Support Vector Machines (SVMs)

Machine Learning - 10601

Geoff Gordon, MiroslavDudík

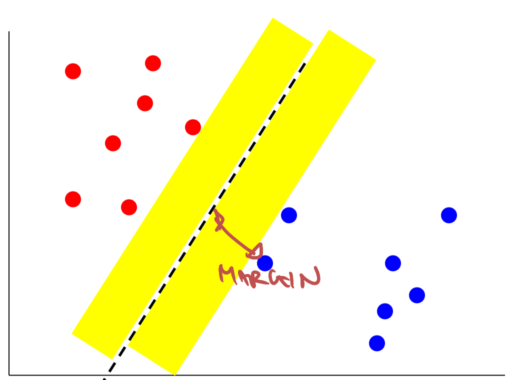
[partly based on slides of Ziv-Bar Joseph]

http://www.cs.cmu.edu/~ggordon/10601/

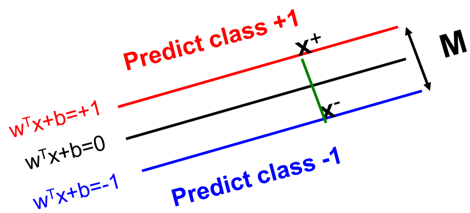
November 16, 2009

SVMs = max margin classifiers

- Instead of fitting all points, focus on the boundary
- Learn a boundary that leads to the largest margin from points on both sides



Finding the optimal parameters



$$M = \frac{2}{\sqrt{w^T w}}$$

We can now search for the optimal parameters by finding a solution that:

w, b

1. Correctly classifies all points
2. Maximizes the margin (or equivalently minimizes $w^T w$)

Several optimization methods can be used:
Gradient descent, simulated annealing, EM etc.

Quadratic programming (QP)

Quadratic programming solves optimization problems of the following form:

$$\min_{\mathbf{u}} \frac{\mathbf{u}^T \mathbf{R} \mathbf{u}}{2} + \mathbf{d}^T \mathbf{u} + c$$

Quadratic term

subject to n linear inequality constraints:

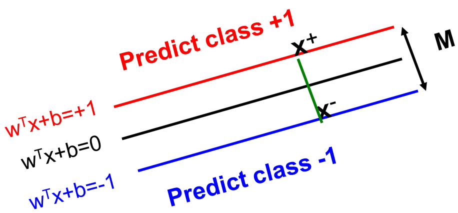
$$a_{11}u_1 + a_{12}u_2 + \ldots \leq b_1$$
$$\vdots \qquad \vdots \qquad \vdots$$
$$a_{n1}u_1 + a_{n2}u_2 + \ldots \leq b_n$$

When a problem can be specified as a QP problem we can use solvers that are better than gradient descent or simulated annealing

and k linear equality constraints:

$$a_{n+1,1}u_1 + a_{n+1,2}u_2 + \ldots = b_{n+1}$$
$$\vdots \qquad \vdots \qquad \vdots$$
$$a_{n+k,1}u_1 + a_{n+k,2}u_2 + \ldots = b_{n+k}$$

SVM as a QP problem



$$M = \frac{2}{\sqrt{w^T w}}$$

Min $(w^T w)/2$

subject to the following inequality constraints:

For all x in class + 1

$w^T x + b \geq 1$

For all x in class - 1

$w^T x + b \leq -1$

A total of n constraints if we have n input samples

$$\min_u \frac{u^T R u}{2} + d^T u + c$$

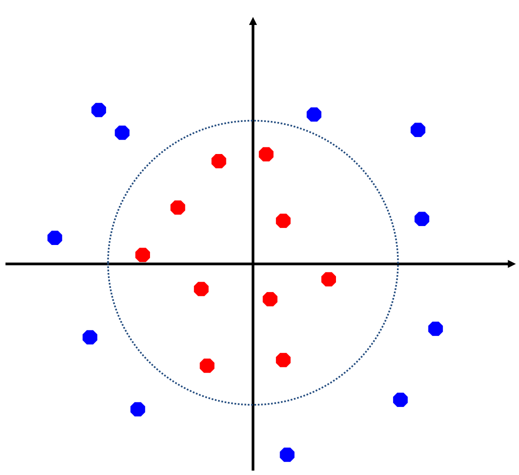
subject to n inequality constraints:

$$\begin{array}{l} a_{11}u_1 + a_{12}u_2 + \ldots \leq b_1 \\ \vdots \quad \vdots \quad \vdots \\ a_{n1}u_1 + a_{n2}u_2 + \ldots \leq b_n \end{array}$$

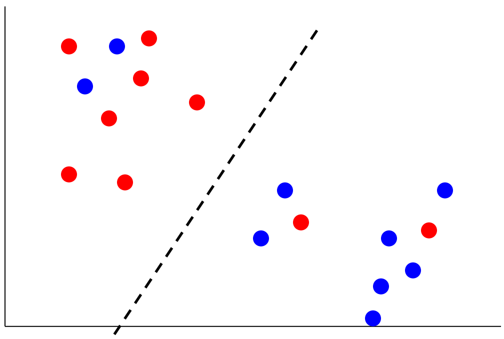
and k equivalency constraints:

$$\begin{array}{l} a_{n+1,1}u_1 + a_{n+1,2}u_2 + \ldots = b_{n+1} \\ \vdots \quad \vdots \quad \vdots \\ a_{n+k,1}u_1 + a_{n+k,2}u_2 + \ldots = b_{n+k} \end{array}$$

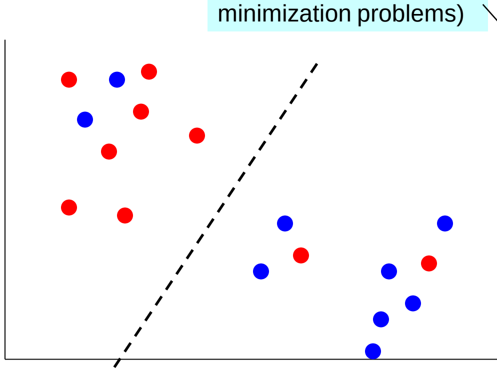
Non-separable case



Non-separable case #2



Non-separable case #2 due to **noise** and **outliers**



Hard to solve (two minimization problems)

How can we convert this to a QP problem?

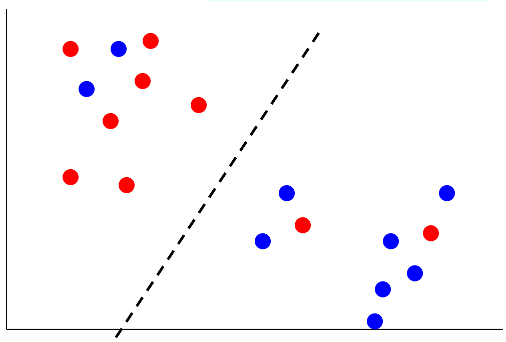
- Minimize training errors?

min w^Tw

min #errors

Non-separable case #2
due to noise and outliers

Hard to solve (two minimization problems)



How can we convert this to a QP problem?

- Minimize training errors?

  min w^Tw

  min #errors

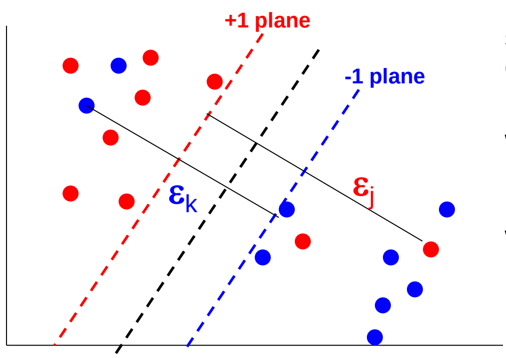
- Penalize training errors:

min w^Tw+C*(#errors)

nonconvex objective
NP-hard to optimize

Non-separable case #2

- Instead of minimizing the number of misclassified points we can minimize the *distance* between these points and their correct plane



The new optimization problem is:

$$\min_{\mathbf{w},b,\varepsilon_i} \frac{\mathbf{w}^T\mathbf{w}}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

subject to the following inequality constraints:

For all x_i in class + 1

$w^Tx+b \geq 1- \boldsymbol{\varepsilon}_i$

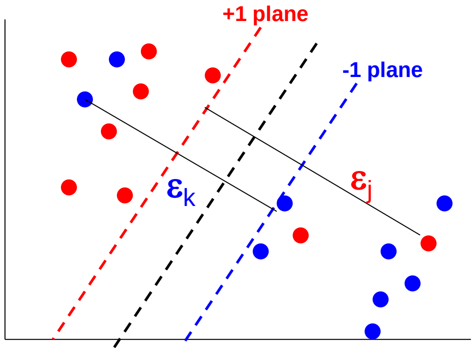
For all x_i in class - 1

$w^Tx+b \leq -1+ \boldsymbol{\varepsilon}_i$

Wait. Are we missing something?

Non-separable case #2

• Instead of minimizing the number of misclassified points we can minimize the *distance* between these points and their correct plane



The new optimization problem is:

$$\min_{\mathbf{w},b,\varepsilon_i} \frac{\mathbf{w}^T\mathbf{w}}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

subject to the following inequality constraints:

For all x_i in class + 1

$w^Tx+b \geq 1- \varepsilon_i$

For all x_i in class - 1

$w^Tx+b \leq -1+ \varepsilon_i$

For all i

$\varepsilon_i \geq 0$

Where we are

Two optimization problems: For the separable and non separable cases

$$\min_{\mathbf{w},b} \frac{\mathbf{w}^T\mathbf{w}}{2}$$

For all x in class + 1

$w^Tx+b \geq 1$

For all x in class - 1

$w^Tx+b \leq -1$

$$\min_{\mathbf{w},b,\varepsilon_i} \frac{\mathbf{w}^T\mathbf{w}}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

For all x_i in class + 1

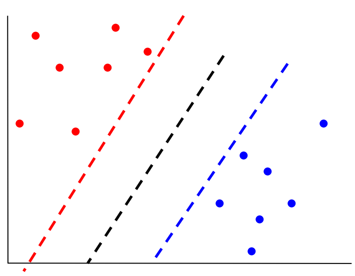
$w^Tx+b \geq 1- \varepsilon_i$

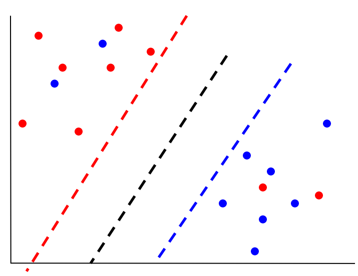
For all x_i in class - 1

$w^Tx+b \leq -1+ \varepsilon_i$

For all i

$\varepsilon_i \geq 0$





Non-separable case

$$\min_{\mathbf{w},b,\varepsilon_i} \frac{\mathbf{w}^T\mathbf{w}}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

For all x_i in class + 1

$w^T x + b \geq 1 - \boldsymbol{\varepsilon}_i$

For all x_i in class - 1

$w^T x + b \leq -1 + \boldsymbol{\varepsilon}_i$

For all i

$\boldsymbol{\varepsilon}_i \geq 0$

$\Rightarrow$

Why?

$$\min_{\mathbf{w},b,\varepsilon_i} \frac{\mathbf{w}^T\mathbf{w}}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

For all i

$(w^T x_i + b) y_i \geq 1 - \boldsymbol{\varepsilon}_i$

$\boldsymbol{\varepsilon}_i \geq 0$

Non-separable case: Hinge loss

$$\min_{\mathbf{w},b,\varepsilon_i} \frac{\mathbf{w}^T\mathbf{w}}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

$y_i f(x_i) \geq 1 - \boldsymbol{\varepsilon}_i$

$\boldsymbol{\varepsilon}_i \geq 0$

Hinge loss vs Log loss

Where we are

Two optimization problems: For the separable and non separable cases

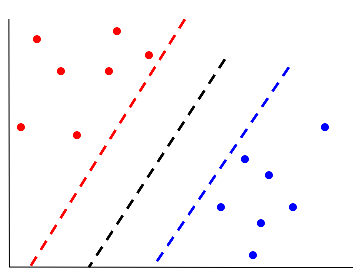
$$\min_{\mathbf{w},b} \frac{\mathbf{w}^T\mathbf{w}}{2}$$

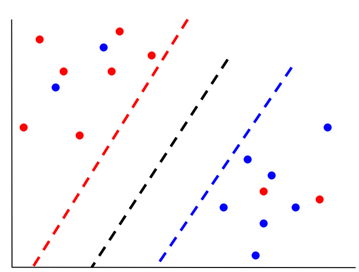
$(w^Tx_i+b)y_i \geq 1$

$$\min_{\mathbf{w},b,\varepsilon_i} \frac{\mathbf{w}^T\mathbf{w}}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

$(w^Tx_i+b)y_i \geq 1- \boldsymbol{\varepsilon}_i$

$\boldsymbol{\varepsilon}_i \geq 0$





Where we are

Two optimization problems: For the separable and non separable cases

$$\min_{\mathbf{w},b} \frac{\mathbf{w}^T\mathbf{w}}{2}$$

$(w^T x_i+b)y_i \geq 1$

$$\min_{\mathbf{w},b,\varepsilon_i} \frac{\mathbf{w}^T\mathbf{w}}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

$(w^T x_i+b)y_i \geq 1- \boldsymbol{\varepsilon}_i$

$\boldsymbol{\varepsilon}_i \geq 0$

- Instead of solving these QPs directly we will solve a dual formulation of the SVM optimization problem
- The main reason for switching to this type of representation is that it would allow us to use a neat trick that will make our lives easier (and the run time faster)

An alternative (dual) representation of the SVM QP

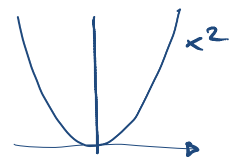
- We will start with the linearly separable case
- We will use Lagrange multipliers to derive an equivalent problem

Min $(w^T w)/2$

$(w^T x_i+b)y_i \geq 1$

Lagrange multipliers

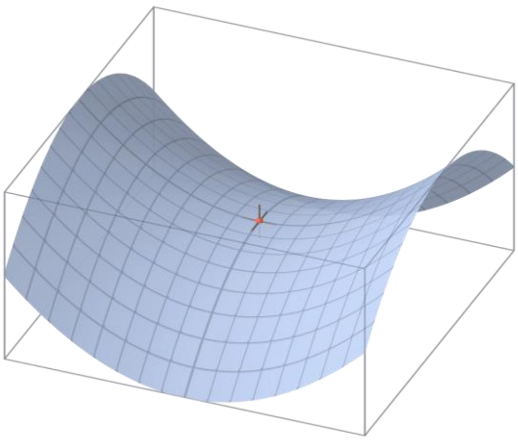
$$\min_x \; x^2$$
$$\text{s.t.} \quad x \geq b$$



Lagrange multipliers

$$\min_x \; x^2$$
$$\text{s.t.} \quad x \geq b$$
$$x - b \geq 0 \qquad (\alpha)$$
$$\|$$
$$\min_x \max_{\alpha \geq 0} \left[x^2 - (x-b)\alpha \right]$$

Lagrange multipliers: saddle-point solution



$\min_{x} \max_{\alpha \geq 0} L(x,\alpha)$

$\max_{\alpha \geq 0} \min_{x} L(x,\alpha)$

Lagrange multipliers for SVMs

Dual formulation

$$\max_{\alpha_i} \min_{\mathbf{w},b} \frac{\mathbf{w}^{\mathsf{T}}\mathbf{w}}{2} - \sum_i \alpha_i[(\mathbf{w}^{\mathsf{T}}\mathbf{x}_i + b)y_i - 1]$$

$\alpha_i \geq 0 \qquad \forall i$

Original formulation

Min $(w^T w)/2$

$(w^T x_i + b) y_i \geq 1$

Using this new formulation we can derive the best action for minimizer, by taking the derivative w.r.t. w and b leading to:

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

$$\sum_i \alpha_i y_i = 0$$

Dual SVM for linearly separable case

Substituting w into our target function and using the additional constraint we get:

Dual formulation

$$\max_{\alpha_i} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x_i x_j}$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

FROM PREVIOUS SLIDE:

$$\max_{\alpha_i} \min_{\mathbf{w},b} \frac{\mathbf{w}^\mathsf{T}\mathbf{w}}{2} - \sum_i \alpha_i [(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b) y_i - 1]$$

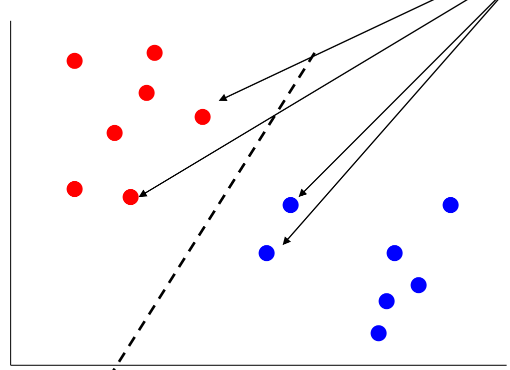
$$\alpha_i \geq 0 \qquad \forall i$$

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

$$\sum_i \alpha_i y_i = 0$$

Dual SVM - interpretation

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



For α's that are not 0

$$\sum_i \alpha_i y_i = 0; \quad \alpha_i \geq 0$$

Dual SVM for linearly separable case

Our dual target function: $\max_{\alpha} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x_i} \mathbf{x_j}$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

Dot product for all training samples

Dot product with training samples

To evaluate a new sample **x** we need to compute:

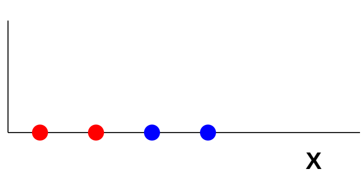
$$\mathbf{w}^T \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x}_i \mathbf{x} + b$$

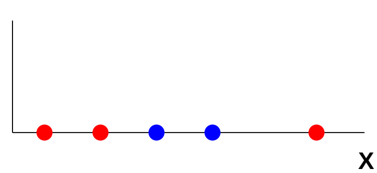
Is this too much computational work (for example when using transformation of the data)?

Classifying in 1-d

Can an SVM correctly classify this data?

What about this?

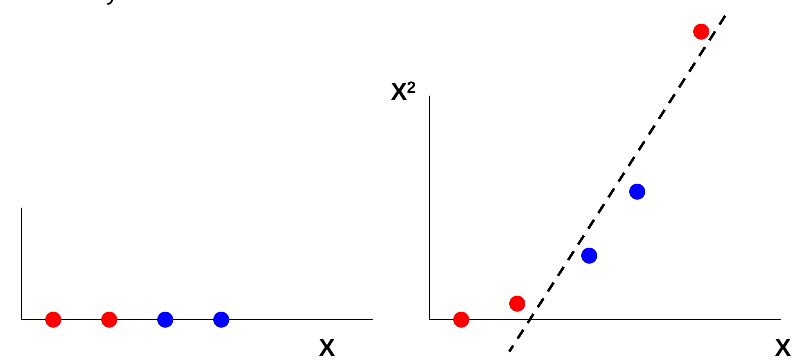




Classifying in 1-d

Can an SVM correctly classify this data?

And now?

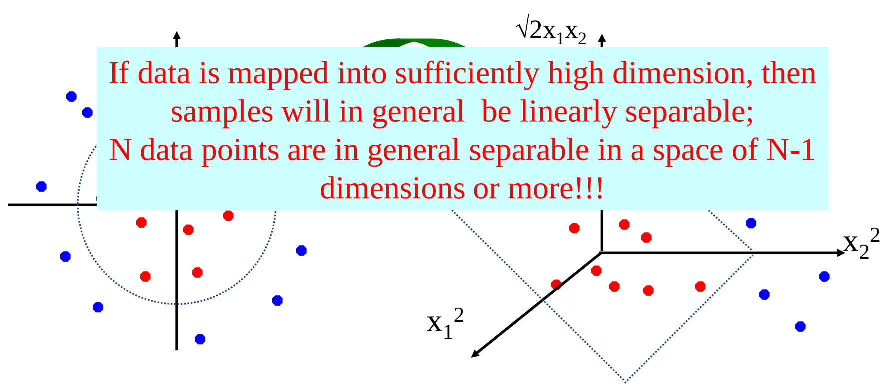


Non-linear SVDs in 2-d

- The original input space (**x**) can be mapped to some higher-dimensional feature space (φ(**x**))where the training set is separable:

$\mathbf{x}=(x_1,x_2)$

$\varphi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$

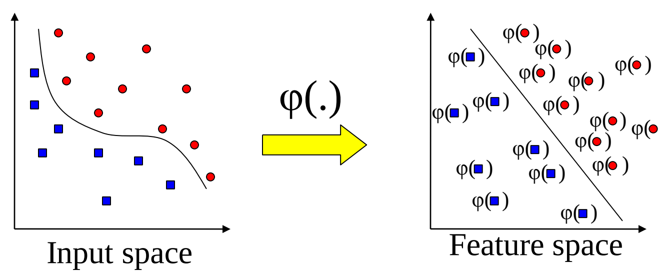


If data is mapped into sufficiently high dimension, then samples will in general be linearly separable;
N data points are in general separable in a space of N-1 dimensions or more!!!

This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

Transformation of Inputs

- Possible problems
  - High computation burden due to high-dimensionality
  - Many more parameters
- SVM solves these two issues simultaneously
  - "Kernel tricks" for efficient computation
  - Dual formulation only assigns parameters to samples, not features



Polynomials of degree two

- While working in higher dimensions is beneficial, it also increases our running time because of the dot product computation
- However, there is a neat trick we can use
- consider all quadratic terms for $x_1, x_2 \dots x_m$

$$\max_\alpha \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j y_i y_j \varphi(\mathbf{x_i})\varphi(\mathbf{x_j})$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

m is the number of features in each vector

The √2 term will become clear in the next slide

$$\varphi(x) = \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \vdots \\ \sqrt{2}x_m \\ x_1^2 \\ \vdots \\ x_m^2 \\ \sqrt{2}x_1x_2 \\ \vdots \\ \sqrt{2}x_{m-1}x_m \end{pmatrix}$$

m+1 linear terms

m quadratic terms

m(m-1)/2 pairwise terms

Polynomials of degree d in m variables

Polynomials of degree d in m variables

Original formulation

Min $(w^T w)/2$

$(w^T \varphi(x_i)+b)y_i \geq 1$

Dual formulation

$$\max_\alpha \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j y_i y_j \varphi(\mathbf{x_i})\varphi(\mathbf{x_j})$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

Dot product for polynomials of degree two

How many operations do we need for the dot product?

$$\varphi(x)\varphi(z) = \begin{bmatrix} 1 \\ \sqrt{2}x_1 \\ \vdots \\ \sqrt{2}x_m \\ x_1^2 \\ \vdots \\ x_m^2 \\ \sqrt{2}x_1x_2 \\ \vdots \\ \sqrt{2}x_{m-1}x_m \end{bmatrix} \bullet \begin{bmatrix} 1 \\ \sqrt{2}z_1 \\ \vdots \\ \sqrt{2}z_m \\ z_1^2 \\ \vdots \\ z_m^2 \\ \sqrt{2}z_1z_2 \\ \vdots \\ \sqrt{2}z_{m-1}z_m \end{bmatrix} = \sum_i 2x_iz_i + \sum_i x_i^2z_i^2 + \sum_i \sum_{j=i+1} 2x_ix_jz_iz_j + 1$$

m m m(m-1)/2 **=~ m²**

The kernel trick

How many operations do we need for the dot product?

$$= \sum_i 2x_iz_i + \sum_i x_i^2z_i^2 + \sum_i \sum_{j=i+1} 2x_ix_jz_iz_j + 1$$

m m m(m-1)/2 **=~ m²**

However, we can obtain dramatic savings by noting that

$$\begin{aligned} (x.z+1)^2 &= (x.z)^2 + 2(x.z) + 1 \\ &= (\sum_i x_iz_i)^2 + \sum_i 2x_iz_i + 1 \\ &= \sum_i 2x_iz_i + \sum_i x_i^2z_i^2 + \sum_i \sum_{j=i+1} 2x_ix_jz_iz_j + 1 \end{aligned}$$

We only need m operations!

Note that to evaluate a new sample we are also using dot products so we save there as well

Where we are

Our dual target function:

$$\max_{\alpha_i} \sum_i \alpha_i - \frac{1}{2}\sum_{\mathrm{i,j}} \alpha_i \alpha_j y_i y_j \varphi(\mathbf{x}_i)\varphi(\mathbf{x}_j)$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

mn^2 operations at each iteration

To evaluate a new sample **x** we need to compute:

$$\mathbf{w}^T \varphi(\mathbf{x}) + b = \sum_i \alpha_i y_i \varphi(\mathbf{x}_i)\varphi(\mathbf{x}) + b$$

mr operations where *r* is the number of support vectors (α_i>0)

Other kernels

•Beyond polynomials there are other very high dimensional basis functions that can be made practical by finding the right k ernel function

- Radial-Basis Function:

$$K(\mathbf{x},\mathbf{z}) = \exp\left(-\frac{(\mathbf{x}-\mathbf{z})^2}{2\sigma^2}\right)$$

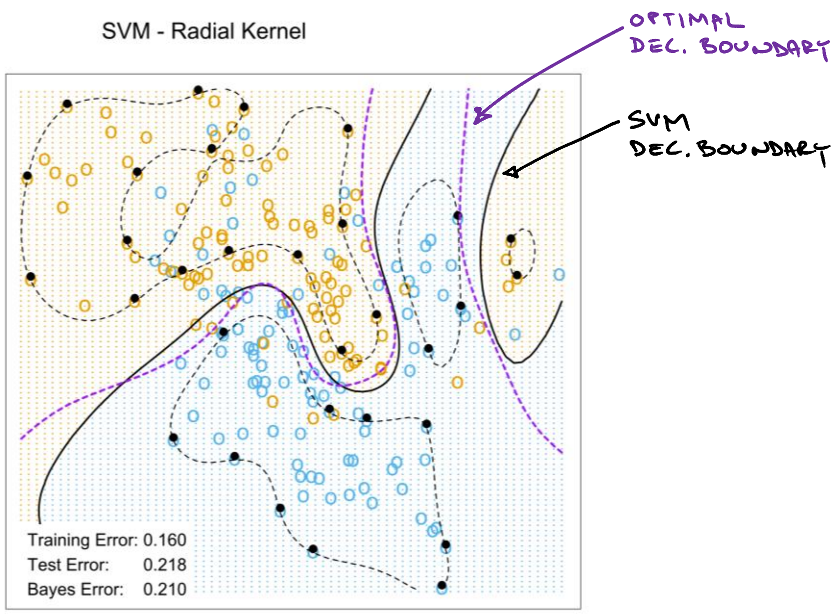
- kernel functions for discrete objects (graphs, strings, etc.)

Kernels measure similarity

$$K(\mathbf{x},\mathbf{z}) = \exp\left(-\frac{(\mathbf{x}-\mathbf{z})^2}{2\sigma^2}\right)$$

Decision rule for a new sample **x:**

$$\mathbf{w}^T\,\varphi(\mathbf{x}) + b = \sum_i \alpha_i y_i K(\mathbf{x}_i,\mathbf{x}) + b$$



This slide is courtesy of Hastie-Tibshirani-Friedman, 2nd ed.

Dual formulation for non-separable case

Dual target function:

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y_i} y_j \mathbf{x_i x_j}$$

$$\sum_i \alpha_i \mathrm{y_i} = 0$$

$$C > \alpha_i \geq 0 \qquad \forall i$$

The only difference is that the α_I's are now bounded

To evaluate a new sample **x** we need to compute:

$$\mathbf{w}^{\mathrm{T}} \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x_i} \mathbf{x} + b$$

Why do SVMs work?

• If we are using huge features spaces (with kernels) how come we are not overfitting the data?

- We maximize margin!

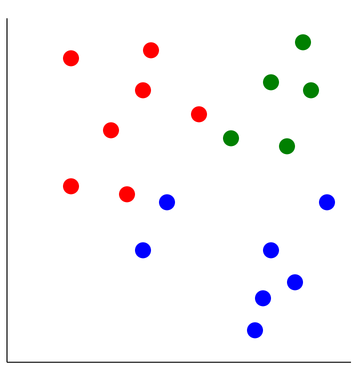
- We minimize loss + regularization

Software

- A list of SVM implementation can be found at http://www.kernel-machines.org/software.html
- Some implementation (such as LIBSVM) can handle multi-class classification
- SVMLight is among one of the earliest implementation of SVM
- Several Matlab toolboxes for SVM are also available

Multi-class classification with SVMs

What if we have data from more than two classes?



• Most common solution: One vs. all

- create a classifier for each class against all other data

- for a new point use all classifiers and compare the margin for all selected classes

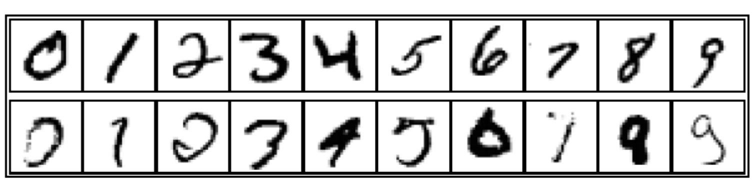
Note that this is not necessarily valid since this is not what we trained the SVM for, but often works well in practice

Applications of SVMs

- Bioinformatics
- Machine Vision
- Text Categorization
- Ranking (e.g., Google searches)
- Handwritten Character Recognition
- Time series analysis

→Lots of very successful applications!!!

Handwritten digit recognition



3-nearest-neighbor = 2.4% error
400–300–10 unit MLP = 1.6% error
LeNet: 768–192–30–10 unit MLP = 0.9% error

Current best (kernel machines, vision algorithms) ≈ 0.6% error

Important points

- Difference between regression classifiers and SVMs'
- Maximum margin principle
- Target function for SVMs
- Linearly separable and non separable cases
- Dual formulation of SVMs
- Kernel trick and computational complexity