

Lecture 22: Support Vector Machines

CSI 5v93: Introduction to machine learning

Baylor University
Computer Science Department

Dr. Greg Hamerly
<http://cs.baylor.edu/~hamerly/>

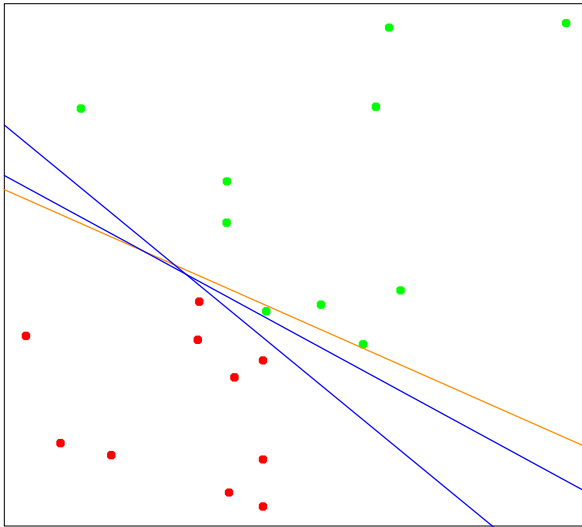
CSI 5v93: Introduction to machine learning, Lecture 22 – p. 1/17

Questions?

CSI 5v93: Introduction to machine learning, Lecture 22 – p. 2/17

Separating hyperplanes

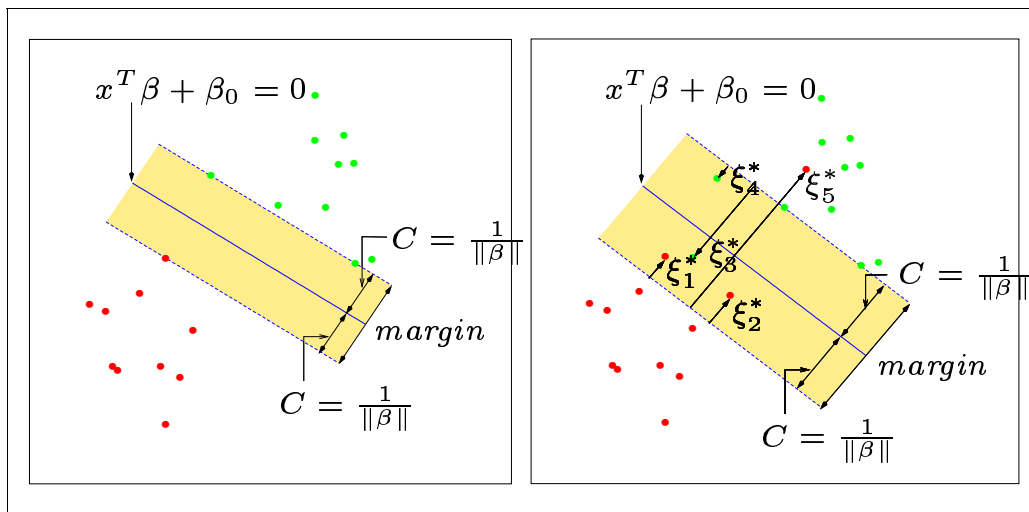
Here is a two-class classification problem, with several linear boundaries:



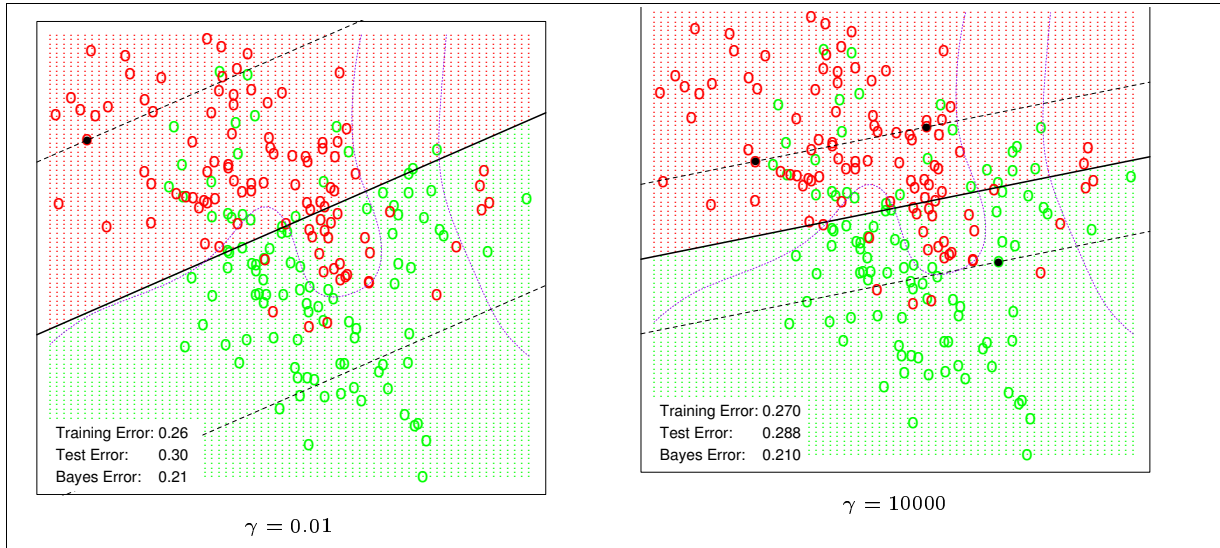
Note that there are many solutions to use a linear boundary to separate the classes.

Separable and Non-separable data

$$y_i(x_i^T \beta + \beta_0) \geq C(1 - \xi_i), \quad \xi_i \geq 0, \quad \sum_{i=1}^n \xi_i \leq \text{constant}$$



Support vector classifier example



CSI 5v93: Introduction to machine learning, Lecture 22 – p. 5/17

Next: flexible (non-linear) support vector machines

Question: How do we better handle data that is not linearly separable with support vector machines?

Answer: Transform the data into a higher dimension where the data is more easily separated!

CSI 5v93: Introduction to machine learning, Lecture 22 – p. 6/17

Moving to higher dimensions: basis expansions

We want to move the data into a higher dimensional space with several basis expansion functions:

$$x_i \rightarrow h(x_i) = (h_1(x_i), h_2(x_i), \dots, h_M(x_i))$$

For example, we might have $x_i \in \mathbb{R}^2$, and choose $M = 4$ basis functions:

$$\begin{aligned}h_1(x) &= 1 \\h_2(x) &= x_1^2 \\h_3(x) &= x_2^2 \\h_4(x) &= x_1 x_2\end{aligned}$$

Then we would learn a β that lives in this higher-dimensional space that separates the classes.

Key idea: transforming into a higher dimension allows β to represent a linear hyperplane in the higher dimensional space, which is non-linear curve in the original dimensions.

Example of moving into higher-dimensional spaces

Rewriting in terms of inner products

Note that we have the Lagrangian form

$$\begin{aligned}L_D &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j h(x_i)^T h(x_j) \\ &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \langle h(x_i), h(x_j) \rangle\end{aligned}$$

And this depends only on the inner product (aka dot product) of $h(x_i)$ and $h(x_j)$; not on h directly.

Note also that we could write $f(x)$ as:

$$\begin{aligned}f(x) &= h(x)^T \beta + \beta_0 \\ &= \sum_{i=1}^n \alpha_i y_i \langle h(x), h(x_i) \rangle + \beta_0\end{aligned}$$

So this also depends only on the dot product, not on h .

The “kernel trick”

We can rewrite all the equations we need for SVMs in terms of dot products in the higher-dimensional space.

It turns out that we can compute these dot products in the high-dimensional space without using the basis expansions!

All we need to know is the kernel function:

$$K(x, y) = \langle h(x), h(y) \rangle$$

Our h could go to many dimensions (even infinite!), but we don't need to actually go there.

Kernel properties

A kernel

- represents an inner product in some dimension (usually a higher dimension)
- should be positive semi-definite

A kernel doesn't have to have a nice basis expansion form!

All you need for a kernel is to define a positive semidefinite function K that does:

$$K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$$

The Support Vector Machine – summary

Given a kernel function $k(\cdot, \cdot)$ and an error parameter γ , maximize

$$L_D = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$

subject to

$$0 \leq \alpha_i \leq C, \quad \sum_{i=1}^n \alpha_i y_i = 0$$

And the decision function for the classifier is:

$$g(x) = \text{sgn} \left(\sum_{i=1}^n \alpha_i y_i k(x, x_i) + \beta_0 \right)$$

Where did β go?

Example kernels

polynomial kernel (of degree d):

$$K(x, x') = (1 + \langle x, x' \rangle)^d$$

radial basis kernel:

$$K(x, x') = \exp(-\|x - x'\|^2/c)$$

neural network kernel:

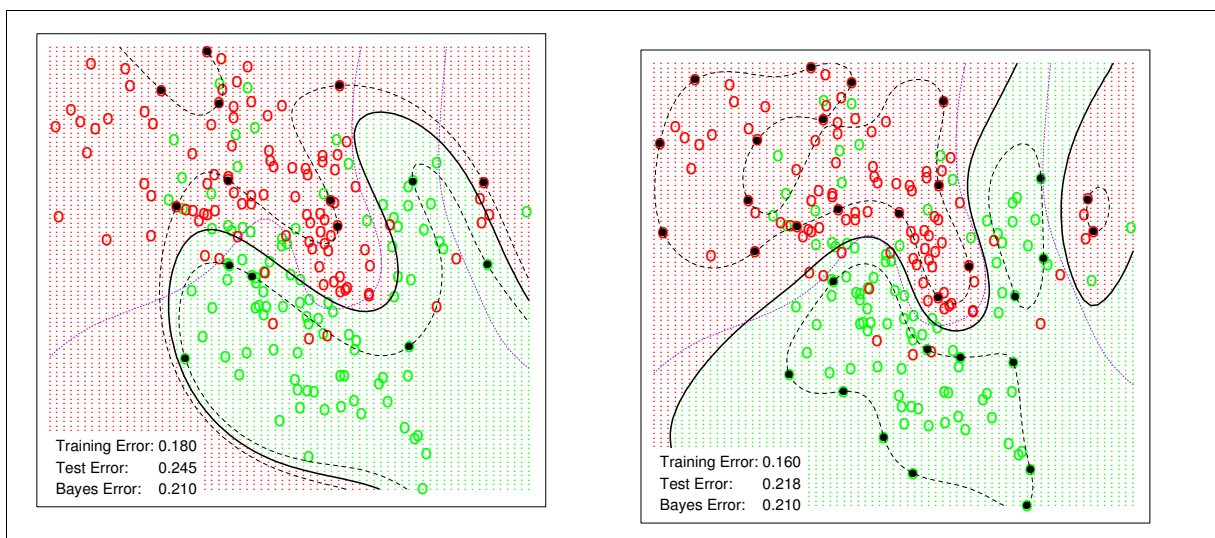
$$K(x, x') = \tanh(\kappa_1 \langle x, x' \rangle + \kappa_2)$$

Representing the space for the polynomial kernel

polynomial kernel (of degree d):

$$K(x, x') = (1 + \langle x, x' \rangle)^d$$

Examples



CSI 5v93: Introduction to machine learning, Lecture 22 – p. 15/17

Using SVMs

You should (probably) never implement an SVM on your own; they are tricky.

Fortunately, there is a lot of good software out there for SVM learning.

When using an SVM learner, you must choose:

- kernel
- error penalty γ

Problems: not intuitive how to choose the kernel or γ

Advantages: SVMs often perform amazingly well

CSI 5v93: Introduction to machine learning, Lecture 22 – p. 16/17

2-minute journal

Please write a response to the following on a piece of paper and hand it in immediately. Please make it anonymous (no names). Write about:

- major points you learned today
- areas not understood or requiring clarification