# UVA CS 6316 – Fall 2015 Graduate: Machine Learning

# Lecture 9: Support Vector Machine (Cont.)

Dr. Yanjun Qi

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

# A Dataset for binary classification

| $X_1$ | $X_2$ | $X_3$ | $Y$ |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

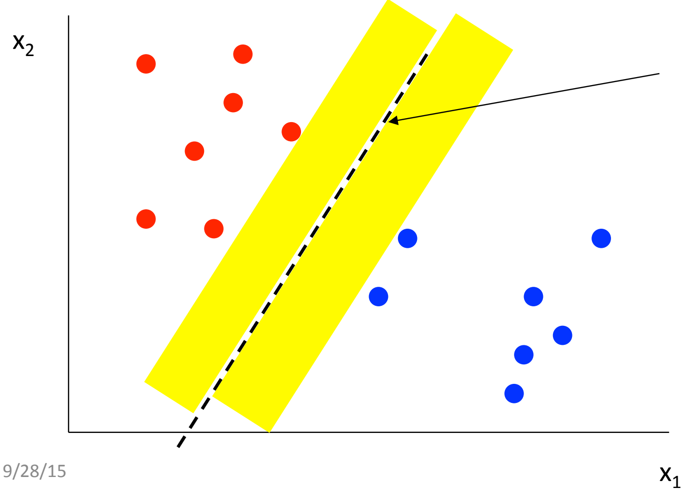
$$f : \boxed{X} \longrightarrow \boxed{Y}$$

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



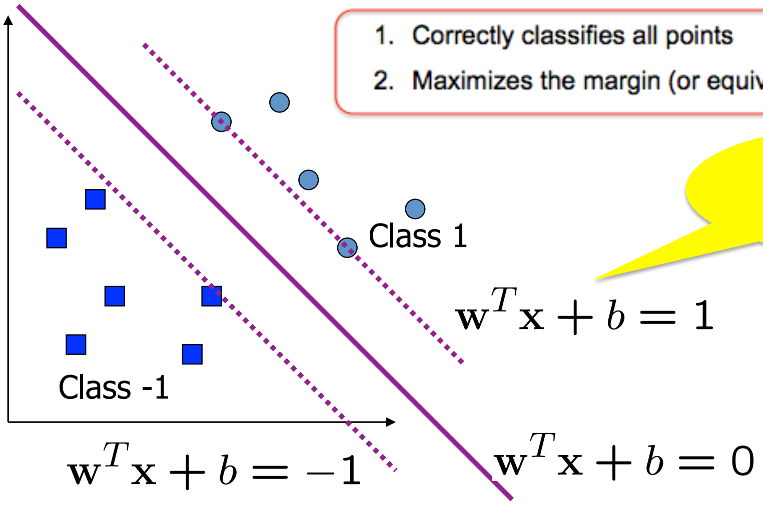
Why?

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

# When linearly Separable Case

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

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


Class 1
Class -1
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = -1$

W is a p-dim vector; b is a scalar

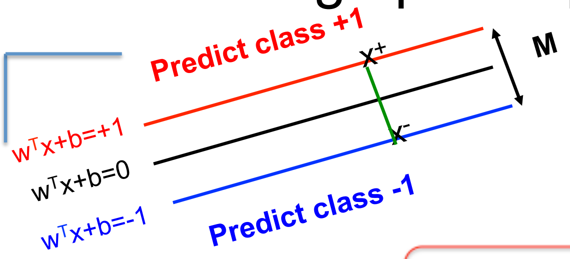
# Today

- □ 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
  - ✓ Practical Guide

# 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
  - ✓ Practical Guide

# 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

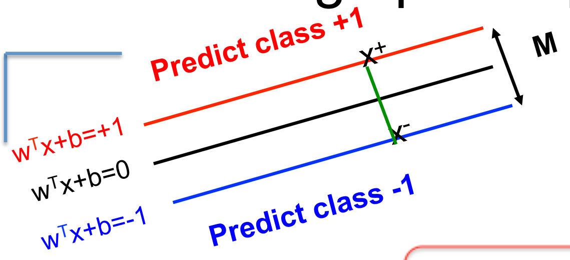
$w^Tx+b >= 1$ $y_i = 1$

For all x in class - 1

$w^Tx+b <= -1$ $y_i = -1$

A total of n constraints if we have n input samples

# 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

$w^Tx+b >= 1$

For all x in class - 1

$w^Tx+b <= -1$

A total of n constraints if we have n input samples

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

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

# Optimization Review: Ingredients

- Objective function
- Variables
- Constraints

**Find values of the variables that minimize or maximize the objective function while satisfying the constraints**

# Optimization with Quadratic programming (QP)

Quadratic programming solves optimization problems of the following form:

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

**Quadratic term**

subject to n inequality constraints:

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

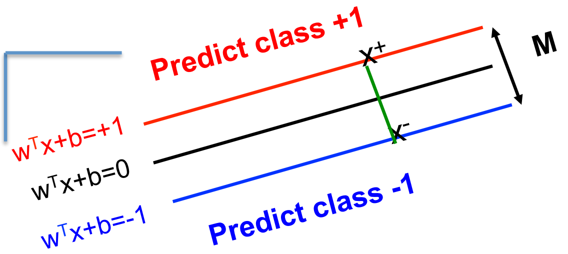
and k equivalency constraints:

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

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

# 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:

$$\begin{aligned} a_{11}u_1 + a_{12}u_2 + \ldots &\le b_1 \\ \vdots \qquad \vdots \qquad &\vdots \\ a_{n1}u_1 + a_{n2}u_2 + \ldots &\le b_n \end{aligned}$$

and k equivalency constraints:

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

Min $(w^Tw)/2$

subject to the following inequality constraints:

For all x in class + 1

$w^Tx+b >= 1$

For all x in class - 1

$w^Tx+b <= -1$

A total of n constraints if we have n input samples

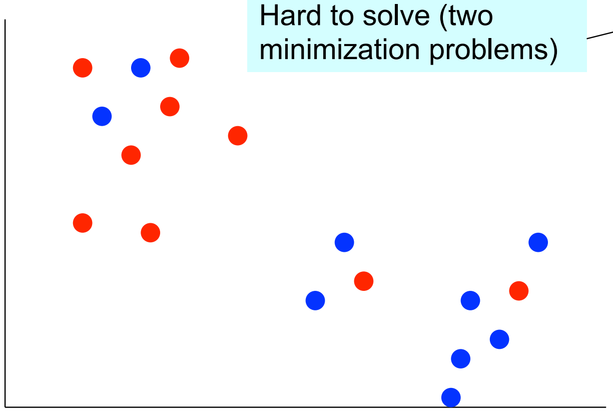
# Today

- ❑ 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
  - ✓ Practical Guide

# Non linearly separable case

- So far we assumed that a linear plane can perfectly separate the points
- But this is not usally the case

- noise, outliers



Hard to solve (two minimization problems)

How can we convert this to a QP problem?

- Minimize training errors?

min $w^Tw$

min #errors

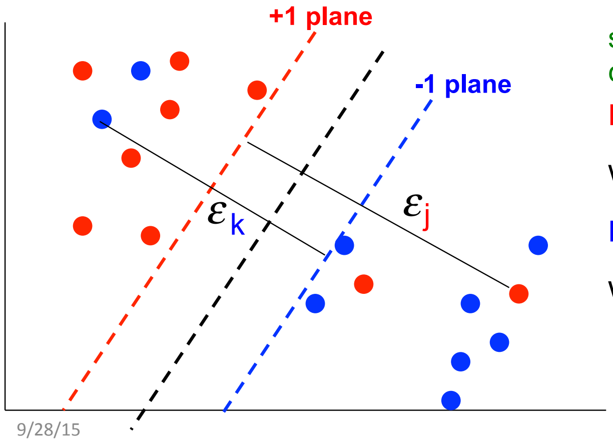
- Penalize training errors:

min $w^Tw$+C*(#errors)

Hard to encode in a QP problem

# 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

$w^Tx_i+b >= 1-\varepsilon_i$

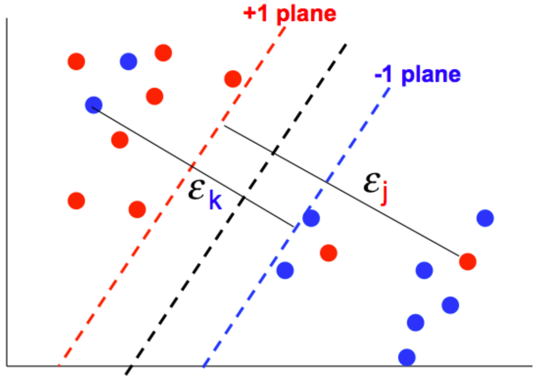
For all $x_i$ in class - 1

$w^Tx_i+b <= -1+\varepsilon_i$

$w: p$
$b: 1$
$\varepsilon_i: n$

Wait. Are we missing something?

# Final optimization for non linearly separable case



The new optimization problem is:

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

→ hyperparam

subject to the following inequality constraints:

For all $x_i$ in class + 1

$w^T x_i + b >= 1 - \varepsilon_i$

For all $x_i$ in class - 1

$w^T x_i + b <= -1 + \varepsilon_i$

A total of n constraints

n

For all i

$$\varepsilon_i \geq 0$$

Another n constraints

n

# 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

$w^T x + b >= 1$

For all x in class - 1

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

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

For all $x_i$ in class + 1

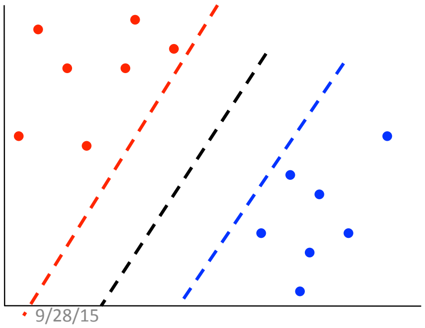
$w^T x_i + b >= 1 - \varepsilon_i$

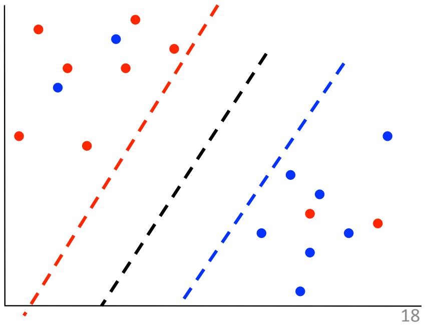
For all $x_i$ in class - 1

$w^T x_i + b <= -1 + \varepsilon_i$

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
  - ✓ Practical Guide

# Where we are

Two optimization problems: For the separable and non separable cases

Min $(w^T w)/2$

For all x in class + 1

$w^T x+b >= 1$

For all x in class - 1

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

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

For all $x_i$ in class + 1

$w^T x_i+b >= 1-\varepsilon_i$

For all $x_i$ in class - 1

$w^T x_i+b <= -1+\varepsilon_i$

For all i

$$\varepsilon_i \geq 0$$

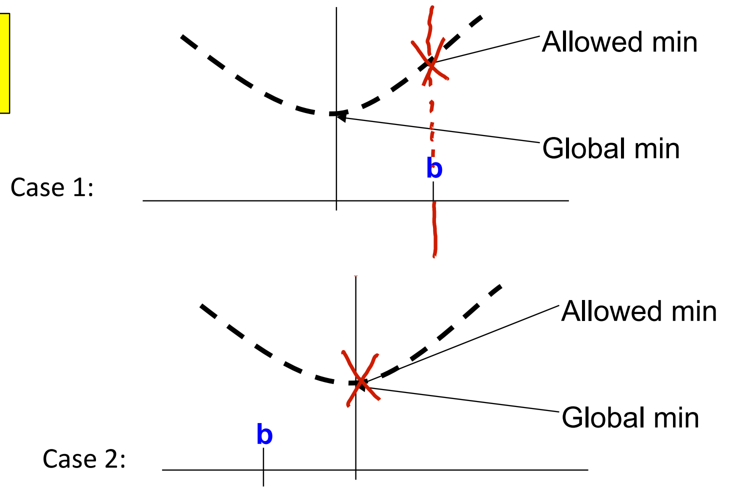
- Instead of solving these QPs directly we will solve a dual formulation of the SVM optimization problem

- 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. $u >= b$



# 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

$(x_1, x_2, \dots, x_n)$

$n$

$(w, b)$

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

- w.r.t. $(x_1 \dots x_n ; \lambda_1 \dots \lambda_k)$ $n+K$

Introducing a Lagrange multiplier for each constraint

Construct the Lagrangian for the original optimization problem

# An alternative 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 multipliers to encode it as part of the our minimization problem

Min $(w^Tw)/2$

For all x in class +1

$w^Tx+b >= 1$

For all x in class -1

$w^Tx+b <= -1$

**Why?**

Min $(w^Tw)/2$

$(w^Tx_i+b)y_i >= 1$

# An alternative (dual) representation of the SVM QP

Min $(w^Tw)/2$

$(w^Tx_i+b)y_i >= 1$

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

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

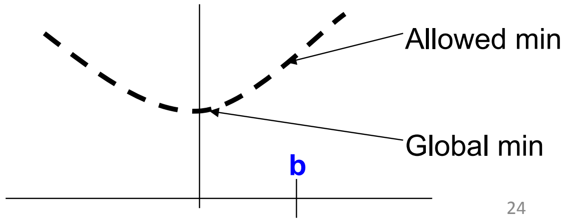
$\min_x x^2$

s.t. $x \geq b$ → $b-x \leq 0$

To

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

s.t. $\alpha \geq 0$



# Lagrange multiplier for SVMs

Lagrange formulation

$$\min_{w,b} \max_{\alpha} \frac{w^T w}{2} - \sum_{i=1}^{N\text{-train}} \alpha_i [(w^T x_i + b) y_i - 1]$$

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

Original formulation

Min $(w^T w)/2$ $\quad x^2$

$(w^T x_i + b) y_i >= 1$ $\quad x \geq b$

$. x^2 - \alpha(x-b)$

for $i \in$ train

$\alpha_i$

Using this new formulation we can derive w and b by taking the derivative w.r.t. w and $\alpha$ 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

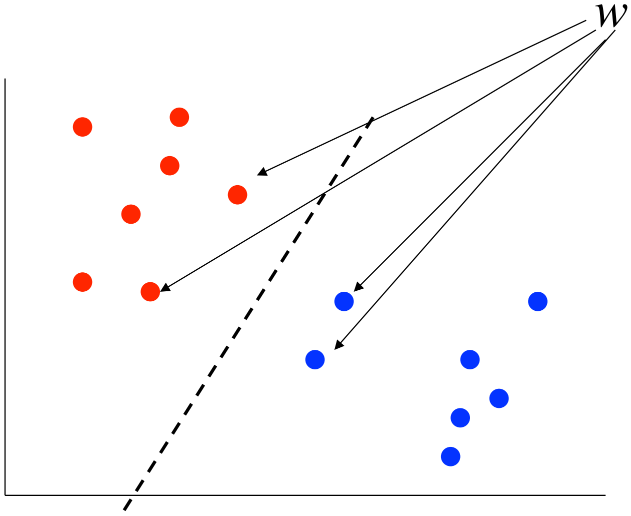
Finally, taking the derivative w.r.t. b we get:

$$\sum_i \alpha_i y_i = 0$$

# Dual SVM - interpretation

training points

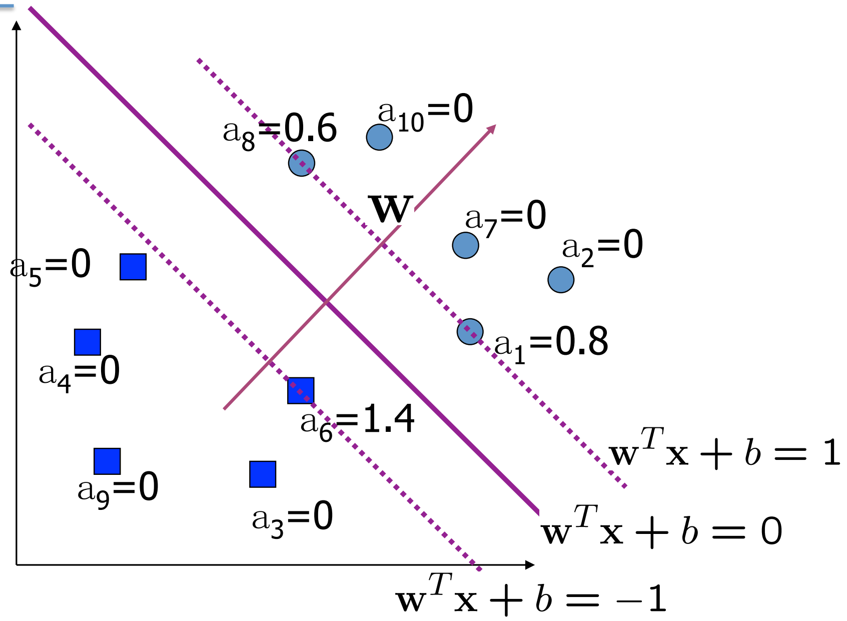
$$w = \sum_i \alpha_i x_i y_i$$



$\alpha_i \geq 0$

For $\alpha$'s that are 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

$n \quad \alpha_i$

$$\max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y}_\mathrm{i} y_j \mathbf{x}_\mathbf{i}^T \mathbf{x}_\mathbf{j}$$

$$\sum_i \alpha_i \mathrm{y}_\mathrm{i} = 0$$

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

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

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

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

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

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

$$\sum_i \alpha_i y_i = 0$$

Easier than original QP, a QP solver can be used to find $a_i$

# Dual SVM for linearly separable case

$$\sum_{i=1}^{n_{train}} \sum_{j=1}^{n_{train}}$$

Our dual target function:

$$\max_\alpha \sum_{i=1}^{n_{train}} \alpha_i - \frac{1}{2}\sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y}_\mathrm{i} y_j \mathbf{x}_\mathbf{i}^T \mathbf{x}_\mathbf{j}$$

Training

$$\sum_i \alpha_i \mathrm{y}_\mathrm{i} = 0$$

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

Dot product for all training samples

Dot product with training samples

To evaluate a new sample $x_{ts}$ we need to compute:

$$\mathrm{w}^\mathrm{T} x_{ts} + b = \sum_\mathrm{i} \alpha_i \mathrm{y}_\mathrm{i} \mathbf{x}_\mathbf{i}^T \mathbf{x}_{ts} + b$$

Testing

~~$\alpha_i \neq 0$~~

$\alpha_i > 0$

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

# Dual formulation for non linearly separable case

Dual target function:

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

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

$$\boxed{C > \alpha_i} \geq 0, \forall i$$

Hyperparameter C should be tuned through k-folds CV

The only difference is that the \alpha are now bounded

To evaluate a new sample $x_j$ we need to compute:

$$\mathrm{w}^{\mathrm{T}} x_j + b = \sum_{\mathrm{i}} \alpha_i \mathrm{y_i} \mathbf{x_i}^T \mathbf{x_j} + b$$

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

Once again, a QP solver can be used to find $a_i$

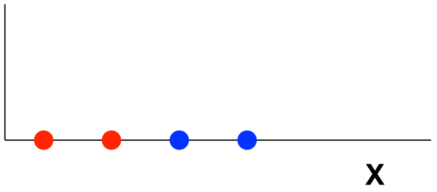
# Today

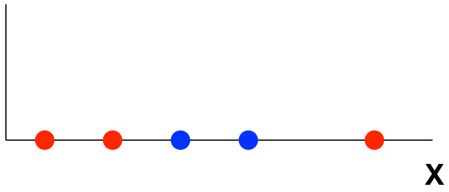
❑ Support Vector Machine (SVM)
- ✓ History of SVM
- ✓ Large Margin Linear Classifier
- ✓ Define Margin (M) in terms of model parameter
- ✓ Optimization to learn model parameters (w, b)
- ✓ Non linearly separable case
- ✓ Optimization with dual form
- ➡ ✓ Nonlinear decision boundary
- ✓ Practical Guide

# Classifying in 1-d

Can an SVM correctly classify this data?

What about this?

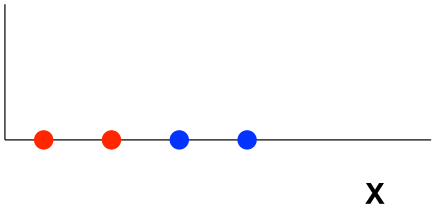


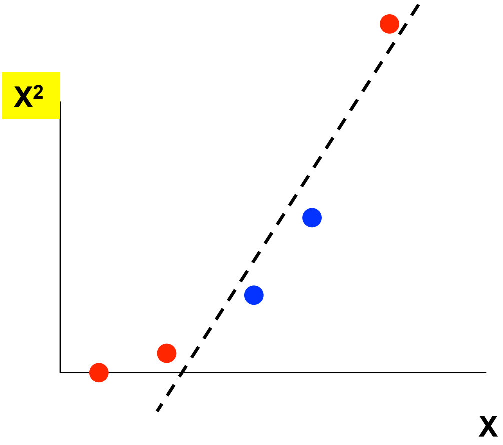


# Classifying in 1-d

Can an SVM correctly classify this data?

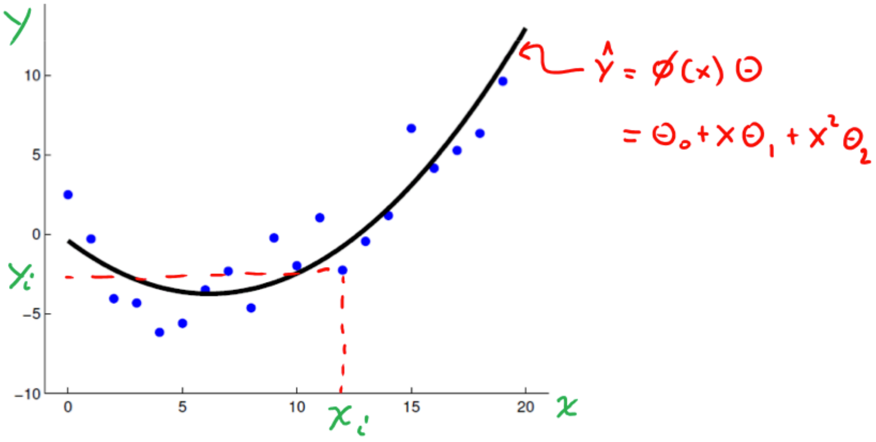
And now? (extend with polynomial basis )





# RECAP: **Polynomial regression**

For example, $\phi(x) = [1, x, x^2]$



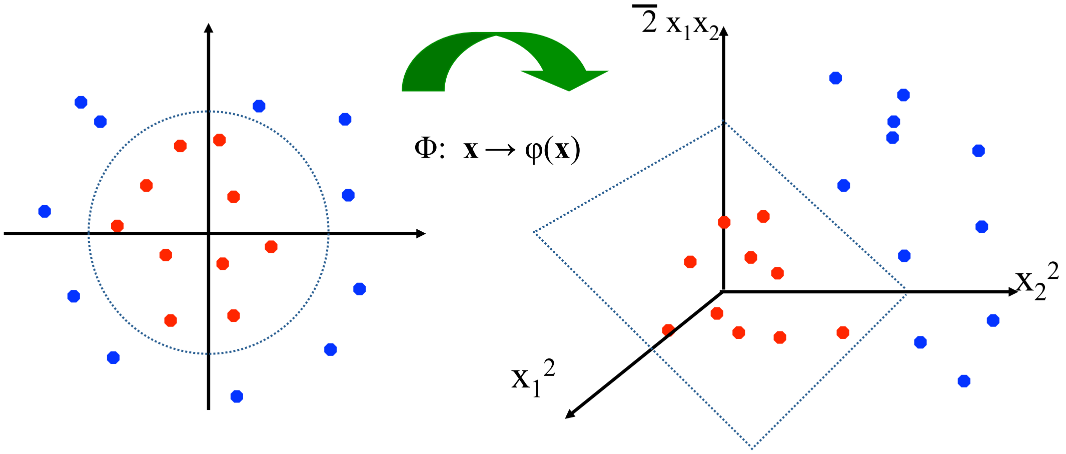
Dr. Nando de Freitas's tutorial slide

# Non-linear SVMs: 2D

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

$\mathrm{x}=(\mathrm{x}_1,\mathrm{x}_2)$

$\varphi(\mathbf{x}) =(\mathrm{x}_1^2,\mathrm{x}_2^2,\ \overline{2}\ \mathrm{x}_1\mathrm{x}_2)$

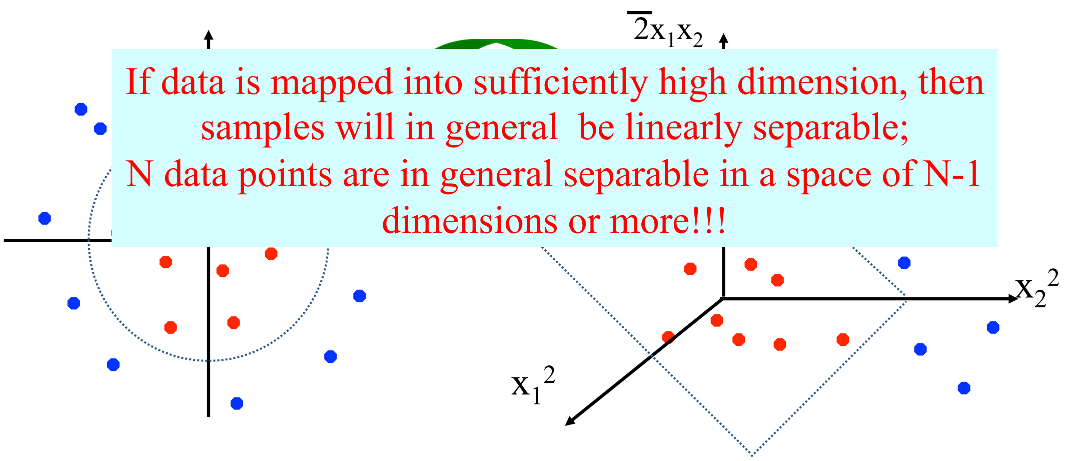

$\overline{2}\ \mathrm{x}_1\mathrm{x}_2$
$\Phi$: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$
$\mathrm{x}_2^2$
$\mathrm{x}_1^2$

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 (ϕ(**x**) )where the training set is separable:

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



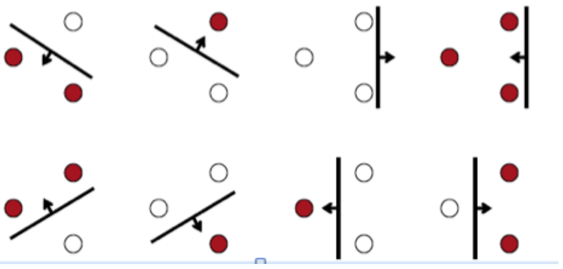
If data is mapped into sufficiently high dimension, then samples will in general be linearly separable;
N data points are in general separable in a space of N-1 dimensions or more!!!

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

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

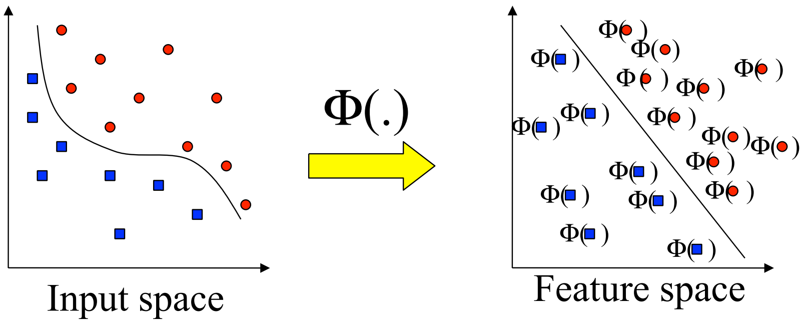
If data is mapped into sufficiently high dimension, then samples will in general be linearly separable;
N data points are in general separable in a space of N-1 dimensions or more!!!

- **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

$m$

$X \to \Phi(X)$

The ⊠2 term will become clear in the next slide

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

m+1 linear terms

m quadratic terms

m(m-1)/2 pairwise terms

$$K(\mathbf{x}, z) := \Phi(\mathbf{x})^T \Phi(z)$$

# Dot product for quadratic kernels

How many operations do we need for the dot product?

$O(m^2)$ $O(m^2)$

$O(m^2)$

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

m m m(m-1)/2 **=~ m²**

$$K(\mathbf{x},z) := \Phi(\mathbf{x})^T\Phi(z)$$

# The kernel trick

How many operations do we need for the dot product?

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

$O(m^2)$

m m m(m-1)/2 **=~ m²**

However, we can obtain dramatic savings by noting that

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

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

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

$K(x,z)$

$O(m)$

**We only need m operations!**

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

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

# Where we are

Our dual target function:

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

$K(x_i, x_j)$

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

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

To evaluate a new sample $x_j$ we need to compute:

$$\mathrm{w}^T \Phi(\mathbf{x}_{\overset{k}{j}}) + b = \sum_{\mathrm{i}} \alpha_i \mathrm{y_i} \boxed{\Phi(\mathbf{x_i})^T \Phi(\mathbf{x}_{\overset{k}{j}})} + b$$

$K(x_i, x_k)$

*mr* operations where *r* are the number of support vectors (whose $\text{alpha}_i$>0)

*mn*$^2$ operations at each iteration

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

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

$K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ is called the kernel function.

# More examples of kernel functions

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

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

$O(m^d)$

$O(m)$

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

- Radial basis kernel

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

In this case., the feature space of the kernel has an infinite number of dimensions

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

# Why do SVMs work?

❑ If we are using huge features spaces (e.g., 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

# 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  
✓ Practical Guide

# 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

# Practical Guide to SVM

- From authors of as LIBSVM:
  - A Practical Guide to Support Vector Classification Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, 2003-2010
  - http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

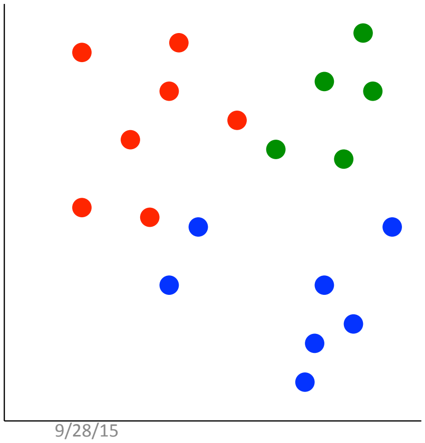
# LIBSVM

- http://www.csie.ntu.edu.tw/~cjlin/libsvm/
  - ✓Developed by Chih-Jen Lin etc.
  - ✓Tools for Support Vector classification
  - ✓Also support multi-class classification
  - ✓C++/Java/Python/Matlab/Perl wrappers
  - ✓Linux/UNIX/Windows
  - ✓SMO implementation, fast!!!

A Practical Guide to Support Vector Classification

# 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

# (a) Data file formats for LIBSVM

- Training.dat

```
+1 1:0.708333 2:1 3:1 4:-0.320755
-1 1:0.583333 2:-1  4:-0.603774 5:1
+1 1:0.166667 2:1 3:-0.333333 4:-0.433962
-1 1:0.458333 2:1 3:1 4:-0.358491 5:0.374429
…
```

- Testing.dat

# (b) Feature Preprocessing

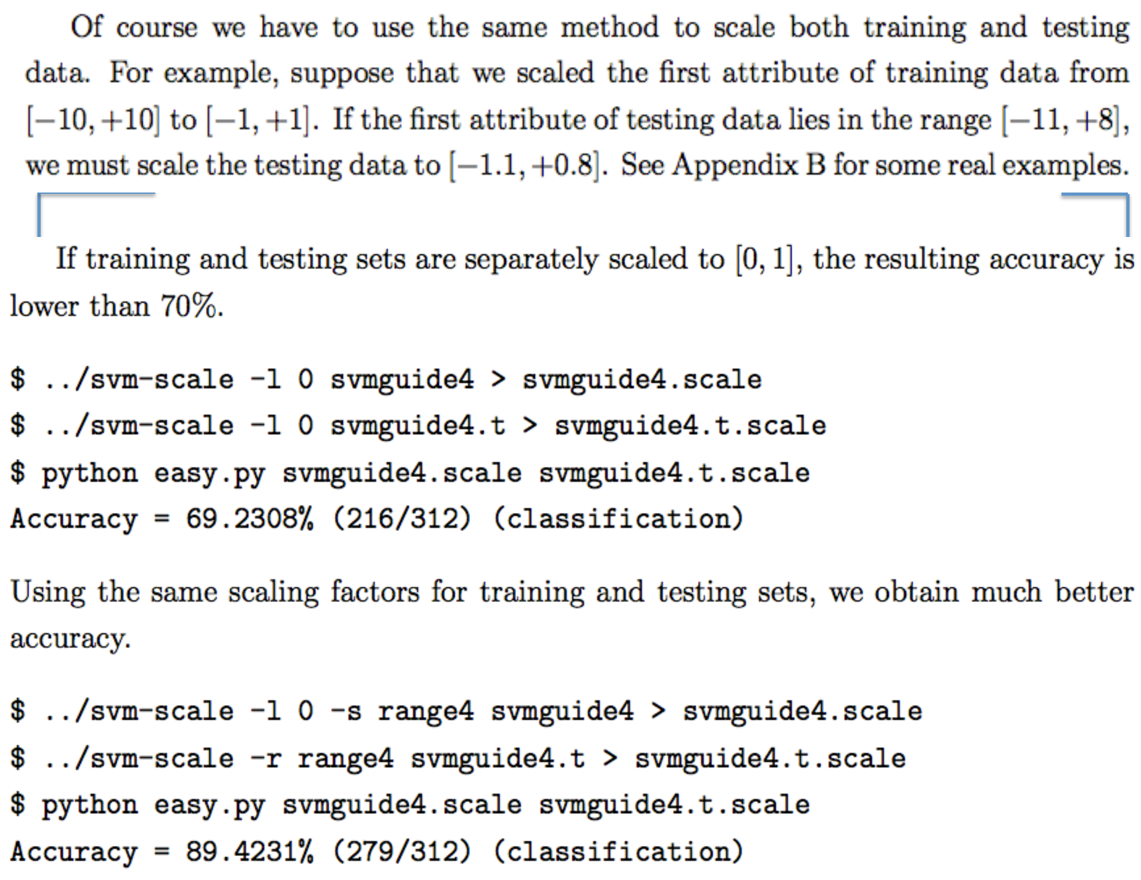
- (1) Categorical Feature
  - Recommend using m numbers to represent an m-category attribute.
  - Only one of the m numbers is one, and others are zero.
  - For example, a three-category attribute such as {red, green, blue} can be represented as (0,0,1), (0,1,0), and (1,0,0)

A Practical Guide to Support Vector Classification

# Feature Preprocessing

- (2) Scaling before applying SVM is very important
  - to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges.
  - to avoid numerical difficulties during the calculation
  - Recommend linearly scaling each attribute to the range [1, +1] or [0, 1].

A Practical Guide to Support Vector Classification



Of course we have to use the same method to scale both training and testing data. For example, suppose that we scaled the first attribute of training data from $[-10,+10]$ to $[-1,+1]$. If the first attribute of testing data lies in the range $[-11,+8]$, we must scale the testing data to $[-1.1,+0.8]$. See Appendix B for some real examples.

If training and testing sets are separately scaled to $[0,1]$, the resulting accuracy is lower than 70%.

```
$ ../svm-scale -l 0 svmguide4 > svmguide4.scale
$ ../svm-scale -l 0 svmguide4.t > svmguide4.t.scale
$ python easy.py svmguide4.scale svmguide4.t.scale
Accuracy = 69.2308% (216/312) (classification)
```

Using the same scaling factors for training and testing sets, we obtain much better accuracy.

```
$ ../svm-scale -l 0 -s range4 svmguide4 > svmguide4.scale
$ ../svm-scale -r range4 svmguide4.t > svmguide4.t.scale
$ python easy.py svmguide4.scale svmguide4.t.scale
Accuracy = 89.4231% (279/312) (classification)
```

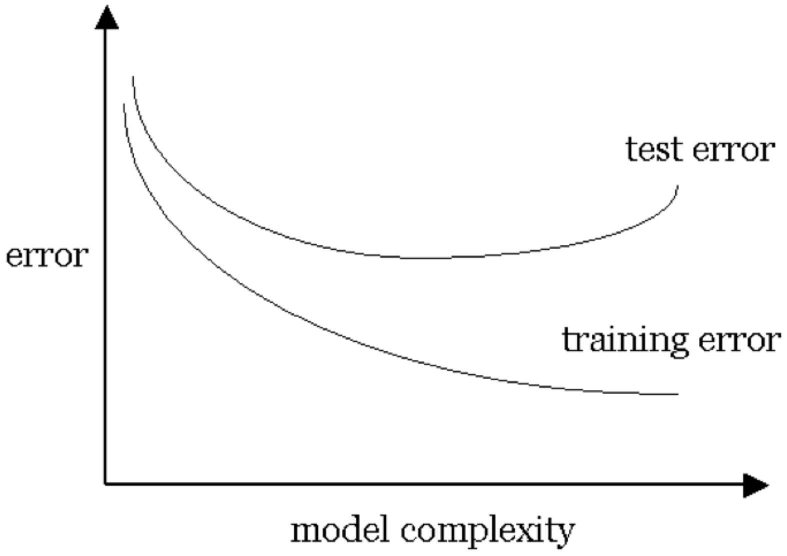
# Feature Preprocessing

- (3) missing value
  - Very very tricky !
  - Easy way: to substitute the missing values by the mean value of the variable
  - A little bit harder way: imputation using nearest neighbors
  - Even more complex: e.g. EM based (beyond the scope)

A Practical Guide to Support Vector Classification

# (c) Model Selection

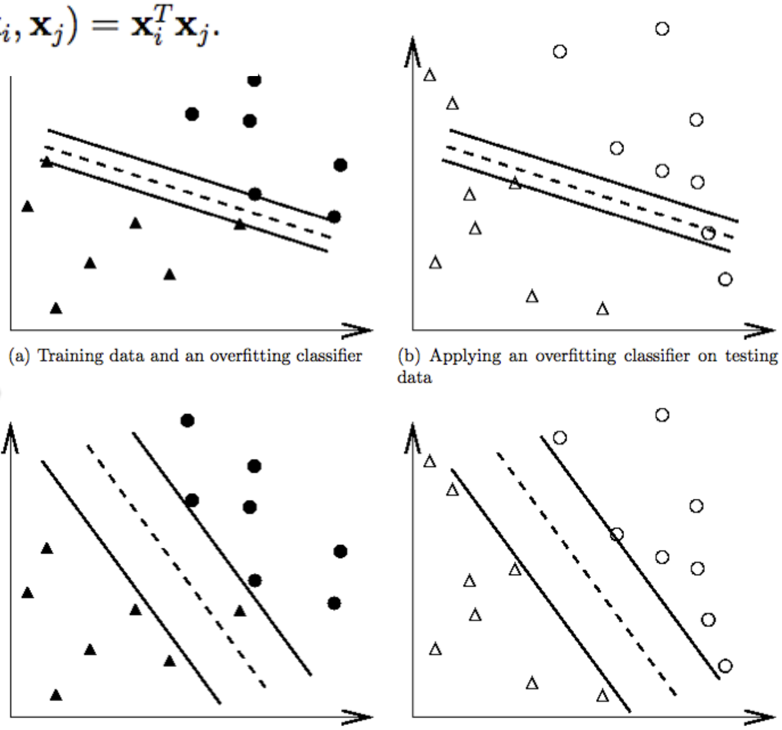
Our goal: find the model $M$ which minimizes the test error:



# (c) Model Selection (e.g. for linear kernel)

- linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$.

Select the right penalty parameter C



(a) Training data and an overfitting classifier

(b) Applying an overfitting classifier on testing data

(c) Training data and a better classifier

(d) Applying a better classifier on testing data

# (c) Model Selection

- radial basis function (RBF): $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$, $\gamma > 0$.

  two parameters for an RBF kernel: $C$ and $\gamma$

- polynomial: $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma {\mathbf{x}_i}^T \mathbf{x}_j + r)^d$, $\gamma > 0$.

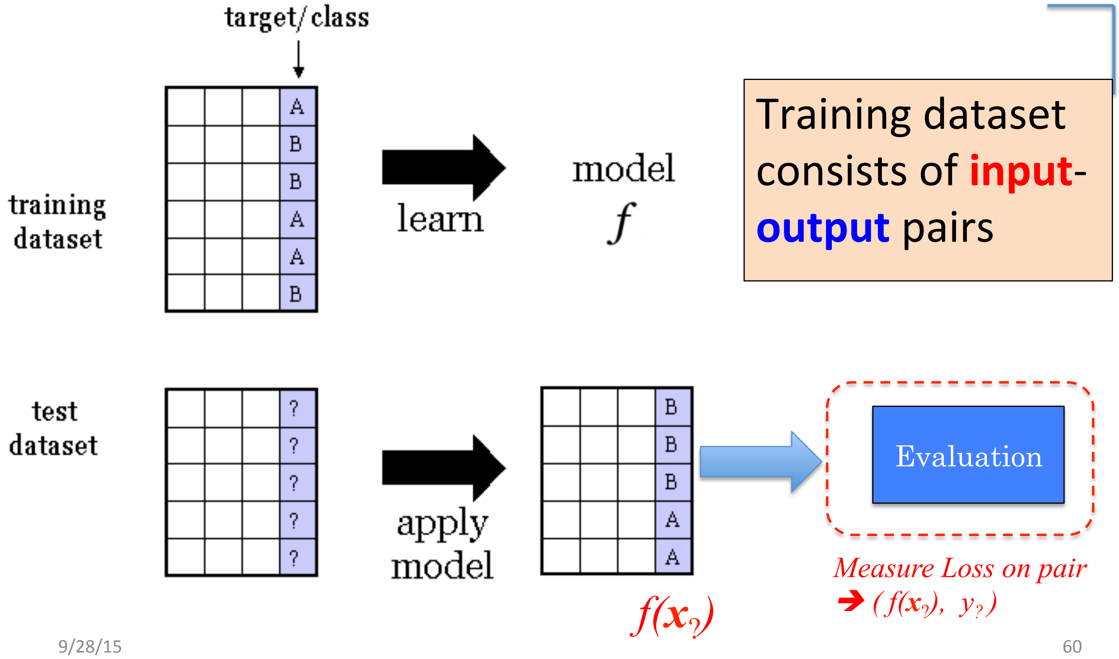
  Three parameters for a polynomial kernel

A Practical Guide to Support Vector Classification

# (d) Pipeline Procedures

- (1) train / test
- (2) k-folds cross validation
- (3) k-CV on train to choose hyperparameter / then test

# Evaluation Choice-I: Train and Test



Training dataset consists of **input-output** pairs

*Measure Loss on pair*
➔ $( f(\mathbf{x}_?),\ y_? )$

# Evaluation Choice-II:

## Cross Validation

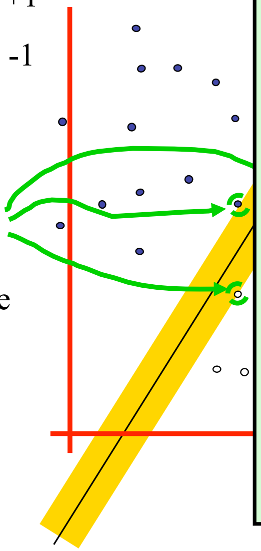
- Problem: don’t have enough data to set aside a test set
- Solution: Each data point is used both as train and test
- Common types:
  - -K-fold cross-validation (e.g. K=5, K=10)
  - -2-fold cross-validation
  - -Leave-one-out cross-validation (LOOCV)

A good practice is : to random shuffle all training sample before splitting

# Why Maximum Margin for SVM ?

● denotes +1

○ denotes -1



Support Vectors are those datapoints that the margin pushes up against

1. Intuitively this feels safest.
2. If we’ve made a small error in the location of the boundary (it’s been jolted in its perpendicular direction) this gives us least chance of causing a misclassification.
3. **LOOCV is easy since the model is immune to removal of any non-support-vector datapoints.**
4. There’s some theory (using VC dimension) that is related to (but not the same as) the proposition that this is a good thing.
5. Empirically it works very very well.

# Evaluation Choice-III:

Many beginners use the following procedure now:

- Transform data to the format of an SVM package
- Randomly try a few kernels and parameters
- Test

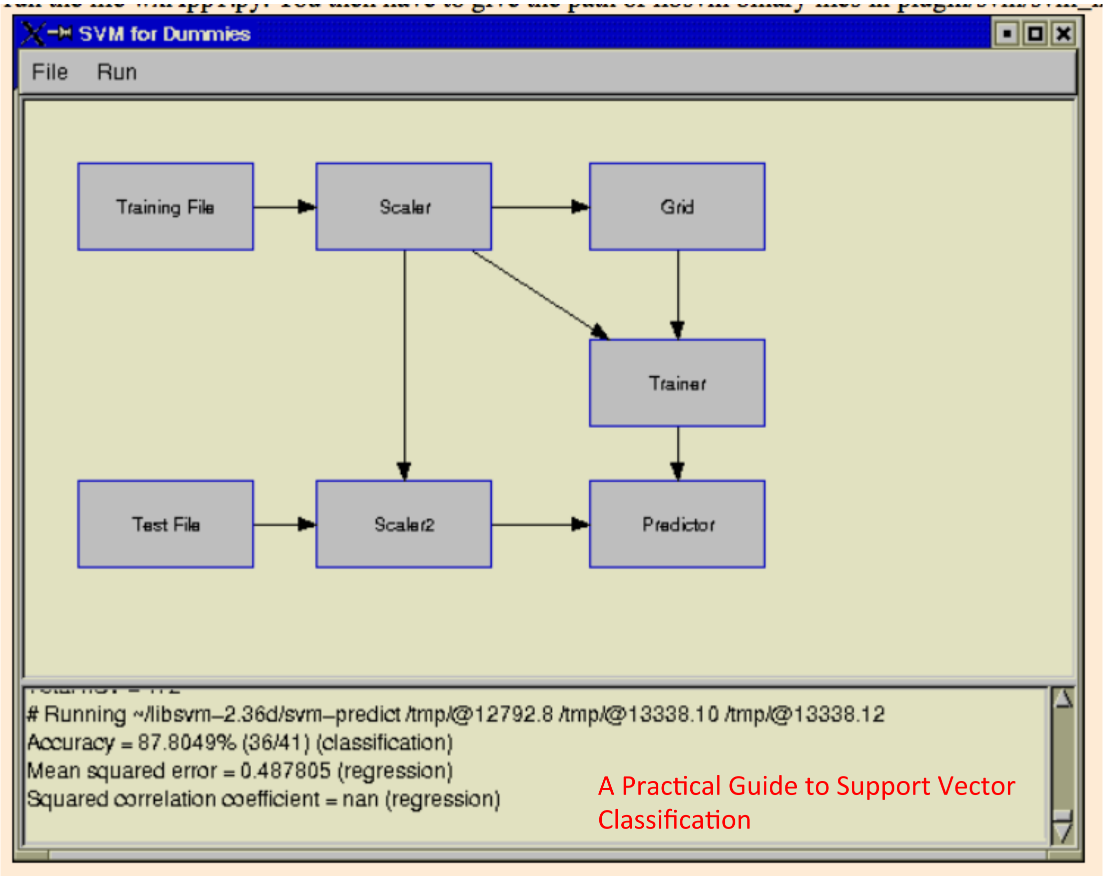
Basic solution
For HW2-Q2

We propose that beginners try the following procedure first:

- Transform data to the format of an SVM package
- Conduct simple scaling on the data
- Consider the RBF kernel $K(\mathbf{x}, \mathbf{y}) = e^{-\gamma\|\mathbf{x}-\mathbf{y}\|^2}$
- Use cross-validation to find the best parameter $C$ and $\gamma$
- Use the best parameter $C$ and $\gamma$ to train the whole training set[5]
- Test

more
advanced
solution
For HW2-Q2

A Practical Guide to Support Vector Classification



```
# Running ~/libsvm-2.36d/svm-predict /tmp/@12792.8 /tmp/@13338.10 /tmp/@13338.12
Accuracy = 87.8049% (36/41) (classification)
Mean squared error = 0.487805 (regression)
Squared correlation coefficient = nan (regression)
```

A Practical Guide to Support Vector Classification

# Today: Review & Practical Guide

- ❑ 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
  - ✓ Practical Guide
    - ✓ File format / LIBSVM
    - ✓ Feature preprocsssing
    - ✓ Model selection
    - ✓ Pipeline procedure

# References

- Big thanks to Prof. Ziv Bar-Joseph @ CMU for allowing me to reuse some of his slides
- Elements of Statistical Learning, by Hastie, Tibshirani and Friedman
- Prof. Andrew Moore @ CMU's slides
- UMN Data Mining Course Slides
- A Practical Guide to Support Vector Classification Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, 2003-2010