Support Vector Machines II

Dr. Jianlin Cheng

Computer Science Department
University of Missouri, Columbia
Fall, 2015

Slides Adapted from Book and CMU, Stanford Machine Learning Courses

The SMO Algorithm

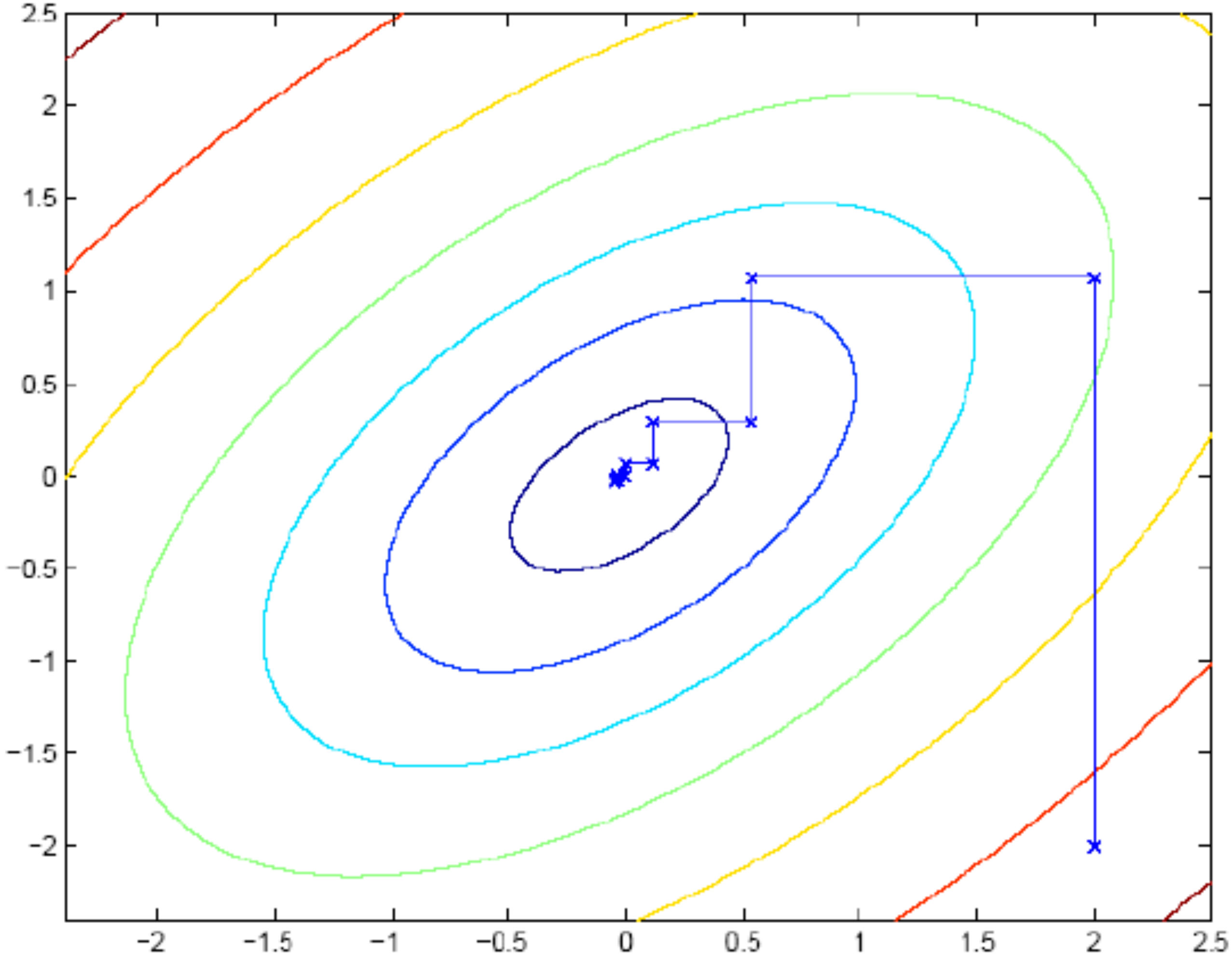
- Consider solving the unconstrained opt problem:

$$\max_{\alpha} W(\alpha_1, \alpha_2, \ldots, \alpha_m)$$

- We've already see three opt algorithms!
  - ?
  - ?
  - ?

- Coordinate ascend:

Coordinate Ascend



Sequential minimal optimization

- Constrained optimization:

$$\max_{\alpha} \quad \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \quad i = 1, \ldots, m$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

- Question: can we do coordinate along one direction at a time (i.e., hold all $\alpha_{[-i]}$ fixed, and update α_i?)

Sequential minimal optimization

Repeat till convergence

1. Select some pair α_i and α_j to update next (using a heuristic that tries to pick the two that will allow us to make the biggest progress towards the global maximum).

How to select?

2. Re-optimize J(α) with respect to α_i and α_j, while holding all the other α_k 's ($k \neq i; j$) fixed.

Will this procedure converge?

Sequential minimal optimization

$$\max_{\alpha} \quad \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

KKT:

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \quad i = 1, \ldots, k$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

- Let's hold $\alpha_3, \ldots, \alpha_m$ fixed and reopt J w.r.t. α_1 and α_2

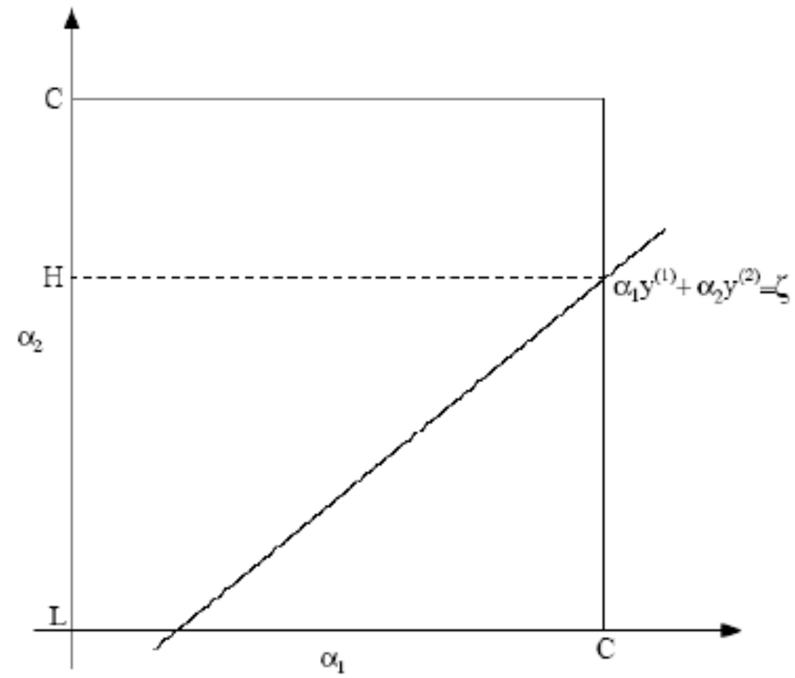
Convergence of SMO

- The constraints:

$$\alpha_1 y_1 + \alpha_2 y_2 = \xi$$

$$0 \leq \alpha_1 \leq C$$

$$0 \leq \alpha_2 \leq C$$



- The objective:

$$\mathcal{J}(\alpha_1, \alpha_2, \ldots, \alpha_m) = \mathcal{J}((\xi - \alpha_2 y_2) y_1, \alpha_2, \ldots, \alpha_m)$$

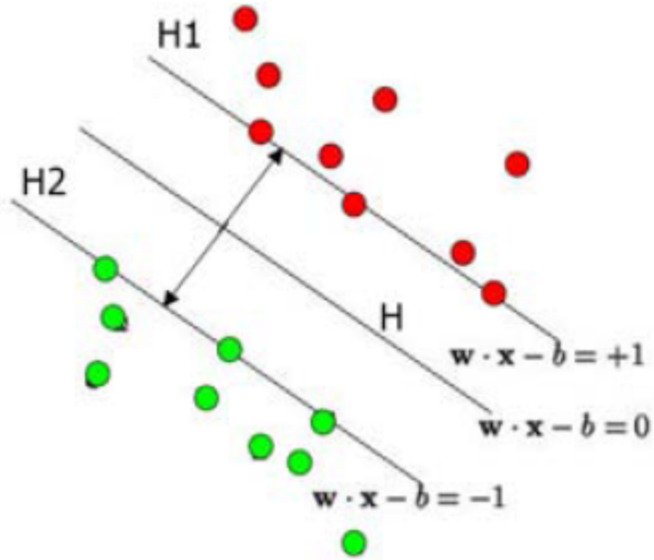
- Constrained opt:

$$\mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

Cross-Validation Error of SVM

- The leave-one-out cross-validation error does not depend on the dimensionality of the feature space but only on the # of support vectors!

$$\text{Leave-one-out CV error} = \frac{\#\text{ support vectors}}{\#\text{ of training examples}}$$

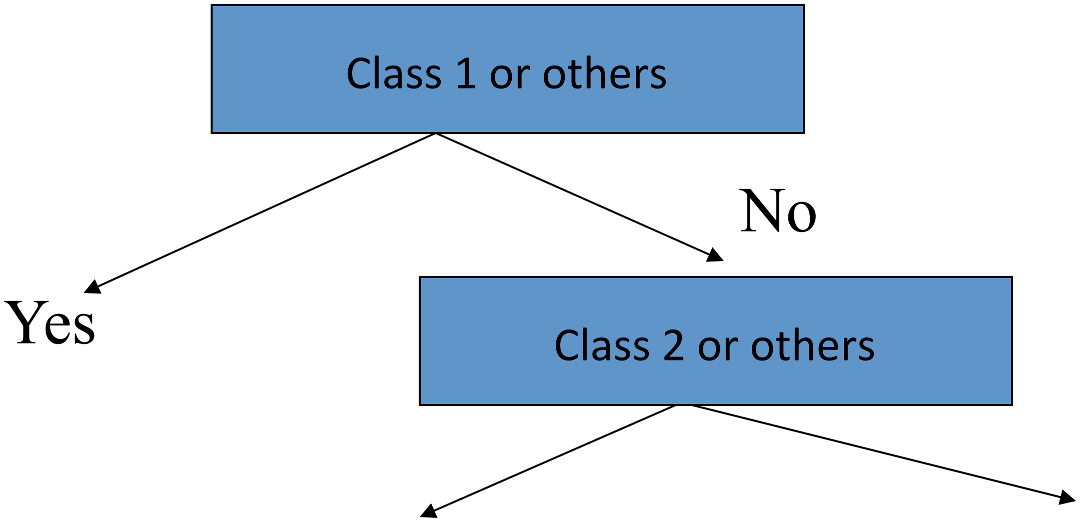


Time Complexity of Testing

- $O(MN_s)$. M is the number of operations required to evaluate inner product. M is $O(d_L)$. N_s is the number of support vectors.

Multi-Class SVM

- Most widely used method: one versus all
- Also direct multi-classification using SVM. (K. Crammer and Y. Singer. On the Algorithmic Implementation of Multi-class SVMs, JMLR, 2001)

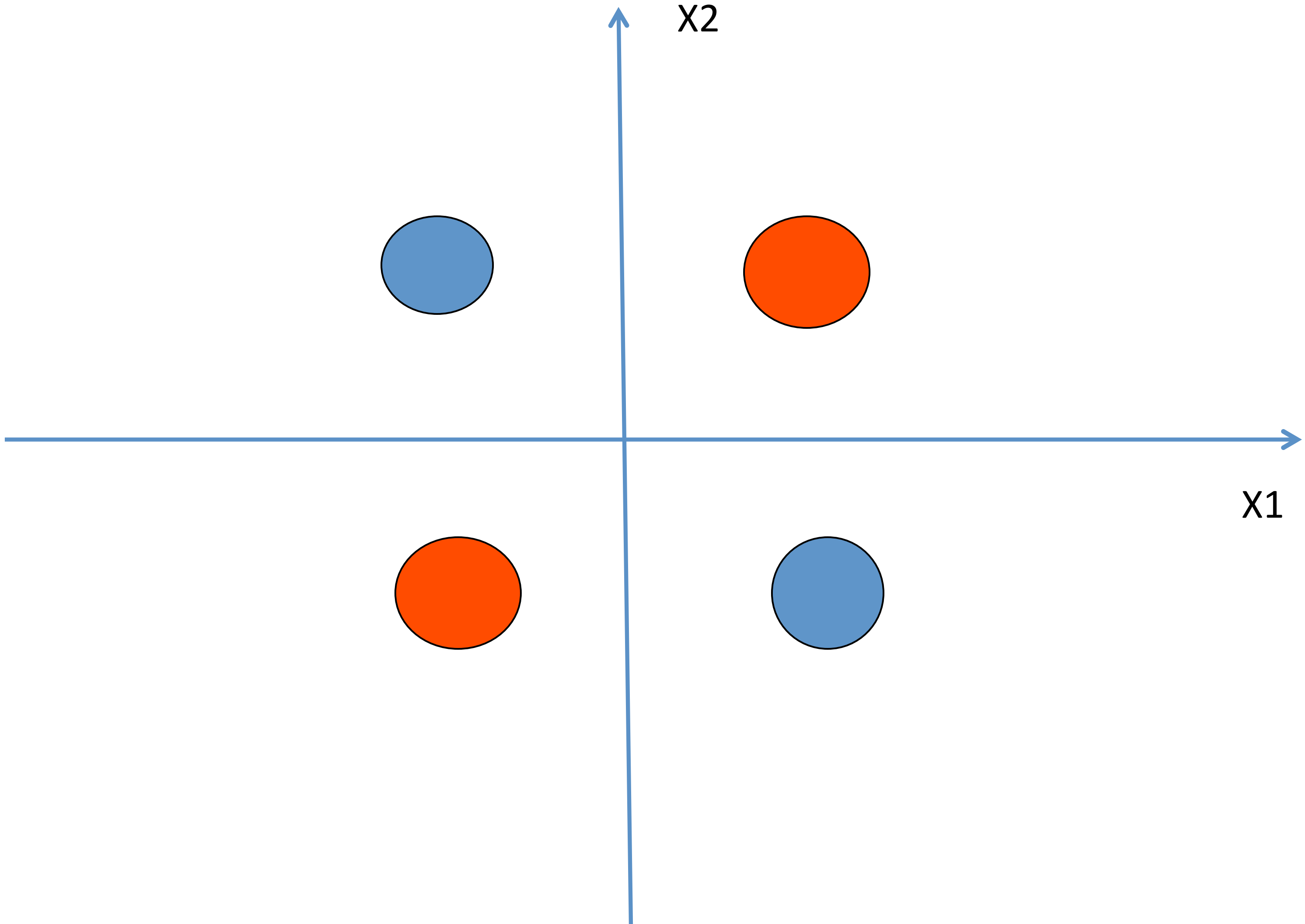


Summary

- Max-margin decision boundary
- Constrained convex optimization
  - Duality
  - The KTT conditions and the support vectors
  - Non-separable case and slack variables
  - The SMO algorithm

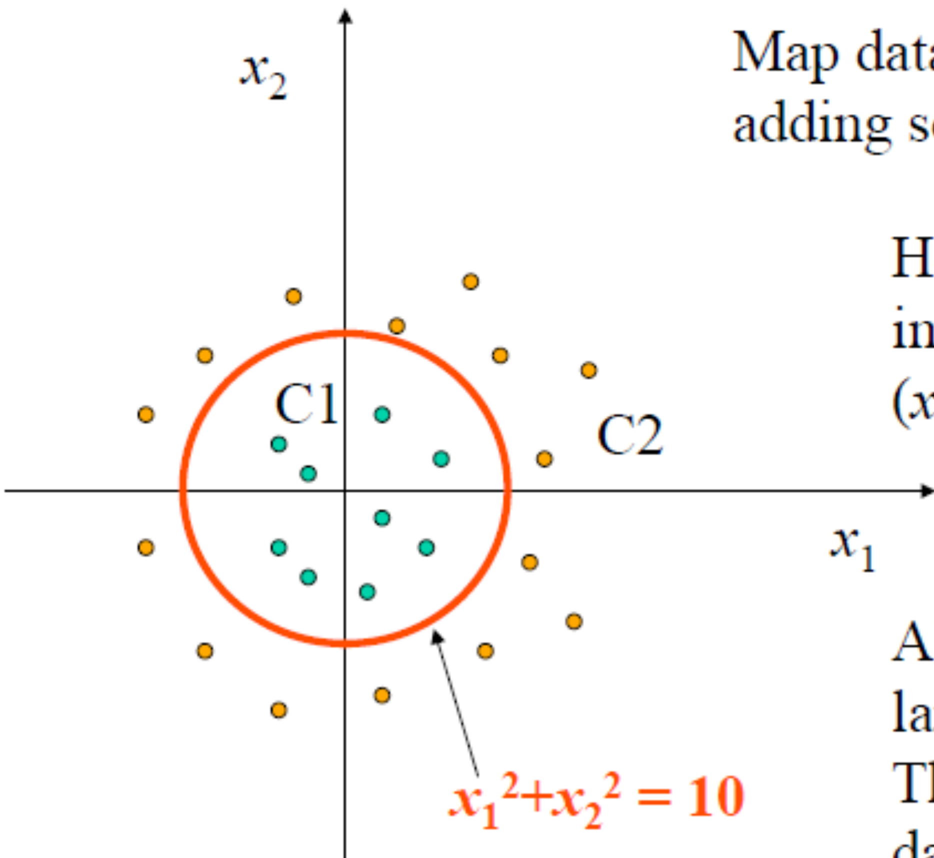
Non-Linear Decision Boundary

- So far, we have only considered large-margin classifier with a linear decision boundary
- How to generalize it to become nonlinear?
- Key idea: transform $\mathbf{x}_i$ to a higher dimensional space to "make life easier"
  - Input space: the space the point $\mathbf{x}_i$ are located
  - Feature space: the space of $\phi(\mathbf{x}_i)$ after transformation
- Why transform?
  - Linear operation in the feature space is equivalent to non-linear operation in input space
  - Classification can become easier with a proper transformation. In the XOR problem, for example, adding a new feature of x_1x_2 make the problem linearly separable (homework)



XOR problem: add a dimension x1 * x2

Support Vector Machine Approach



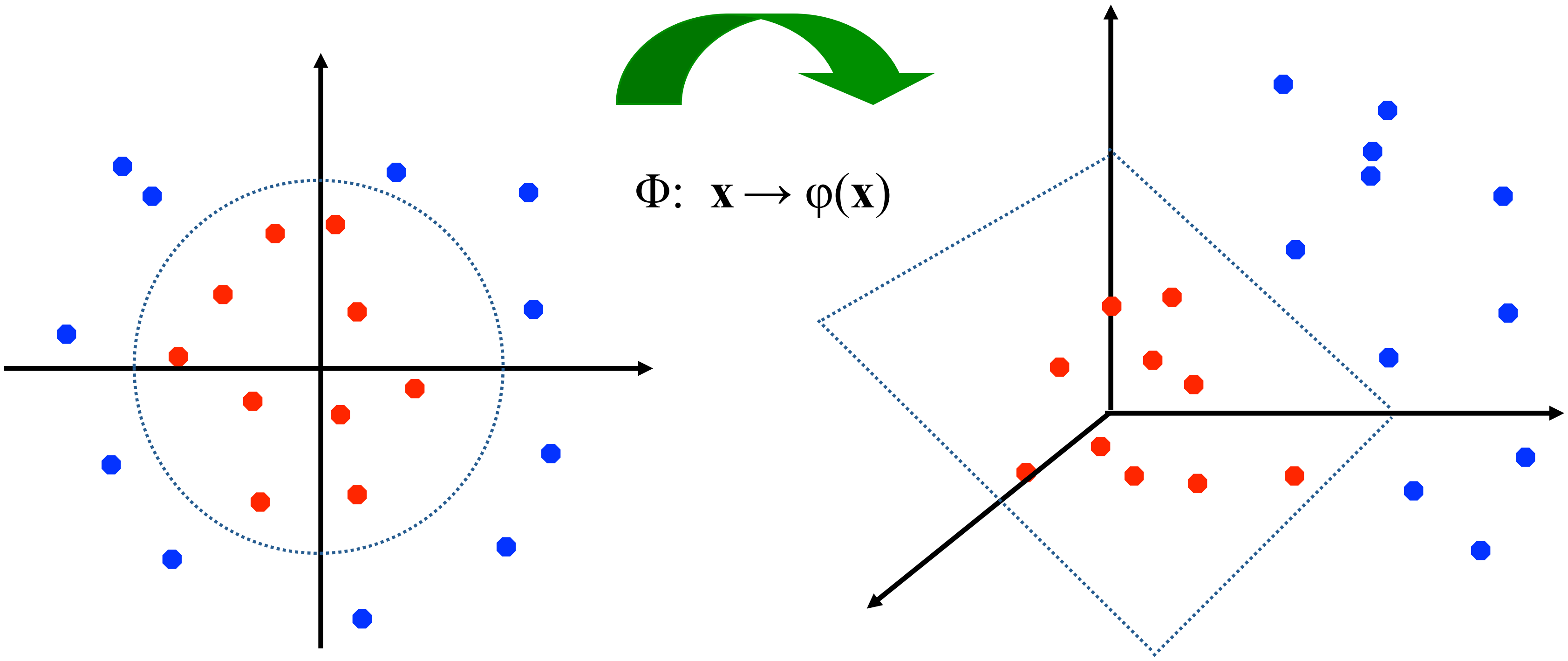
Map data point into high dimension, e.g. adding some non-linear features.

How about we augument feature into three dimension (x_1, x_2, $x_1^2+x_2^2$).

All data points in class C2 have a larger value for the third feature Than data points in C1. Now data is linearly separable.

Non-linear SVMs: Feature spaces

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



M. Tan's slides, Univ. British Columbia

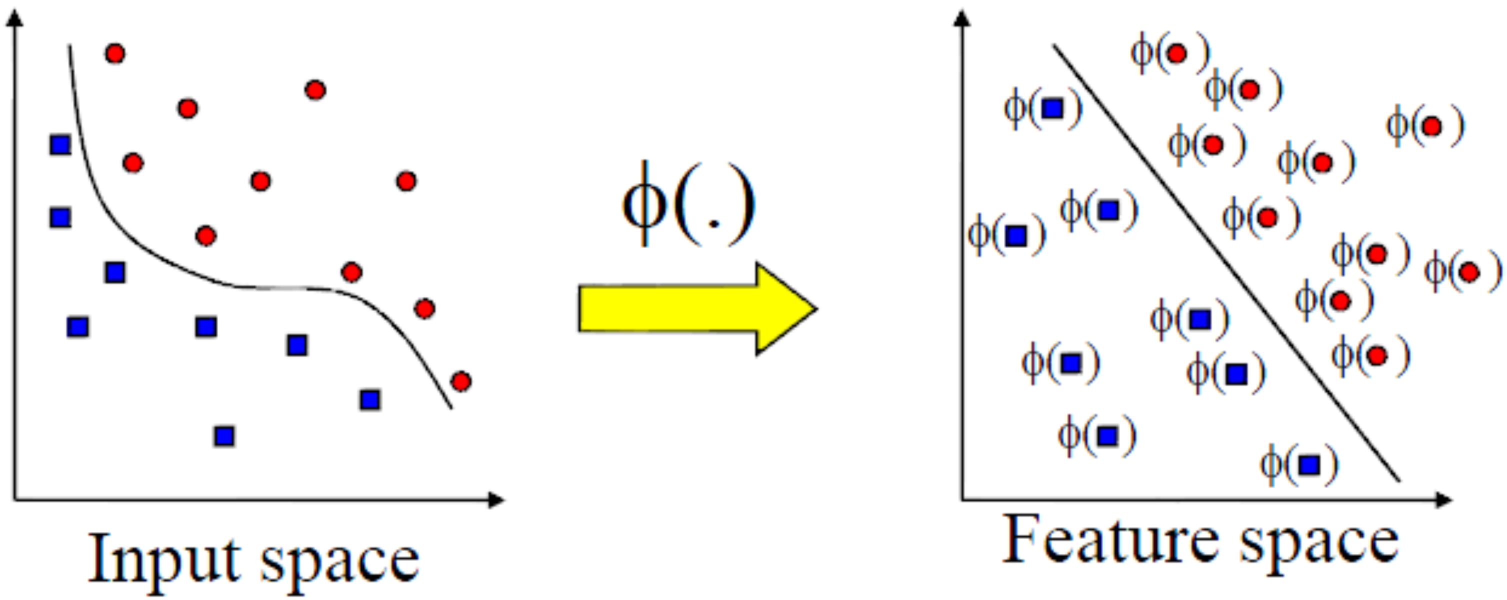
Nonlinear Support Vector Machines

- In the L_D function, what really matters is dot products: $x_i.x_j$.
- Idea: map the data to some other (possibly infinite dimensional) Euclidean space *H*, using a mapping.

$$\Phi : R^d \mapsto H$$

Then the training algorithm would only depend on the data through dot products in *H*, i.e. $\Phi(x_i).\ \Phi(x_j)$.

Transforming the Data



Note: feature space is of higher dimension than the input space in practice

Kernel Trick

- If there were a kernel function *K* such that $K(x_i,x_j) = \Phi(x_i).\Phi(x_j)$, we would only need to use *K* in the training algorithm and would never need to explicitly do the mapping Φ.
- So we simply replace $x_i.x_j$ with $K(x_i,x_j)$ in the training algorithm, the algorithm will happily produce a support vector machine which lives in a new space
- *Is training time on the mapped data significantly different from the un-mapped data?*

Kernel Trick

- Recall the SVM optimization problem

$$\max_{\alpha} \quad \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \quad i = 1, \dots, m$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

- The data points only appear as inner product
- As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly
- Many common geometric operations (angles, distances) can be expressed by inner products
- Define the kernel function K by $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

How to Use the Machine?

- We can’t get w if we do not do explicit mapping.
- Once again we use kernel trick.

$$f(x) = (\sum_{i=1}^{Ns} a_i y_i \Phi(s_i))\Phi(x) + b = \sum_{i=1}^{Ns} a_i y_i K(s_i, x) + b$$

What’s the problem from a computational point of view?

An Example of Feature Mapping

- Consider an input $\mathbf{x}=[x_1,x_2]$
- Suppose $\phi(.)$ is given as follows

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = 1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2$$

- An inner product in the feature space is

$$\left\langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} x_1' \\ x_2' \end{bmatrix}\right) \right\rangle =$$

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

$$K(\mathbf{x},\mathbf{x}') = \left(1 + \mathbf{x}^T\mathbf{x}'\right)^2$$

Common Kernels

(1) $K(x,y) = (x.y + 1)^p$. p is degree. p = 1, linear kernel.

(2) Gaussian radial basis kernel $$K(x,y) = e^{-|x-y|^2/2\sigma^2}$$

(3) Hyperbolic Tanh kernel $$K(x,y) = \tanh(kx.y - \delta)$$

Note: RBF kernel, the weights (a_i) and centers (S_i) are automatically learned. Tanh kernel is equivalent to two-layer neural network, where number of hidden units is number of support vectors. a_i corresponds to the weights of the second layer.

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Kernel Matrix

- Suppose for now that K is indeed a valid kernel corresponding to some feature mapping ϕ, then for $x_1, \ldots, x_m$, we can compute an $m \times m$ matrix $K = \{K_{i,j}\}$, where $K_{i,j} = \phi(x_i)^T \phi(x_j)$
- This is called a kernel matrix! **Or Gram Matrix**
- Now, if a kernel function is indeed a valid kernel, and its elements are dot-product in the transformed feature space, it must satisfy:
  - Symmetry $K=K^T$

    proof $K_{i,j} = \phi(x_i)^T \phi(x_j) = \phi(x_j)^T \phi(x_i) = K_{j,i}$
  - Positive –semidefinite $y^T K y \geq 0 \quad \forall y$

    proof?

Proof

- K is positive semi-definite, i.e. $\alpha K \alpha \geq 0$ for all $\alpha \in \mathbb{R}^m$ and all kernel matrices $K \in \mathbb{R}^{m \times m}$. Proof (from class):

$$\sum_{i,j}^{m} \alpha_i \alpha_j K_{ij} = \sum_{i,j}^{m} \alpha_i \alpha_j \langle \Phi(x_i), \Phi(x_j) \rangle$$

$$= \langle \sum_{i}^{m} \alpha_i \Phi(x_i), \sum_{j}^{m} \alpha_j \Phi(x_j) \rangle = || \sum_{i}^{m} \alpha_i \Phi(x_i) ||^2 \geq 0$$

Mercer Kernel

Theorem (Mercer): Let K: $\mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$ be given. Then for K to be a valid (Mercer) kernel, it is necessary and sufficient that for any $\{x_i, \ldots, x_m\}$, $(m < \infty)$, the corresponding kernel matrix is symmetric positive semi-denite.

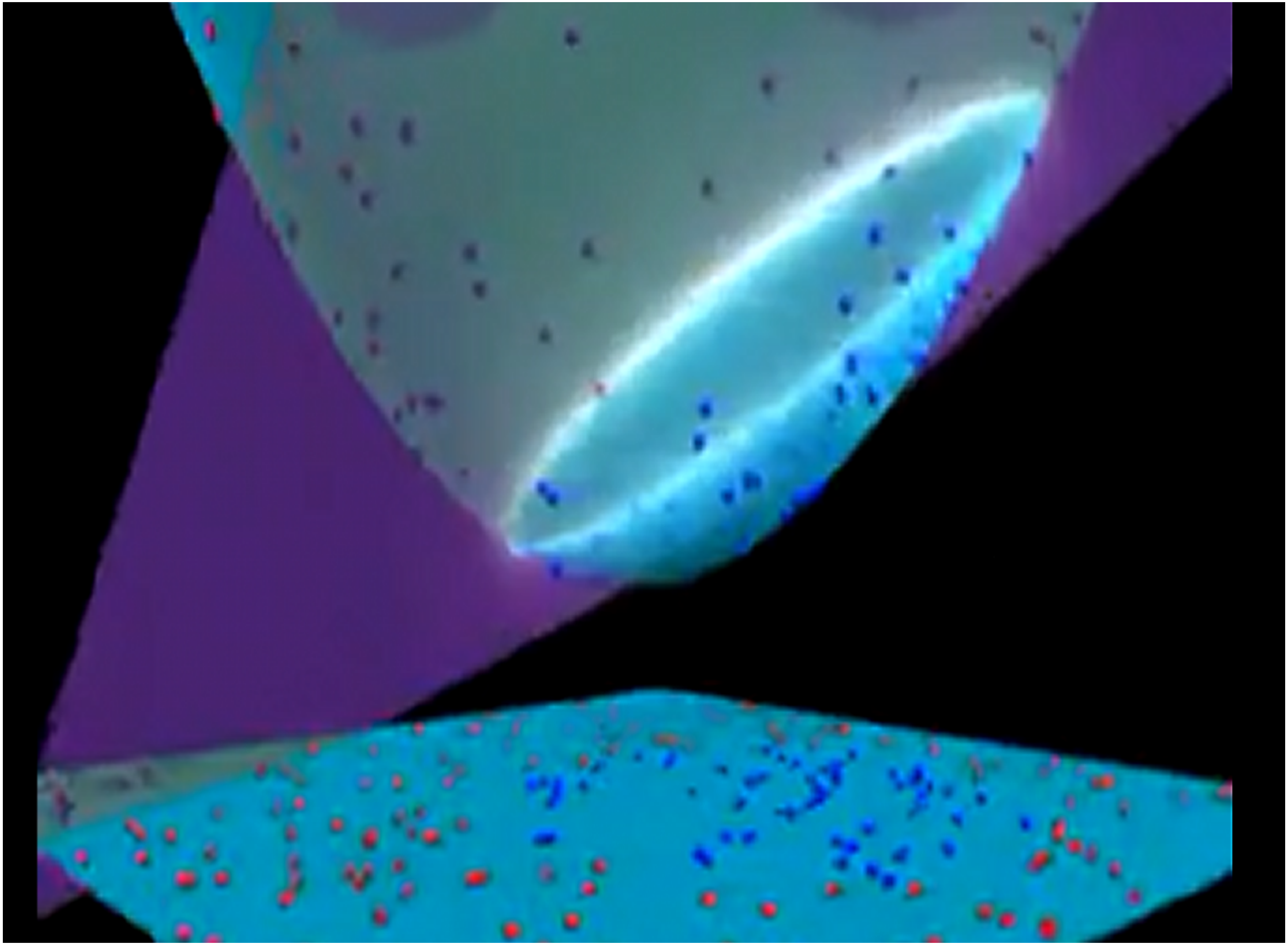
Define Your Own Kernel Function or Combine Standard Kernel Function

- We can write our own kernel function
- Some non-kernel function may still work in practice
- Combine standard kernels: k1 + k2 is a kernel, a*k1 is a kernel, etc. Can you prove?

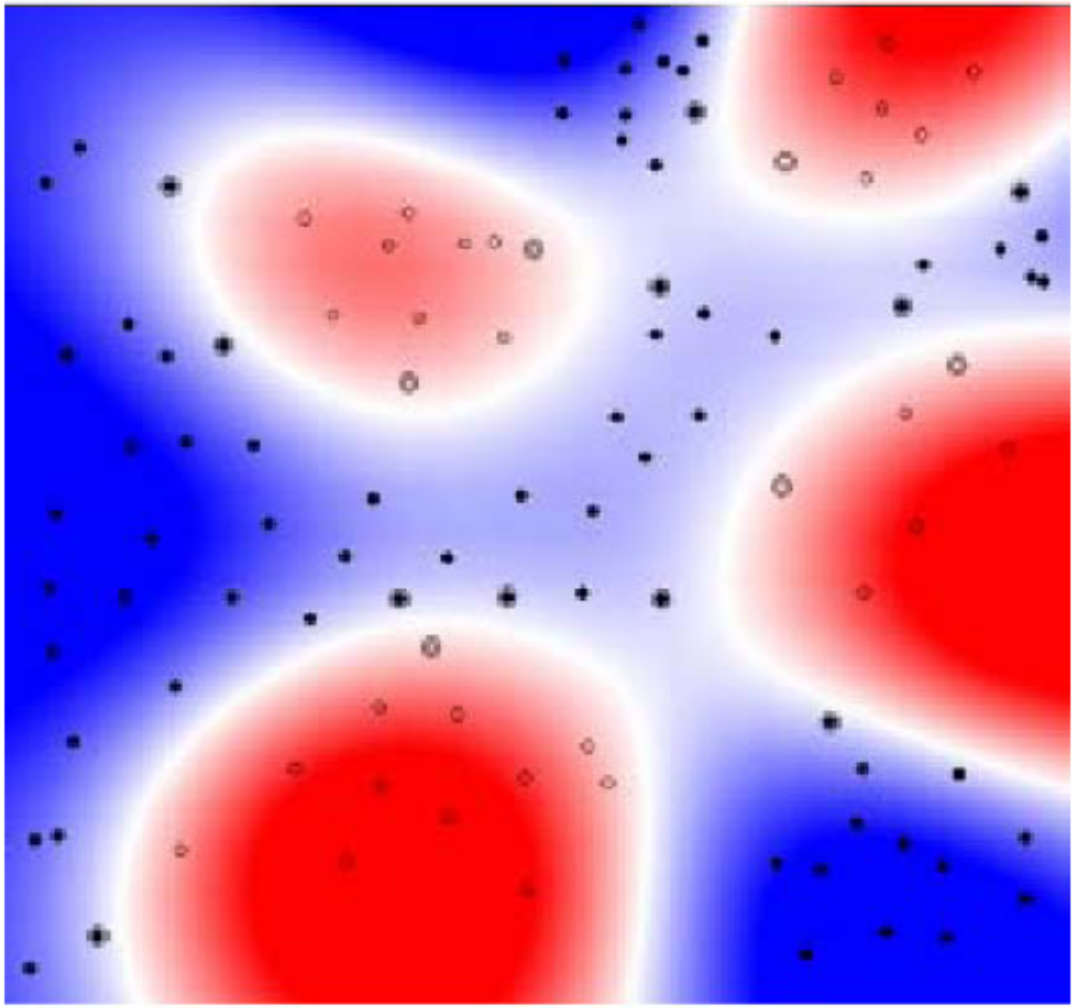
Non-Linear SVM Demo

- http://www.youtube.com/watch?v=3liCbRZPrZA

http://cs.stanford.edu/people/karpathy/svmjs/demo/

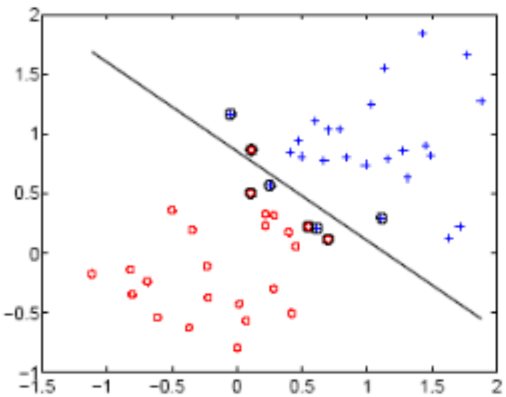


Nonlinear rbf kernel

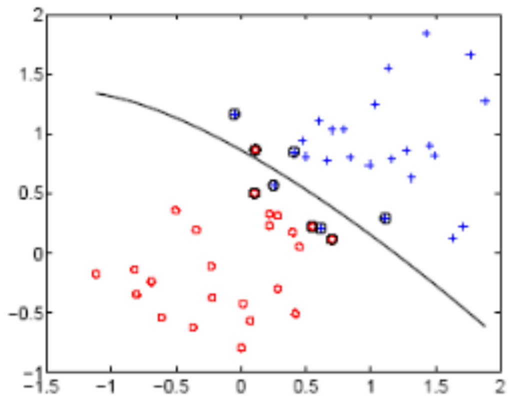


http://www.cs.ucf.edu/courses/cap6412/fall2009/papers/Berwick2003.pdf

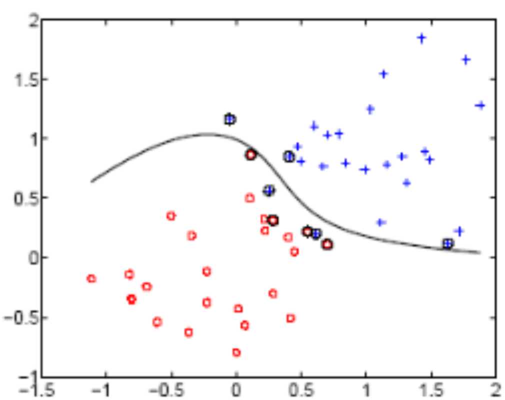
SVM Examples



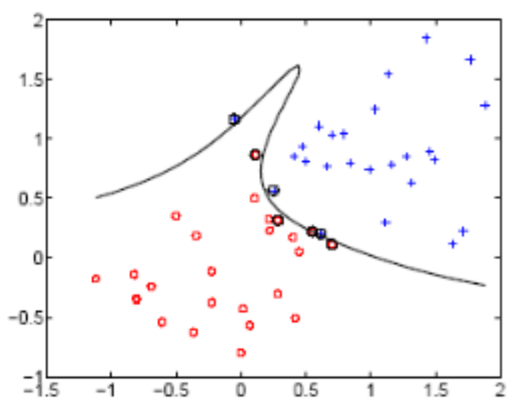
linear



2^{nd} order polynomial

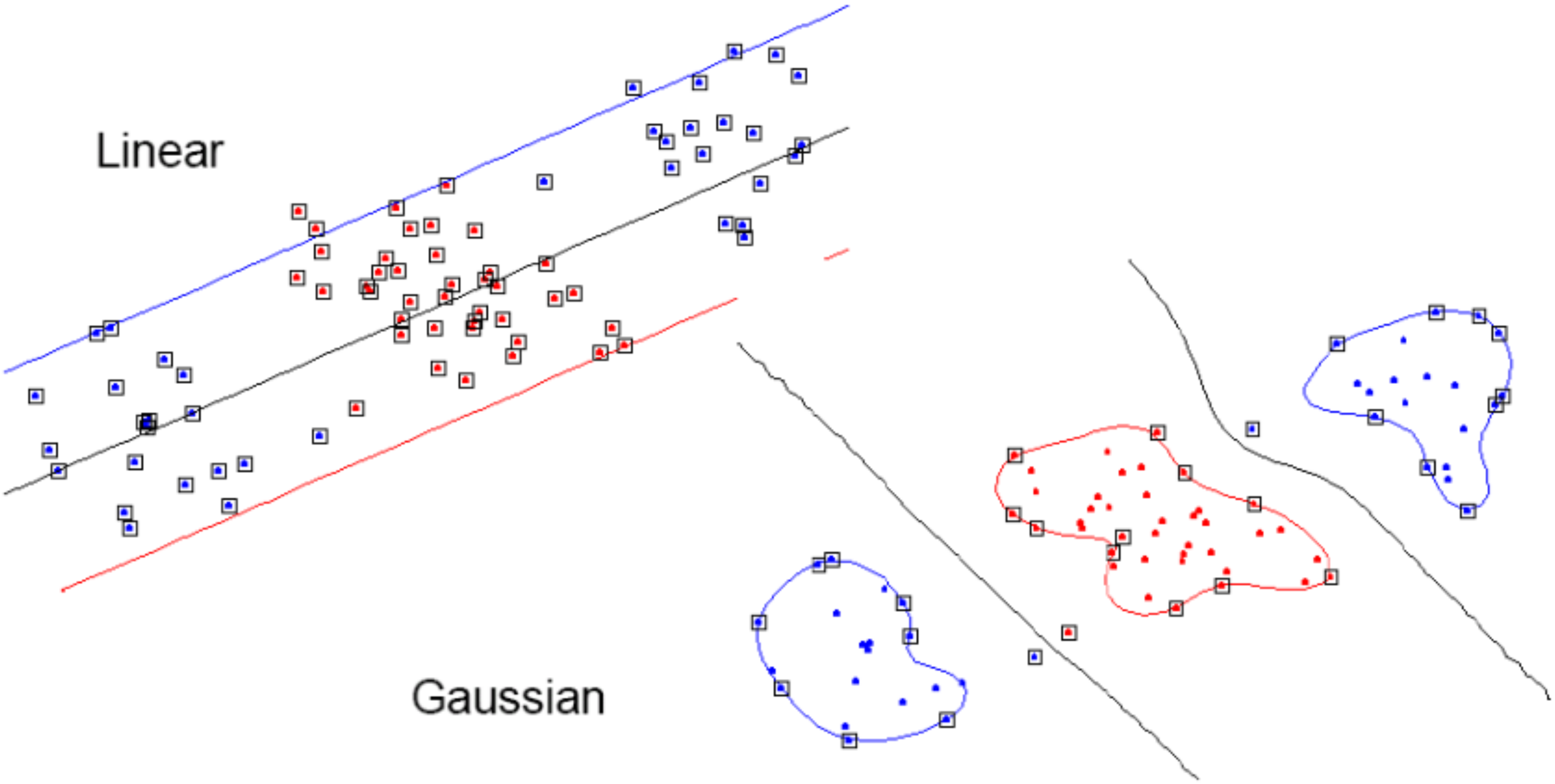


4^{th} order polynomial



8^{th} order polynomial

Gaussian Kernel Examples



SVM Multi-Classification

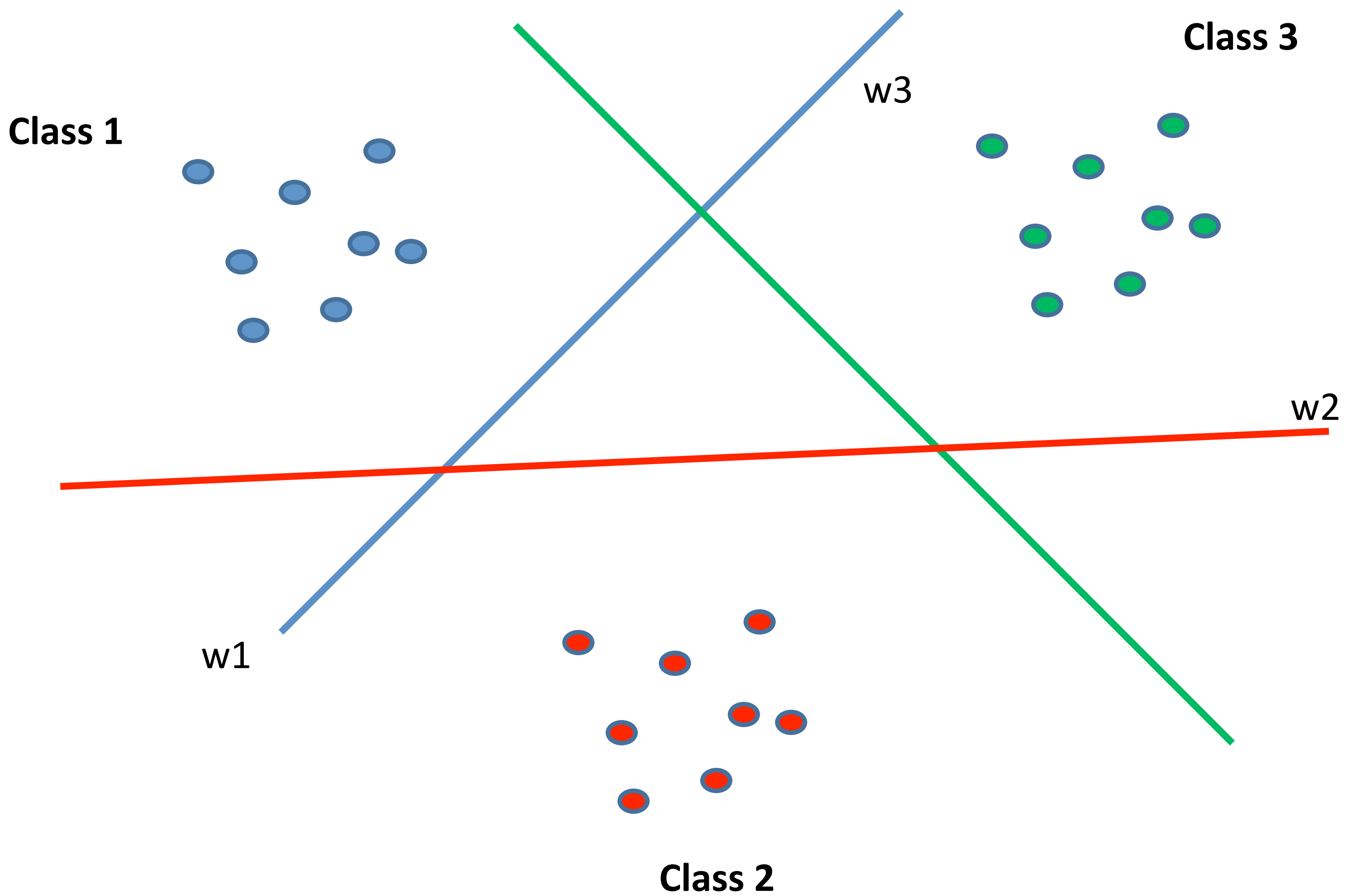
Let $S = \{(\bar{x}_1, y_1), \ldots, (\bar{x}_m, y_m)\}$ be a set of m training examples. We assume that each example $\bar{x}_i$ is drawn from a domain $\mathcal{X} \subseteq \Re^n$ and that each label y_i is an integer from the set $\mathcal{Y} = \{1, \ldots, k\}$. A (multiclass) classifier is a function $H : \mathcal{X} \rightarrow \mathcal{Y}$ that maps an instance $\bar{x}$ to an element y of $\mathcal{Y}$. In this paper we focus on a framework that uses classifiers of the form

$$H_{\mathrm{M}}(\bar{x}) = \arg\max_{r=1}^{k} \{\bar{M}_r \cdot \bar{x}\} ,$$

where $\mathbf{M}$ is a matrix of size $k \times n$ over $\Re$ and $\bar{M}_r$ is the rth row of $\mathbf{M}$. We interchangeably call the value of the inner-product of the rth row of $\mathbf{M}$ with the instance $\bar{x}$ the *confidence* and the *similarity score* for the r class. Therefore, according to our definition above, the predicted label is the index of the row attaining the highest similarity score with $\bar{x}$.

Crammer, Singer, 2001

SVM Multi-Classification



SVM Multi-Classification

$$\min_{M} \quad \frac{1}{2}\|M\|_2^2$$
$$\text{subject to}: \quad \forall i, r \quad \bar{M}_{y_i} \cdot \bar{x}_i + \delta_{y_i,r} - \bar{M}_r \cdot \bar{x}_i \geq 1 \quad .$$

Note that m of the constraints for $r = y_i$ are automatically satisfied since,

$$\bar{M}_{y_i} \cdot \bar{x}_i + \delta_{y_i,y_i} - \bar{M}_{y_i} \cdot \bar{x}_i = 1 \ .$$

Note: here M is the weight matrix

Define the l_2-norm of a matrix $\mathbf{M}$ to be the l_2-norm of the vector represented by the concatenation of $\mathbf{M}$'s rows, $\|M\|_2^2 = \|(\bar{M}_1, \ldots, \bar{M}_k)\|_2^2 = \sum_{i,j} M_{i,j}^2$. Note that if the constraints

Soft Margin Formulation

$$\min_{M,\xi} \quad \frac{1}{2}\beta\|M\|_2^2 + \sum_{i=1}^{m}\xi_i$$

$$\text{subject to : } \quad \forall i, r \quad \bar{M}_{y_i} \cdot \bar{x}_i + \delta_{y_i,r} - \bar{M}_r \cdot \bar{x}_i \geq 1 - \xi_i$$

Crammer, Singer, 2001

Primal Optimization

$$\mathcal{L}(M, \xi, \eta) = \frac{1}{2}\beta \sum_r \|\bar{M}_r\|_2^2 + \sum_{i=1}^{m} \xi_i$$
$$+ \sum_{i,r} \eta_{i,r} \left[\bar{M}_r \cdot \bar{x}_i - \bar{M}_{y_i} \cdot \bar{x}_i - \delta_{y_i,r} + 1 - \xi_i\right]$$
$$\text{subject to :} \quad \forall i, r \quad \eta_{i,r} \geq 0 \quad .$$

Crammer, Singer, 2001

Dual Optimization

$$Q(\eta) = -\frac{1}{2}\beta^{-1}\sum_{i,j}(\bar{x}_i \cdot \bar{x}_j)\left[\sum_r (\delta_{y_i,r} - \eta_{i,r})(\delta_{y_j,r} - \eta_{j,r})\right] - \sum_{i,r}\eta_{i,r}\delta_{y_i,r}$$

Crammer, Singer, 2001

Dual Optimization

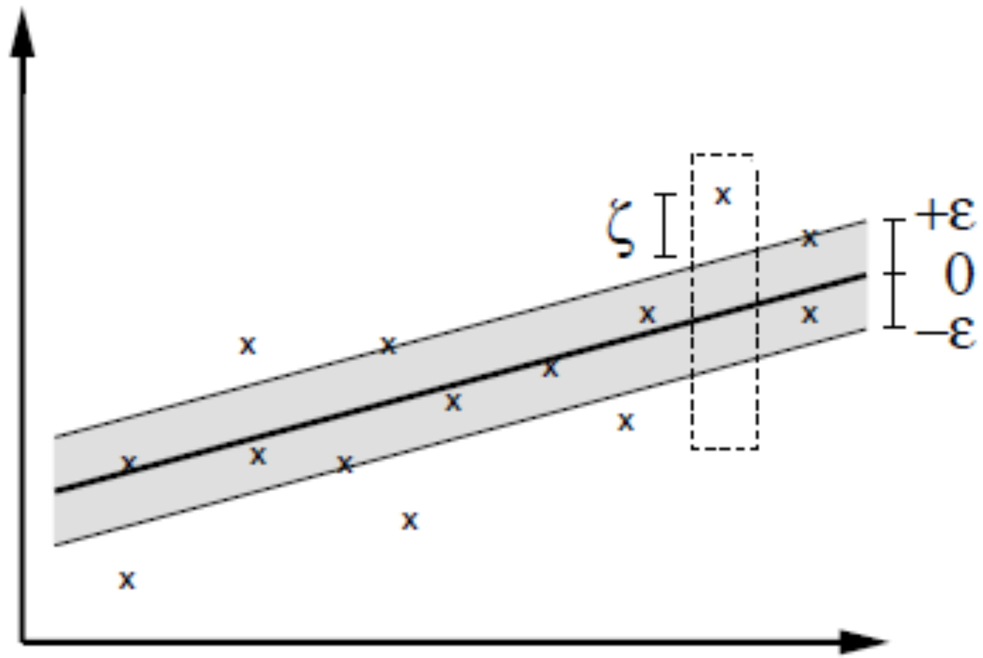
$$Q(\eta) \;=\; -\frac{1}{2}\beta^{-1}\sum_{i,j}(\bar{x}_i \cdot \bar{x}_j)\left[\sum_r (\delta_{y_i,r} - \eta_{i,r})(\delta_{y_j,r} - \eta_{j,r})\right] - \sum_{i,r}\eta_{i,r}\delta_{y_i,r}$$

How to extend it to non-linear multi-classification problem?

Crammer, Singer, 2001

SVM Regression

Regression: *f*(x) = wx + b



Smola and Scholkopf, 2003

Hard Margin Formulation

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}||w||^2 \\ \text{subject to} & \left\{ \begin{array}{lll} y_i - \langle w, x_i \rangle - b & \leq & \varepsilon \\ \langle w, x_i \rangle + b - y_i & \leq & \varepsilon \end{array} \right. \end{array}$$

Questions: can both constraints associated with the same data point be violated at the same time?

Smola and Scholkopf, 2003

Software Margin Formulation

$$\text{minimize} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*)$$

$$\text{subject to} \quad \begin{cases} y_i - \langle w, x_i\rangle - b & \leq \quad \varepsilon + \xi_i \\ \langle w, x_i\rangle + b - y_i & \leq \quad \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geq \quad 0 \end{cases}$$

The constant $C > 0$ determines the trade-off between the flatness of f and the amount up to which deviations larger than ε are tolerated.

Smola and Scholkopf, 2003

Primal Optimization

$$
\begin{aligned}
L \quad := \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*) - \sum_{i=1}^{\ell}(\eta_i\xi_i + \eta_i^*\xi_i^*) \\
& - \sum_{i=1}^{\ell}\alpha_i(\varepsilon + \xi_i - y_i + \langle w, x_i\rangle + b) \\
& - \sum_{i=1}^{\ell}\alpha_i^*(\varepsilon + \xi_i^* + y_i - \langle w, x_i\rangle - b)
\end{aligned}
$$

Here L is the Lagrangian and $\eta_i, \eta_i^*, \alpha_i, \alpha_i^*$ are Lagrange multipliers. Hence the dual variables in (5) have to satisfy positivity constraints, i.e.

$$\alpha_i^{(*)}, \eta_i^{(*)} \geq 0.$$

Note that by $\alpha_i^{(*)}$, we refer to α_i and α_i^*.

Smola and Scholkopf, 2003

Dual Optimization

$$
\begin{aligned}
\partial_b L &= \textstyle\sum_{i=1}^{\ell}(\alpha_i^* - \alpha_i) &= 0 \quad (7)\\
\partial_w L &= w - \textstyle\sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)x_i &= 0 \quad (8)\\
\partial_{\xi_i^{(*)}} L &= C - \alpha_i^{(*)} - \eta_i^{(*)} &= 0 \quad (9)
\end{aligned}
$$

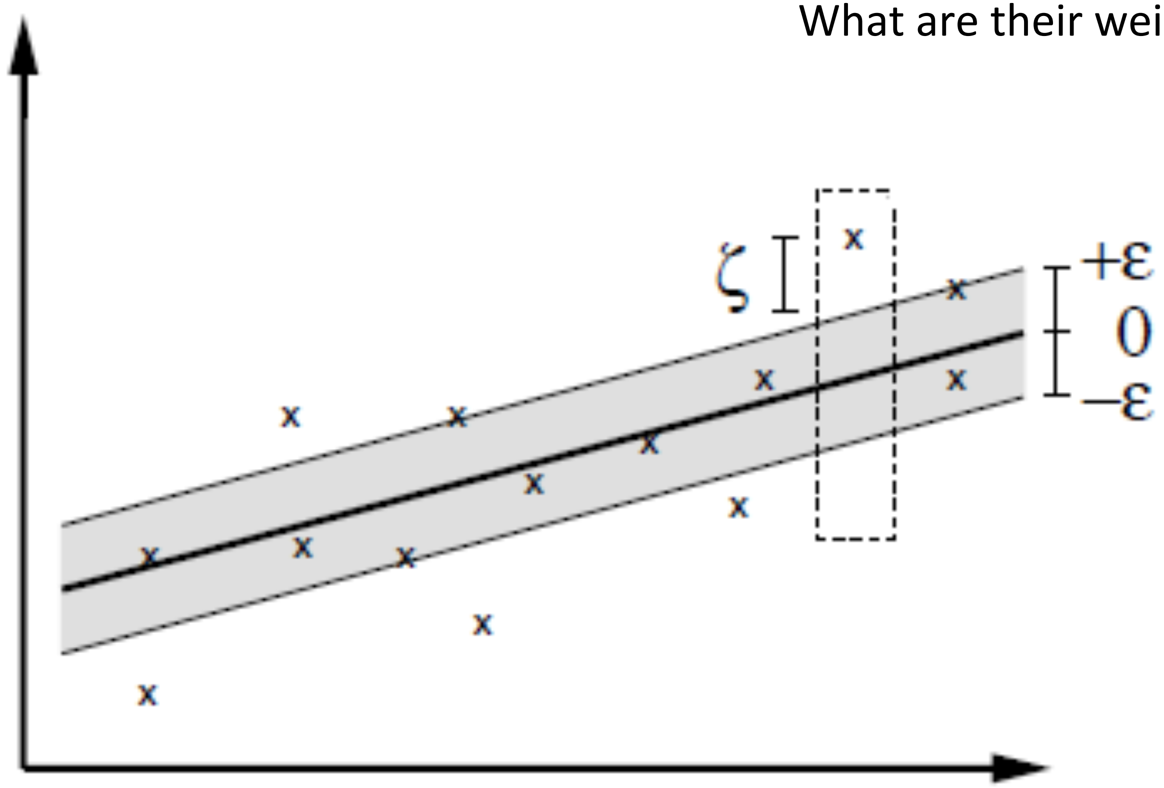
Substituting (7), (8), and (9) into (5) yields the dual optimization problem.

$$
\begin{aligned}
\text{maximize} \quad & \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{\ell}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j\rangle \\ -\varepsilon\sum_{i=1}^{\ell}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{\ell} y_i(\alpha_i - \alpha_i^*) \end{cases} \quad (10)\\
\text{subject to} \quad & \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*) = 0 \text{ and } \alpha_i, \alpha_i^* \in [0, C]
\end{aligned}
$$

Smola and Scholkopf, 2003

Support Vectors and Weights

Which data points are support vectors?
What are their weights?



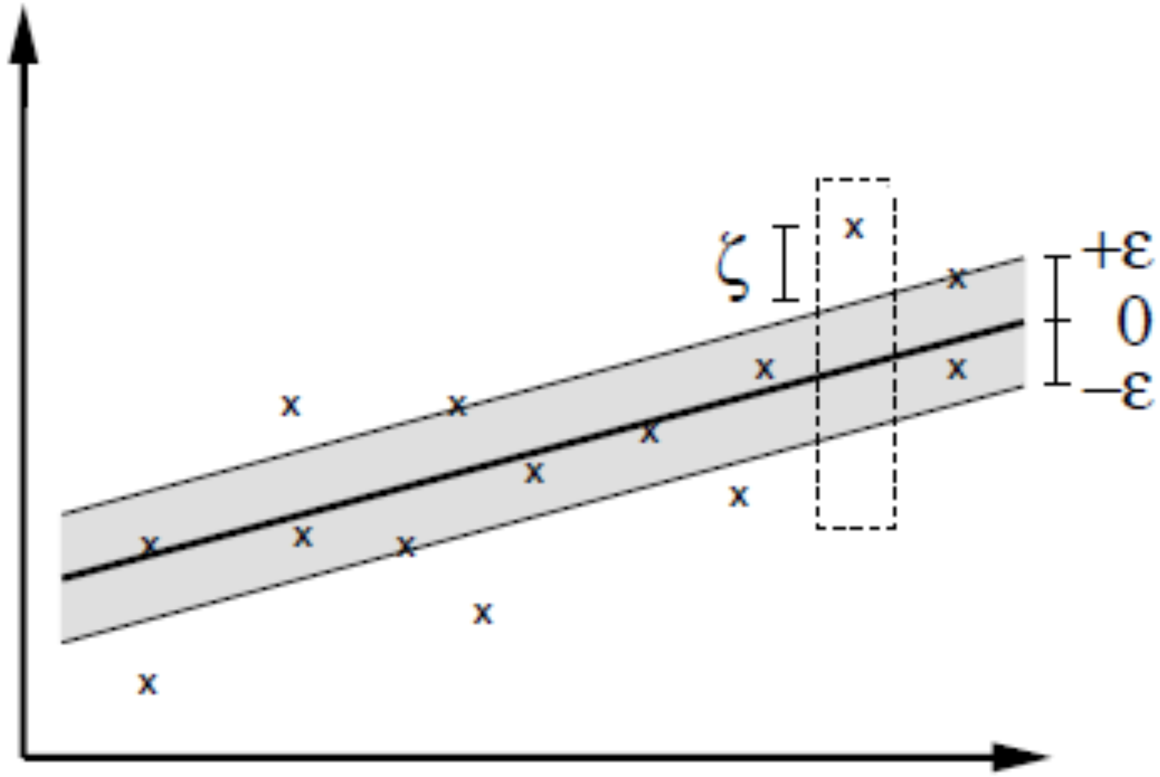
Smola and Scholkopf, 2003

Complementary Slackness

$$\begin{aligned}
\alpha_i(\varepsilon + \xi_i - y_i + \langle w, x_i \rangle + b) &= 0 \\
\alpha_i^*(\varepsilon + \xi_i^* + y_i - \langle w, x_i \rangle - b) &= 0 \\
(C - \alpha_i)\xi_i &= 0 \\
(C - \alpha_i^*)\xi_i^* &= 0.
\end{aligned}$$

Smola and Scholkopf, 2003

Support Vectors



Which data points are support vectors and what are their weights?

Computing b

- How?
- Can any support vector have both a, a* non-zero?

SVM for Non-Linear Regression

$$\text{maximize} \quad \begin{cases} -\frac{1}{2} \sum_{i,j=1}^{\ell} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) k(x_i, x_j) \\ -\varepsilon \sum_{i=1}^{\ell} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{\ell} y_i(\alpha_i - \alpha_i^*) \end{cases}$$

$$\text{subject to} \quad \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*) = 0 \text{ and } \alpha_i, \alpha_i^* \in [0, C]$$

Likewise the expansion of f (11) may be written as

$$w = \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*) \Phi(x_i) \text{ and } f(x) = \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*) k(x_i, x) + b.$$

Properties of SVM

- **Flexibility in choosing a similarity function**
- **Sparseness of solution when dealing with large data sets**
  - only support vectors are used to specify the separating hyperplane
- **Ability to handle large feature spaces**
  - complexity does not depend on the dimensionality of the feature space
- **Overfitting can be controlled by soft margin approach**
- **Nice math property: a simple convex optimization problem which is guaranteed to converge to a single global solution**
- **Feature Selection**
- **Sensitive to noise**

M. Tan's slides, Univ. British Columbia

SVM Applications

- **SVM has been used successfully in many real-world problems**

  - text and hypertext categorization

  - image classification

  - bioinformatics (protein classification, cancer classification)

  - hand-written character recognition

M. Tan's slides, Univ. British Columbia

Application 1: Cancer Classification

- High Dimensional
  - g>1000; n<100

Genes				
Patients	g-1	g-2	......	g-p
P-1				
p-2				
....				
p-n				

- Imbalanced
  - less positive samples

- Many irrelevant features
- Noisy

SVM is sensitive to noisy (mis-labeled) data ☹

FEATURE SELECTION

In the linear case,
w_i^2 gives the ranking of dim i

M. Tan's slides, Univ. British Columbia

Application 2: Text Categorization

- Task: The classification of natural text (or hypertext) documents into a fixed number of predefined categories based on their content.

  - email filtering, web searching, sorting documents by topic, etc..

- A document can be assigned to more than one category, so this can be viewed as a series of binary classification problems, one for each category

M. Tan's slides, Univ. British Columbia

Representation of Text

IR's vector space model (aka bag-of-words representation)

- A doc is represented by a vector indexed by a pre-fixed set or dictionary of terms
- Values of an entry can be binary or weights

$$\phi_i(x) = \frac{\mathrm{tf}_i \log(\mathrm{idf}_i)}{\kappa},$$

- Normalization, stop words, word stems
- Doc x => $\boldsymbol{\phi}(x)$

M. Tan's slides, Univ. British Columbia

Text Categorization using SVM

- **The similarity between two documents is $\phi(x) \cdot \phi(z)$**

- **$K(x,z) = \langle \phi(x) \cdot \phi(z)$ is a valid kernel, SVM can be used with $K(x,z)$ for discrimination.**

- **Why SVM?**

  -High dimensional input space

  -Few irrelevant features (dense concept)

  -Sparse document vectors (sparse instances)

  -Text categorization problems are linearly separable

M. Tan's slides, Univ. British Columbia

Some Issues

- **Choice of kernel**
  - Gaussian or polynomial kernel is default
  - if ineffective, more elaborate kernels are needed
  - domain experts can give assistance in formulating appropriate similarity measures

- **Choice of kernel parameters**
  - e.g. σ in Gaussian kernel
  - σ is the distance between closest points with different classifications
  - In the absence of reliable criteria, applications rely on the use of a validation set or cross-validation to set such parameters.

- **Optimization criterion** – Hard margin v.s. Soft margin
  - a lengthy series of experiments in which various parameters are tested

M. Tan's slides, Univ. British Columbia

Additional Resources

- **An excellent tutorial on VC-dimension and Support Vector Machines:**

  C.J.C. Burges. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2):955-974, 1998.

- **The VC/SRM/SVM Bible:**

  Statistical Learning Theory by Vladimir Vapnik, Wiley-Interscience; 1998

http://www.kernel-machines.org/

M. Tan's slides, Univ. British Columbia

SVM Tools

- SVM-light: http://svmlight.joachims.org/
- LIBSVM: http://www.csie.ntu.edu.tw/~cjlin/libsvm/
- Gist: http://bioinformatics.ubc.ca/gist/
- More: http://www.kernel-machines.org/software.html

M. Tan's slides, Univ. British Columbia