UVA CS 4501 - 001 / 6501 – 007 Introduction to Machine Learning and Data Mining

Lecture 10: Classification with Support Vector Machine (cont. 2)

Yanjun Qi / Jane

University of Virginia
Department of
Computer Science

Where we are ? ➔ Five major sections of this course

- ❑ Regression (supervised)
- ❑ Classification (supervised)
- ❑ Unsupervised models
- ❑ Learning theory
- ❑ Graphical models

Where we are ? → Three major sections for classification

- We can divide the large variety of classification approaches into roughly three major types

1. Discriminative
   - directly estimate a decision rule/boundary
   - e.g., support vector machine, decision tree

2. Generative:
   - build a generative statistical model
   - e.g., Bayesian networks

3. Instance based classifiers
   - Use observation directly (no models)
   - e.g. K nearest neighbors

Today

- Support Vector Machine (SVM)

Last Lecture
- ✓ History of SVM
- ✓ Large Margin Linear Classifier
- ✓ Define Margin (M) in terms of model parameter
- ✓ Optimization to learn model parameters (w, b)
- ✓ Non linearly separable case

- ✓ Optimization with dual form
- ✓ Nonlinear decision boundary
- ✓ Multiclass SVM

Today

- ❑ Support Vector Machine (SVM)
  - review ✓ History of SVM
  - ✓ Large Margin Linear Classifier
  - ✓ Define Margin (M) in terms of model parameter
  - ✓ Optimization to learn model parameters (w, b)
  - ✓ Non linearly separable case
  - ✓ Optimization with dual form
  - ✓ Nonlinear decision boundary
  - ✓ Multiclass SVM

History of SVM

Young / theoretically sound / Impactful

- SVM is inspired from statistical learning theory [3]
- SVM was first introduced in 1992 [1]
- SVM becomes popular because of its success in handwritten digit recognition
  - 1.1% test error rate for SVM. This is the same as the error rates of a carefully constructed neural network, LeNet 4.
    - See Section 5.11 in [2] or the discussion in [3] for details
- SVM is now regarded as an important example of "kernel methods", arguably the hottest area in machine learning 10 years ago

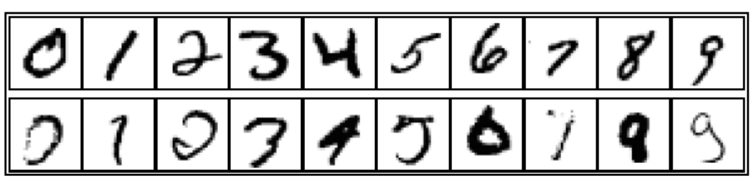
[1] B.E. Boser *et al.* A Training Algorithm for Optimal Margin Classifiers. Proceedings of the Fifth Annual Workshop on Computational Learning Theory 5 144-152, Pittsburgh, 1992.
[2] L. Bottou *et al.* Comparison of classifier methods: a case study in handwritten digit recognition. Proceedings of the 12th IAPR International Conference on Pattern Recognition, vol. 2, pp. 77-82, 1994.
[3] V. Vapnik. The Nature of Statistical Learning Theory. 2nd edition, Springer, 1999.

Applications of SVMs

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

→Lots of very successful applications!!!

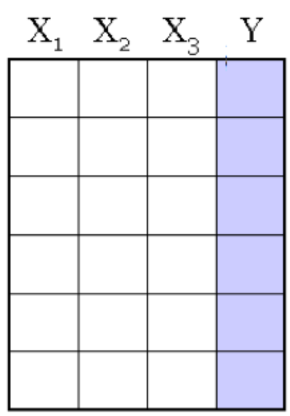
Handwritten digit recognition



3-nearest-neighbor = 2.4% error
400–300–10 unit MLP = 1.6% error
LeNet: 768–192–30–10 unit MLP = 0.9% error
1999, SVM → best (kernel machines, vision algorithms) ≈ 0.6% error

Today

- ❑ Support Vector Machine (SVM)
  - ✓ History of SVM
  - review ✓ Large Margin Linear Classifier
  - ✓ Define Margin (M) in terms of model parameter
  - ✓ Optimization to learn model parameters (w, b)
  - ✓ Non linearly separable case
  - ✓ Optimization with dual form
  - ✓ Nonlinear decision boundary



A Dataset for binary classification

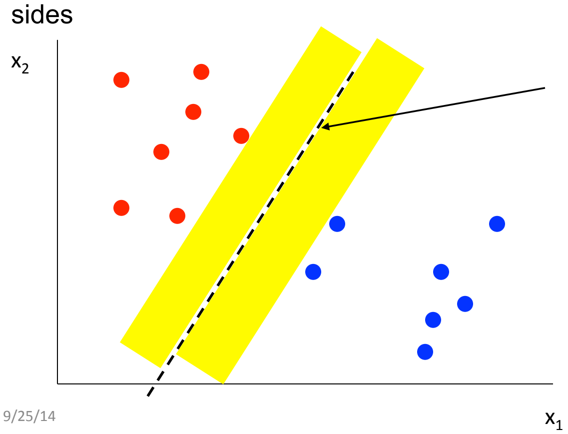
$$f : \boxed{X} \longrightarrow \boxed{Y}$$

Output as Binary Class Label: 1 or -1

- **Data**/*points/instances/examples/samples/records*: [rows]
- **Features**/*attributes/dimensions/independent variables/covariates/predictors/regressors*: [columns, except the last]
- **Target**/*outcome/response/label/dependent variable*: special column to be predicted [last column]

Max margin classifiers

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

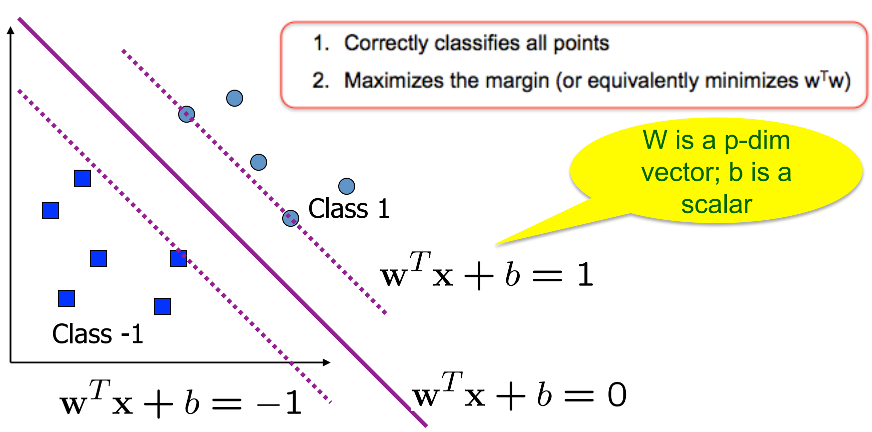


Why?

- Intuitive, 'makes sense'
- Some theoretical support
- Works well in practice

Max-margin & Decision Boundary

- The decision boundary should be as far away from the data of both classes as possible



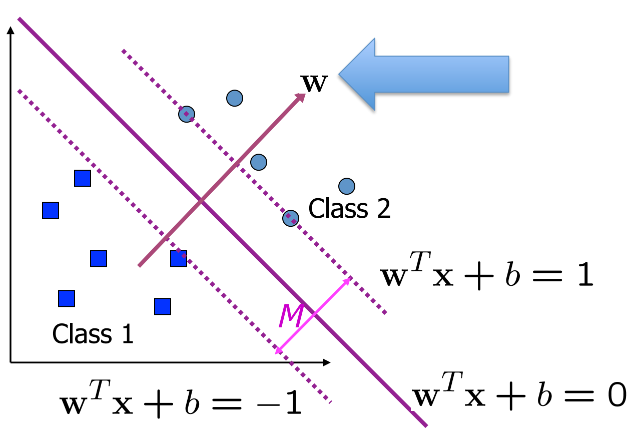
Today

- Support Vector Machine (SVM)
  - ✓ History of SVM
  - ✓ Large Margin Linear Classifier
  - ✓ Define Margin (M) in terms of model parameter
  - ✓ Optimization to learn model parameters (w, b)
  - ✓ Non linearly separable case
  - ✓ Optimization with dual form
  - ✓ Nonlinear decision boundary
  - ✓ Multiclass SVM

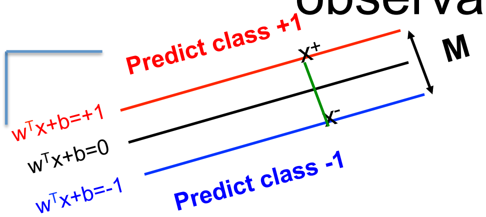
review

Maximizing the margin: observation-1

- Observation 1: the vector w is orthogonal to the +1 plane



Maximizing the margin: observation-2



Classify as +1 if $w^Tx+b \geq 1$

Classify as -1 if $w^Tx+b \leq -1$

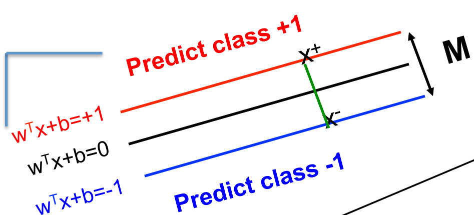
Undefined if $-1 < w^Tx+b < 1$

- Observation 1: the vector w is orthogonal to the +1 and -1 planes
- Observation 2: if x^+ is a point on the +1 plane and x^- is the closest point to x^+ on the -1 plane then

$$x^+ = \lambda w + x^-$$

Since w is orthogonal to both planes we need to 'travel' some distance along w to get from x^+ to x^-

Putting it together



- $w^T x^+ + b = +1$
- $w^T x^- + b = -1$
- $x^+ = \lambda w + x^-$
- $| x^+ - x^- | = M$

$w^T x^+ + b = +1$

$\Rightarrow$

$w^T (\lambda w + x^-) + b = +1$

$\Rightarrow$

$w^Tx^- + b + \lambda w^Tw = +1$

$\Rightarrow$

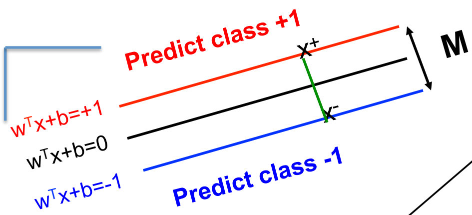
$-1 + \lambda w^Tw = +1$

$\Rightarrow$

$\lambda = 2/w^Tw$

We can now define M in terms of w and b

Putting it together



- $w^T x^+ + b = +1$
- $w^T x^- + b = -1$
- $x^+ = \lambda w + x^-$
- $| x^+ - x^- | = M$
- $\lambda = 2/w^Tw$

$$M = |x^+ - x^-|$$

$$\Rightarrow$$

$$M = |\lambda w| = \lambda |w| = \lambda\sqrt{w^T w}$$

$$\Rightarrow$$

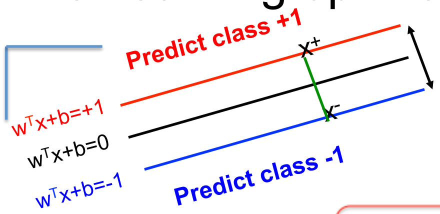
$$M = 2\frac{\sqrt{w^T w}}{w^T w} = \frac{2}{\sqrt{w^T w}}$$

We can now define M in terms of w and b

Today

- ❑ Support Vector Machine (SVM)
  - ✓ History of SVM
  - ✓ Large Margin Linear Classifier
  - ✓ Define Margin (M) in terms of model parameter
  - ✓ Optimization to learn model parameters (w, b)
  - review ✓ Non linearly separable case
  - ✓ Optimization with dual form
  - ✓ Nonlinear decision boundary
  - ✓ Multiclass SVM

Optimization Step
i.e. learning optimal parameter for SVM



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

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

Min $(w^Tw)/2$

subject to the following constraints:

For all x in class + 1

$w^Tx+b \geq 1$

For all x in class - 1

$w^Tx+b \leq -1$

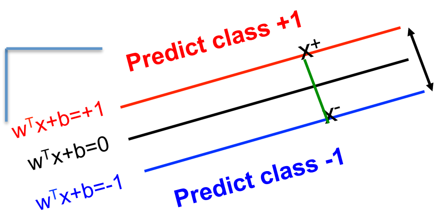
A total of n constraints if we have n input samples

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{p} w_i^2$$

$$\text{subject to } \forall \mathbf{x}_i \in Dtrain : y_i\left(\mathbf{x}_i \cdot \mathbf{w} + b\right) \geq 1$$

SVM as a QP problem

R as I matrix, d as zero vector, c as 0 value



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

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

subject to n inequality constraints:

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

and k equivalency constraints:

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

Min $(w^Tw)/2$

subject to the following inequality constraints:

For all x in class + 1

$w^Tx+b \geq 1$

For all x in class - 1

$w^Tx+b \leq -1$

A total of n constraints if we have n input samples

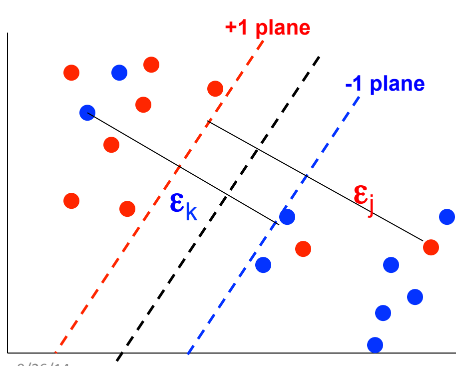
Today

- ❑ Support Vector Machine (SVM)
  - ✓ History of SVM
  - ✓ Large Margin Linear Classifier
  - ✓ Define Margin (M) in terms of model parameter
  - ✓ Optimization to learn model parameters (w, b)
  - ✓ Non linearly separable case
  - ✓ Optimization with dual form
  - ✓ Nonlinear decision boundary
  - ✓ Multiclass SVM

review

Non linearly separable case

- 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:

$$\min_w \frac{w^T w}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

subject to the following inequality constraints:

For all x_i in class + 1

$w^T x + b \geq 1 - \varepsilon_i$

For all x_i in class - 1

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

A total of n constraints

For all i

$\varepsilon_i \geq 0$

Another n constraints

Where we are

Two optimization problems: For the separable and non separable cases

$$\min_w \frac{w^T w}{2}$$

For all x in class + 1

$w^T x + b \geq 1$

For all x in class - 1

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

$$\min_w \frac{w^T w}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

For all x_i in class + 1

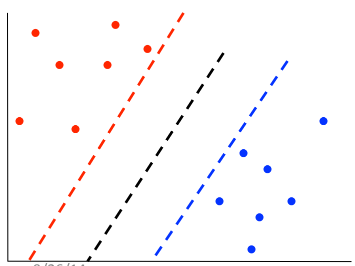
$w^T x + b \geq 1 - \varepsilon_i$

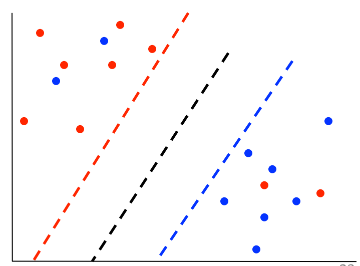
For all x_i in class - 1

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

For all i

$\varepsilon_i \geq 0$





Today

- ❑ Support Vector Machine (SVM)
  - ✓ History of SVM
  - ✓ Large Margin Linear Classifier
  - ✓ Define Margin (M) in terms of model parameter
  - ✓ Optimization to learn model parameters (w, b)
  - ✓ Non linearly separable case
  - ✓ Optimization with dual form
  - ✓ Nonlinear decision boundary
  - ✓ Multiclass SVM

Where we are

Two optimization problems: For the separable and non separable cases

Min $(w^T w)/2$

For all x in class + 1

$w^T x + b \geq 1$

For all x in class - 1

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

$$\min_w \frac{w^T w}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

For all x_i in class + 1

$w^T x + b \geq 1 - \varepsilon_i$

For all x_i in class - 1

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

For all i

$\varepsilon_i \geq 0$

- 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)

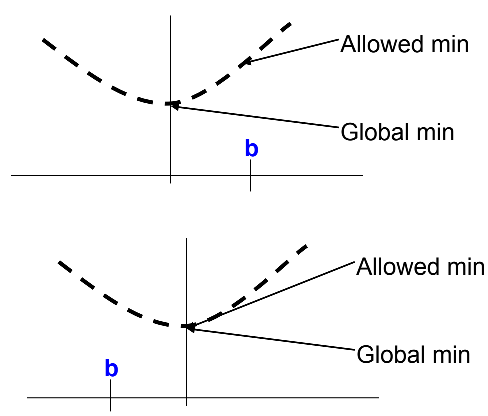
Optimization Review: Constrained Optimization

$\min_u u^2$

s.t. $u \geq b$

Case 1:

Case 2:



Optimization Review: Constrained Optimization with Lagrange

- When equal constraints
- ➔ optimize *f(x),* subject to $g_i(x)=0$

- Method of Lagrange multipliers: convert to a higher-dimensional problem
- Minimize

$$f(x)+\sum \lambda_i g_i(x)$$

- w.r.t. $(x_1 \ldots x_n ; \lambda_1 \ldots \lambda_k)$

Introducing a Lagrange multiplier for each constraint
Construct the Lagrangian for the original optimization problem

27

Optimization Review: Dual Problem

- Using dual problem
  - Constrained optimization → unconstrained optimization
- Need to change maximization to minimization
- Only valid when the original optimization problem is convex/concave (strong duality)

Dual Problem

$$\lambda^* = \arg\min_{\lambda} l(\lambda)$$

Primal Problem

$$x^* = \arg\max_{x} f(x)$$

subject to $g(x)=c$

$x^*=\lambda^*$
When convex/concave

$$l(\lambda)=\sup_{x}(f(x)+\lambda(g(x)-c))$$

An alternative (dual) representation of the SVM QP

- We will start with the linearly separable case
- Instead of encoding the correct classification rule and constraint we will use LaGrange multiples to encode it as part of the our minimization problem

Min $(w^Tw)/2$

For all x in class +1

$w^Tx+b \geq 1$

For all x in class -1

$w^Tx+b \leq -1$

Why? $\Downarrow$

Min $(w^Tw)/2$

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

An alternative (dual) representation of the SVM QP

Min $(w^Tw)/2$

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

- We will start with the linearly separable case
- Instead of encoding the correct classification rule a constraint we will use Lagrange multiples to encode it as part of the our minimization problem

Recall that Lagrange multipliers can be applied to turn the following problem:

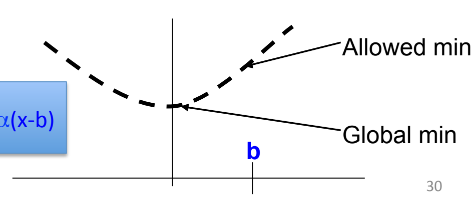
$\min_x x^2$

s.t. $x \geq b$ → $b-x \leq 0$

To

$\text{Min}_{x,\alpha}\ x^2 + \alpha(b-x)$ ← $\min_x \max_\alpha x^2 - \alpha(x-b)$

s.t. $\alpha \geq 0$



Lagrange multiplier for SVMs

Dual formulation

$$\min_{w,b} \max_{\alpha} \frac{w^T w}{2} - \sum_i \alpha_i [(w^T x_i + b) y_i - 1]$$

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

Original formulation

Min $(w^T w)/2$

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

Using this new formulation we can derive w and b by taking the derivative w.r.t. w and α leading to:

$$w = \sum_i \alpha_i x_i y_i$$

$$b = y_i - w^T x_i$$

$$for \quad i \quad s.t. \quad \alpha_i > 0$$

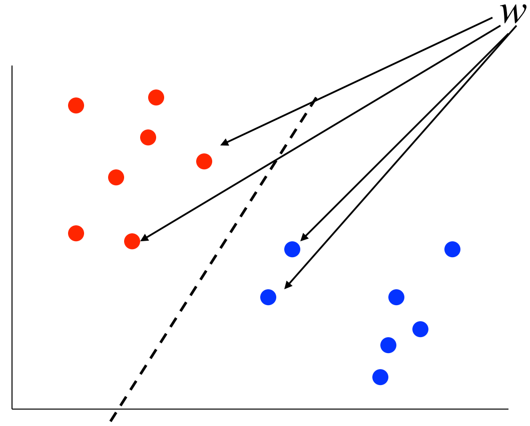
Set partial derivatives to 0

Finally, taking the derivative w.r.t. b we get:

$$\sum_i \alpha_i y_i = 0$$

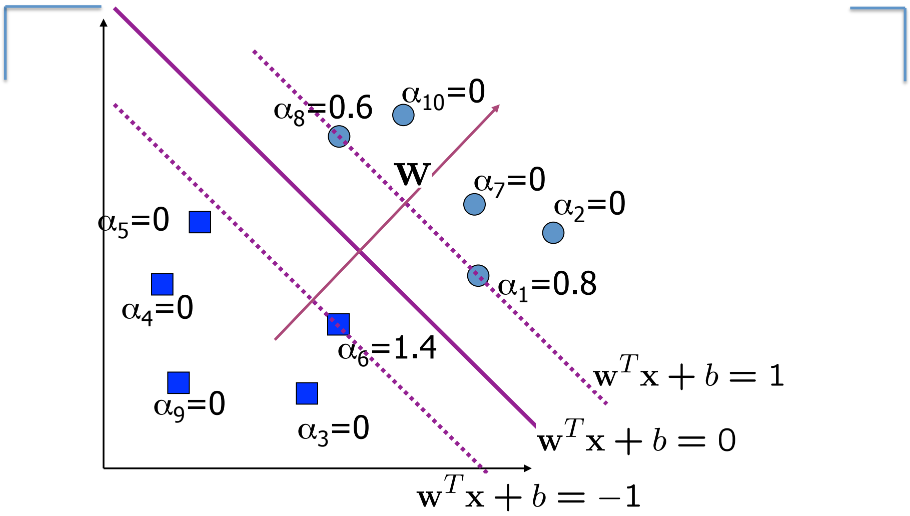
Dual SVM - interpretation

$$w = \sum_i \alpha_i x_i y_i$$



For α's that are not 0, no influence

A Geometrical Interpretation



Dual SVM for linearly separable case

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

Dual formulation

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

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

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

$$\min_{w,b} \frac{\mathrm{w^T w}}{2} - \sum_i \alpha_i [(\mathrm{w^T} x_i + b) y_i - 1]$$

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

$$w = \sum_i \alpha_i x_i y_i$$

$$b = y_i - \mathrm{w^T} x_i$$

$$for \quad i \quad s.t. \quad \alpha_i > 0$$

$$\sum_i \alpha_i y_i = 0$$

Dual SVM for linearly separable case

Our dual target function: $\max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$

$$\sum_i \alpha_i y_i = 0$$

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

Dot product for all training samples

Dot product with training samples

To evaluate a new sample x_j we need to compute:

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

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

Dual formulation for non linearly separable case

Dual target function:

$$\max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\sum_i \alpha_i y_i = 0$$

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

Hyperparameter C should be tuned through k-folds CV

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

To evaluate a new sample x_j we need to compute:

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

This is very similar to the optimization problem in the linear separable case, except that there is an upper bound *C* on α_i now

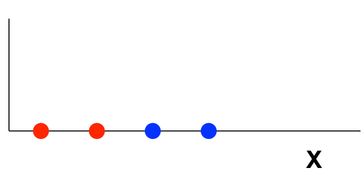
Once again, a QP solver can be used to find α_i

Today

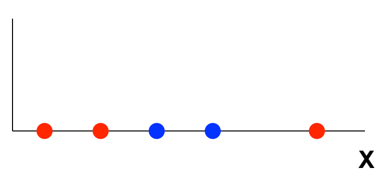
- ❑ Support Vector Machine (SVM)
  - ✓ History of SVM
  - ✓ Large Margin Linear Classifier
  - ✓ Define Margin (M) in terms of model parameter
  - ✓ Optimization to learn model parameters (w, b)
  - ✓ Non linearly separable case
  - ✓ Optimization with dual form
  - ✓ Nonlinear decision boundary
  - ✓ Multiclass SVM

Classifying in 1-d

Can an SVM correctly classify this data?



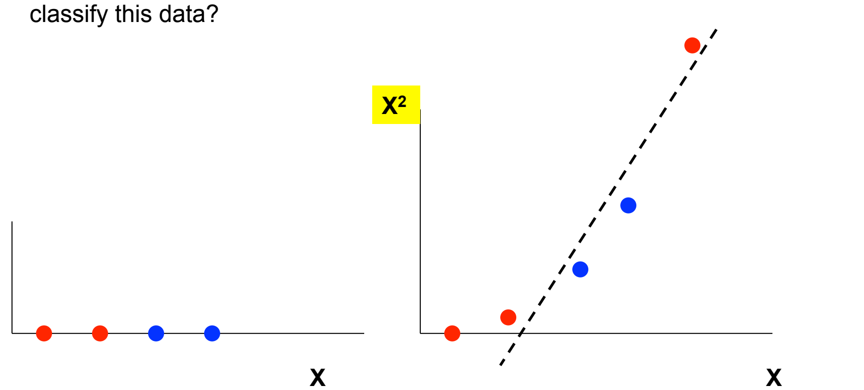
What about this?



Classifying in 1-d

Can an SVM correctly classify this data?

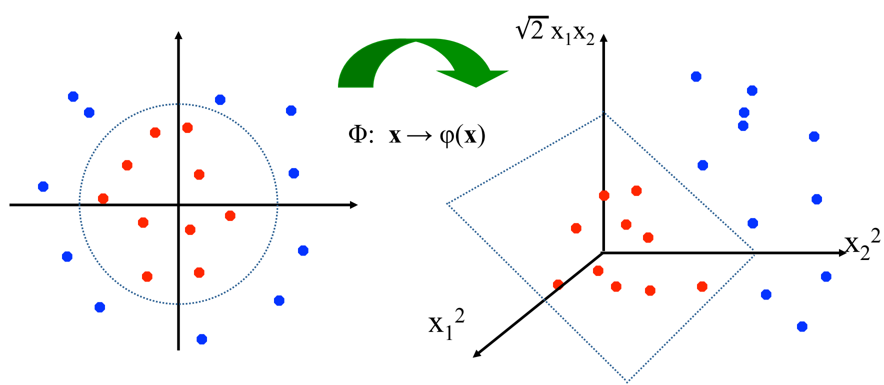
And now? (extend with polynomial basis)



Non-linear SVMs: 2D

- The original input space (x) can be mapped to some higher-dimensional feature space ($\phi(\mathbf{x})$)where the training set is separable:

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

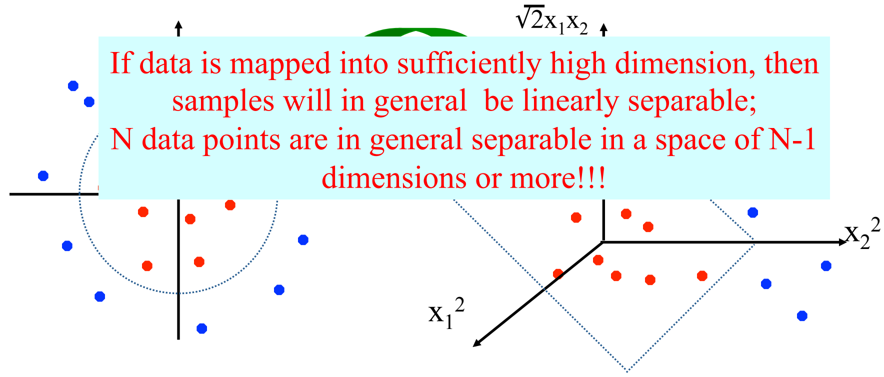


This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

Non-linear SVMs: 2D

- The original input space (x) can be mapped to some higher-dimensional feature space ($\phi(\mathbf{x})$)where the training set is separable:

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



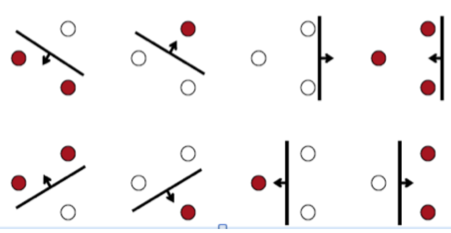
If data is mapped into sufficiently high dimension, then samples will in general be linearly separable;
N data points are in general separable in a space of N-1 dimensions or more!!!

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

A little bit theory: Vapnik-Chervonenkis (VC) dimension

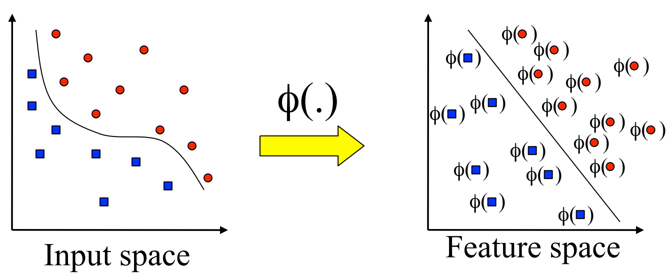
If data is mapped into sufficiently high dimension, then samples will in general be linearly separable;
N data points are in general separable in a space of N-1 dimensions or more!!!

- **VC dimension of the set of oriented lines in R^2 is 3**
  - It can be shown that the VC dimension of the family of oriented separating hyperplanes in R^N is at least N+1



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



Quadratic kernels

- 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 $x_1, x_2 \dots x_m$

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

$$\sum_i \alpha_i y_i = 0$$

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

m is the number of features in each vector

The √2 term will become clear in the next slide

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

m+1 linear terms

m quadratic terms

m(m-1)/2 pairwise terms

Dot product for quadratic kernels

How many operations do we need for the dot product?

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

m m m(m-1)/2 **=~ m²**

The kernel trick

How many operations do we need for the dot product?

$$\Phi(x)^T\Phi(z) = \sum_i 2x_iz_i + \sum_i x_i^2z_i^2 + \sum_i\sum_{j=i+1} 2x_ix_jz_iz_j + 1$$

m m m(m-1)/2 **=~ m²**

However, we can obtain dramatic savings by noting that

$$\Phi(x)^T\Phi(z) = (x^Tz+1)^2 = (x.z+1)^2 = (x.z)^2 + 2(x.z) + 1$$

$$= (\sum_i x_iz_i)^2 + \sum_i 2x_iz_i + 1$$

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

We only need m operations!

So, if we define the kernel function as follows, there is no need to carry out $\phi(.)$ explicitly

$$K(\mathbf{x}, z) = (x^Tz+1)^2$$

Where we are

Our dual target function:

$$\max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x_i})^T \Phi(\mathbf{x_j})$$

$$\sum_i \alpha_i y_i = 0$$

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

*mn*² operations at each iteration

To evaluate a new sample x_j we need to compute:

$$w^T\Phi(\mathbf{x_j}) + b = \sum_i \alpha_i y_i \Phi(\mathbf{x_i})^T \Phi(\mathbf{x_j}) + b$$

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

So, if we define the **kernel function** as follows, there is no need to carry out $\phi(.)$ explicitly

$$K(\mathbf{x}, z) = (x^T z + 1)^2$$

9/26/14 47

More examples of kernel functions

- Linear kernel (we've seen it) $K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T\mathbf{x}'$
- Polynomial kernel (we just saw an example)

$$K(\mathbf{x}, \mathbf{x}') = \left(1 + \mathbf{x}^T\mathbf{x}'\right)^p$$

where p = 2, 3, … To get the feature vectors we concatenate all pth order polynomial terms of the components of x (weighted appropriately)

- Radial basis kernel

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\left\|\mathbf{x} - \mathbf{x}'\right\|^2\right)$$

In this case the feature space consists of functions and results in a non-parametric classifier.

Never represent features explicitly
◆ Compute dot products in closed form
Very interesting theory – Reproducing Kernel Hilbert Spaces
☐ Not covered in detail here

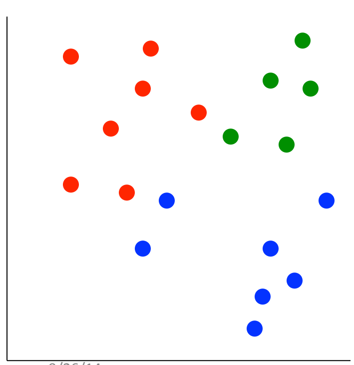
48

Today

- ❑ Support Vector Machine (SVM)
  - ✓ History of SVM
  - ✓ Large Margin Linear Classifier
  - ✓ Define Margin (M) in terms of model parameter
  - ✓ Optimization to learn model parameters (w, b)
  - ✓ Non linearly separable case
  - ✓ Optimization with dual form
  - ✓ Nonlinear decision boundary
  - ✓ Multiclass SVM

Multi-class classification with SVMs

What if we have data from more than two classes?



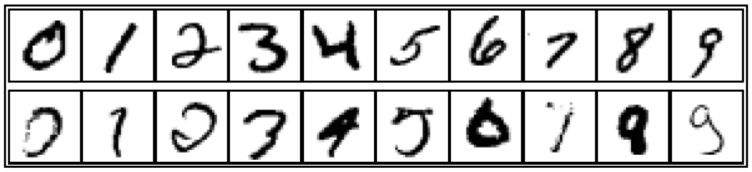
- Most common solution: One vs. all

- 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

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

1999, SVM

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

Why do SVMs work?

- If we are using huge features spaces (with kernels) how come we are not overfitting the data?
  - Number of parameters remains the same (and most are set to 0)
  - While we have a lot of input values, at the end we only care about the support vectors and these are usually a small group of samples
  - The minimization (or the maximizing of the margin) function acts as a sort of regularization term leading to reduced overfitting

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

References

- Big thanks to Prof. Ziv Bar-Joseph @ CMU for allowing me to reuse some of his slides
- Prof. Andrew Moore @ CMU's slides
- Elements of Statistical Learning, by Hastie, Tibshirani and Friedman