\rightarrow \max_\alpha \rightarrow \alpha = 0$
$\min_x \rightarrow$ happy!!

$x < b$: $x - b < 0$ constraint not sat.
$-\alpha(x-b) \rightarrow$ positive $\rightarrow \max_\alpha \rightarrow \alpha = +\infty$
$\min_x \rightarrow$ v. unhappy!!

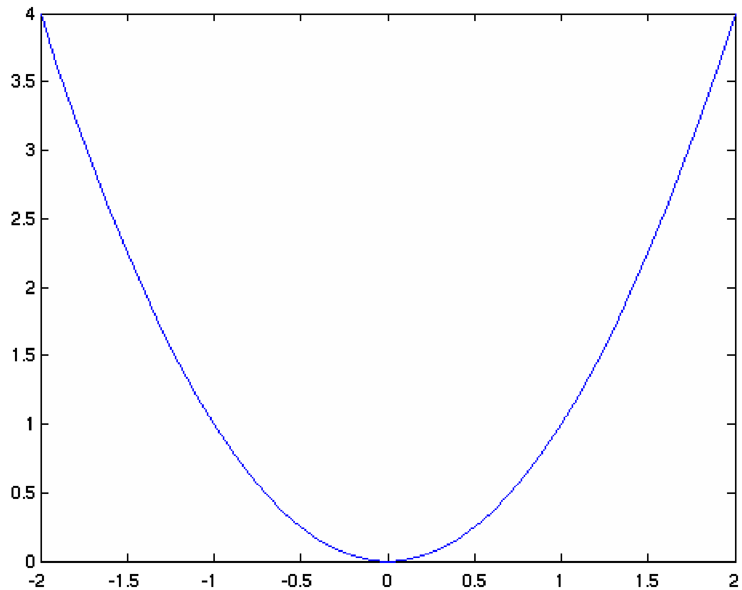
$\min_x \rightarrow$ only suggest x that sat. constraint!!

Solve:

$$\min_x \max_\alpha \; L(x, \alpha)$$

$$\text{s.t.} \quad \alpha \geq 0$$

Lagrange multipliers – Dual variables



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

s.t. $\alpha \geq 0$

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

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

$x = 0$

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

either $\alpha = 0$ and constraint is ignored

or if $\alpha = 0$ term $\alpha(x - b)$ irrelevant

or $\alpha \neq 0$ constraint plays a role

Dual SVM derivation (1) – the linearly separable case

simpler eqs.

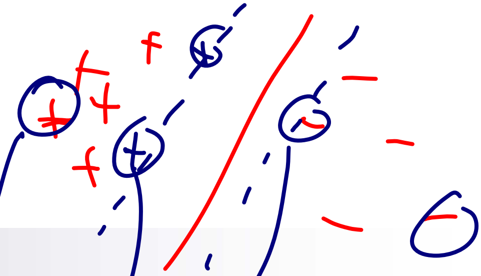
$$\text{minimize}_{\mathbf{w}} \quad \frac{1}{2}\mathbf{w}.\mathbf{w}$$

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

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

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

Dual SVM derivation (2) – the linearly separable case



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

$$\alpha_i \geq 0, \ \forall j$$

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

when $(w.x_j + b).y_j > 1 \Rightarrow \alpha_j = 0$

when $(w.x_j + b)\, y_j = 1$, $\alpha_j > 0$

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

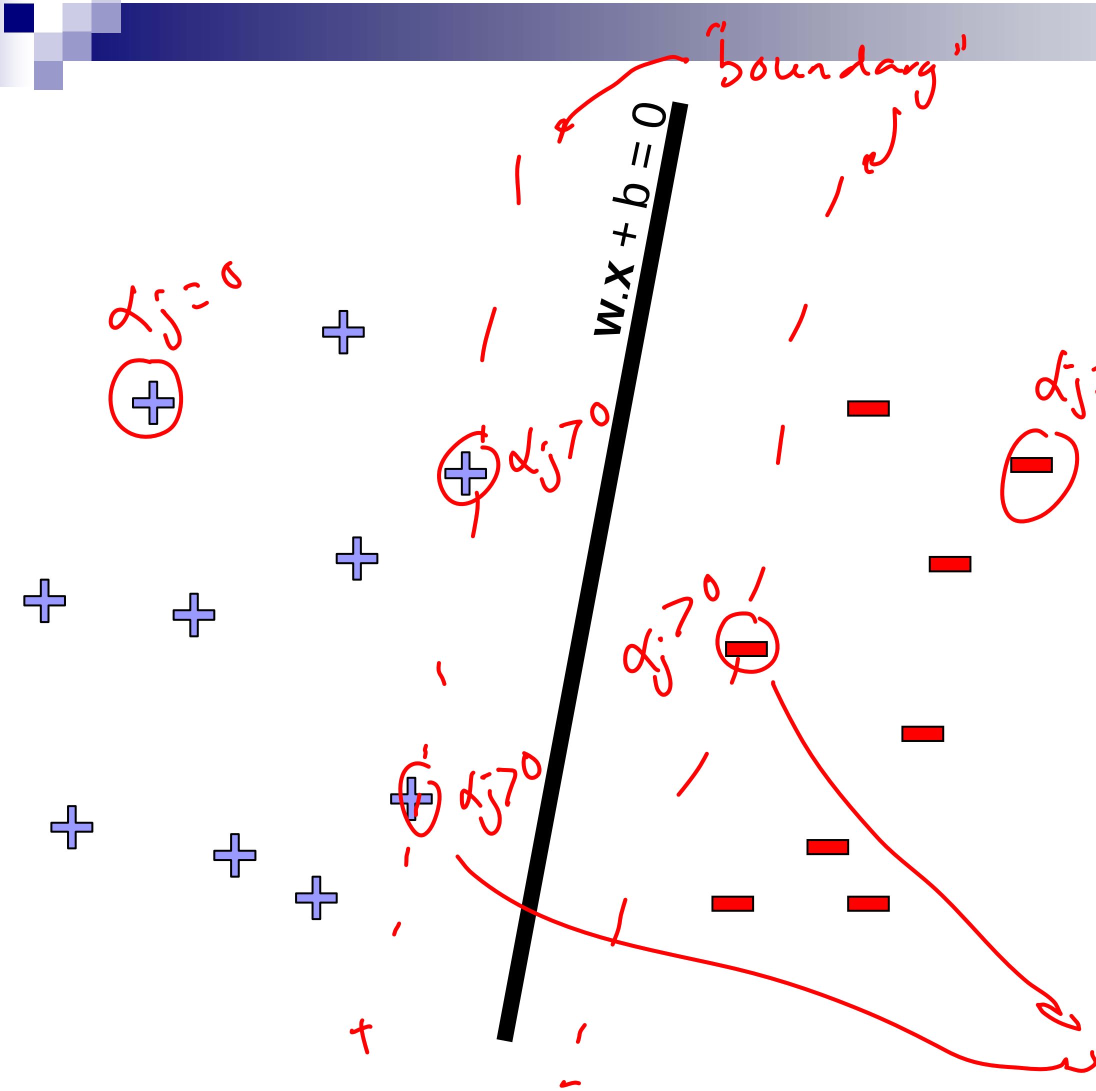
$$\text{minimize}_{\mathbf{w}} \quad \frac{1}{2}\mathbf{w}.\mathbf{w}$$

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

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

for any k where $\alpha_k > 0$

Dual SVM interpretation



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

$$= \sum_{i \in \text{on boundary}} \alpha_i y_i x_i$$

weight can be written as linear combination of input $y_i x_i$, but only care about support vectors

Dual SVM formulation – the linearly separable case

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

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0$$

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

dual program, solve SVM :

→ solve dual

obtain the α

get w, b

obj function dual → quadratic → dual quadric program

Dual SVM derivation – the non-separable case

$$\text{minimize}_{\mathbf{w}} \quad \tfrac{1}{2}\mathbf{w}.\mathbf{w} + C \sum_j \xi_j$$

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

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

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

$$- \sum_j \mu_j \xi_j$$

Dual SVM formulation – the non-separable case

m~~in~~aximize$_\alpha$ $\sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j$

$:= \{\alpha_1, \alpha_2, \ldots, \alpha_N\}$

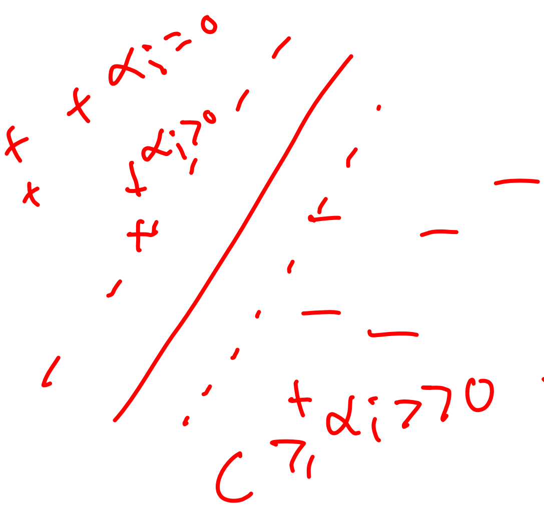
N training examples

$$\sum_i \alpha_i y_i = 0$$

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

α_i can't be too large

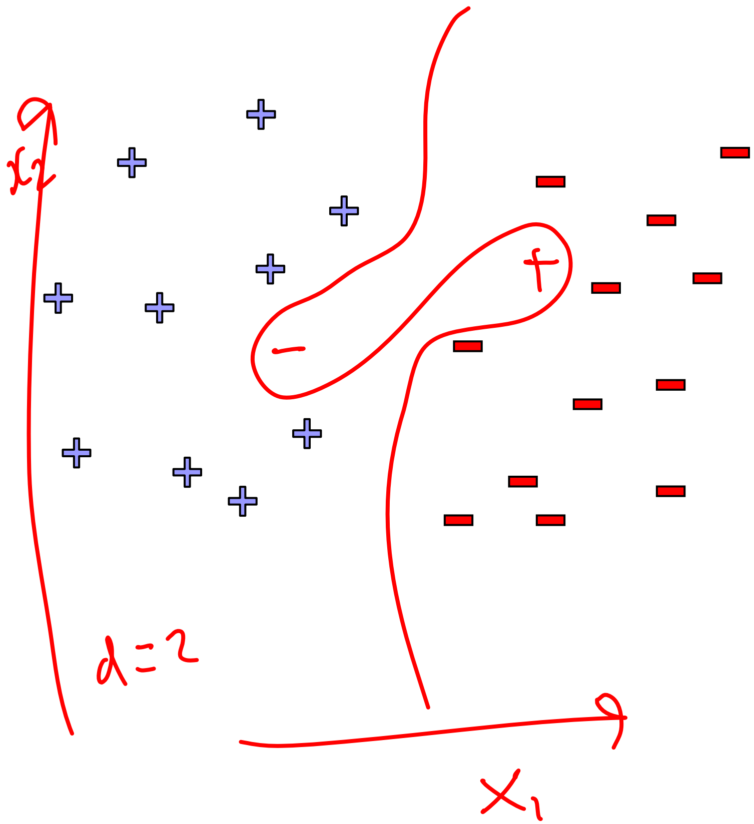
$$\boxed{\begin{array}{c} \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \\ \\ b = y_k - \mathbf{w}.\mathbf{x}_k \\ \text{for any } k \text{ where } C > \alpha_k > 0 \end{array}}$$



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 mapping

$x = (x_1, x_2)$

$$\phi(x) = \begin{pmatrix} 1 \\ x \\ x^2 \\ x^3 \\ \cancel{\sin x} \\ \vdots \\ x^4 \\ \vdots \end{pmatrix} \qquad \phi(x) = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \\ x_1 x_2 \\ x_1^3 \\ x_1 x_2^2 \\ x_1^2 x_2 \\ \vdots \end{pmatrix}$$

input d dim → poly degree p

size of $\phi(x)$

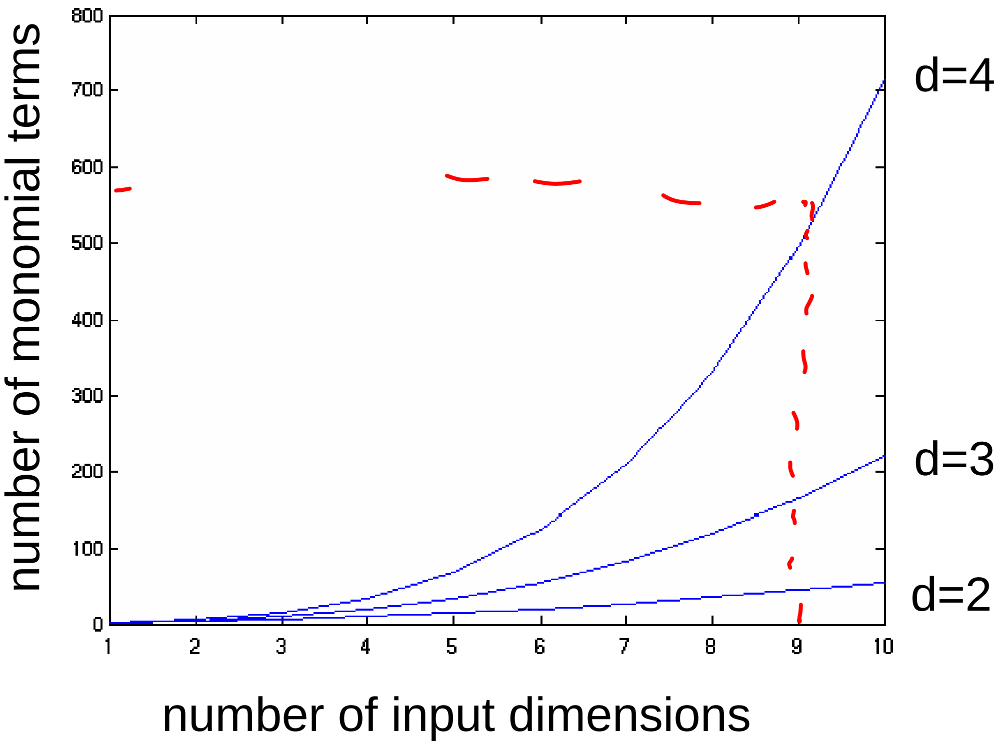
Feature space can get really large really quickly!

Higher order polynomials

degree of poly

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

m – input features
d – degree of polynomial



input 4

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

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

only thing is x

dot prod.

$x_i x_j = x_i \cdot x_j$

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

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

no w!

use features $\phi(x)$

all I need is $\phi(x_j) \cdot \phi(x_i)$

$K(x_j, x_i) = \phi(x_j) \cdot \phi(x_i)$

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

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$
$$\sum_i \alpha_i y_i = 0$$
$$C \geq \alpha_i \geq 0$$

Dot-product of polynomials

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

degree 1 ⇒ $\phi(u) = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$ $\quad \phi(v) = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$ $\quad \phi(u).\phi(v) = u_1 v_1 + u_2 v_2$

degree 2 ⇒ $\phi(u) = \begin{pmatrix} u_1^2 \\ u_1 u_2 \\ u_2 u_1 \\ u_2^2 \end{pmatrix}$ $\quad \phi(v) = \begin{pmatrix} v_1^2 \\ v_1 v_2 \\ v_2 v_1 \\ v_2^2 \end{pmatrix}$ $\quad \phi(u).\phi(v) = u_1^2 v_1^2 + u_1 u_2 v_1 v_2 + u_2 u_1 v_2 v_1 + u_2^2 v_2^2$

12 multiplies

$$\phi(u) \cdot \phi(v) = u_1^2 v_1^2 + u_1 u_2 v_1 v_2 + u_2 u_1 v_2 v_1 + u_2^2 v_2^2$$

$$= (u_1 v_1 + u_2 v_2)^2 \quad \leftarrow \text{3 multiplies}$$

$$= (u \cdot v)^2$$

Polynomials of degree exactly d: $\phi(u).\phi(v) = (u.v)^d$

Finally: the "kernel trick"!

i, j all pairs of data points including datapoint with itself.

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

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

$\sum_i \alpha_i y_i = 0$

$C \geq \alpha_i \geq 0$

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

if using poly. degree exactly d

$\forall ij$ compute $x_i \cdot x_j$

set $K(x_i, x_j) = (x_i \cdot x_j)^d$

solve dual Q.P.

get α

Finally: the “kernel trick”!

i,j all pairs of data points including datapoint with itself.

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

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

$$\sum_i \alpha_i y_i = 0$$

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

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

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

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

- 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:

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

- How about all monomials of degree up to d?
  - Solution 0: $\phi(u)\cdot\phi(v) = \sum_{i=0}^{d} (u\cdot v)^i$
  - Better solution: $(u\cdot v+1)^2 = (u\cdot v)^2 + u\cdot v + v\cdot u + 1$ $d=2$

$$\phi(u)\cdot\phi(v) = (u\cdot v+1)^d$$

$O(d)$ time

Common kernels

- Polynomials of degree exactly d

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

- Polynomials of degree up to including d

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

- Gaussian kernels

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

correspond infinite dimensional feature space

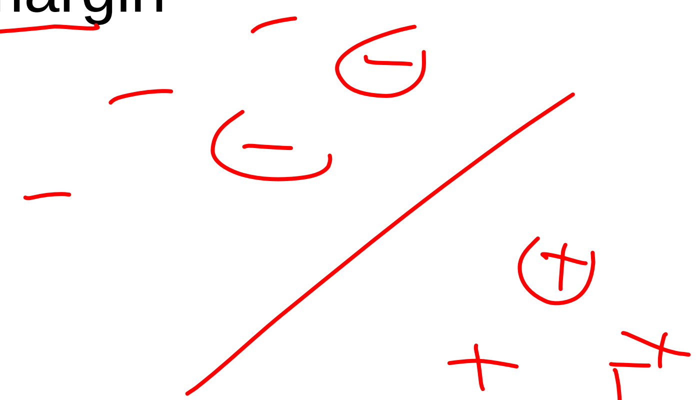
$\dim[\phi(x)] = \text{infinite}$

- Sigmoid

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

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



sparse solutions →
a few support vectors
→ less overfitting

What about at classification time

- For a new input **x**, if we need to represent $\Phi(\mathbf{x})$, we are in trouble! if have to write w, b, too large
- Recall classifier: sign(**w**.Φ(**x**)+b)
- Using kernels we are cool!

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

$$w \cdot \phi(x) = \sum_i \alpha_i y_i \underbrace{\phi(x) \cdot \phi(x_i)}_{\text{easy to compute}}$$

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

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

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:

new x, old data

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

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

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

Acknowledgment

- SVM applet:
  - http://www.site.uottawa.ca/~gcaron/applets.htm