

# CSE 6740 Lecture 11

## *How Can I Learn Fancier (Nonlinear) Models? (Kernelization)*

Alexander Gray

agray@cc.gatech.edu

Georgia Institute of Technology

# Quiz

1. VC theory does not give you an estimate of the test error, just an upper bound on it. T.
2. In the bootstrap, we draw points from the dataset without replacement, i.e. the same point cannot be drawn twice. F.
3. In expectation, stacking never does worse than the best single model. T.

# Today

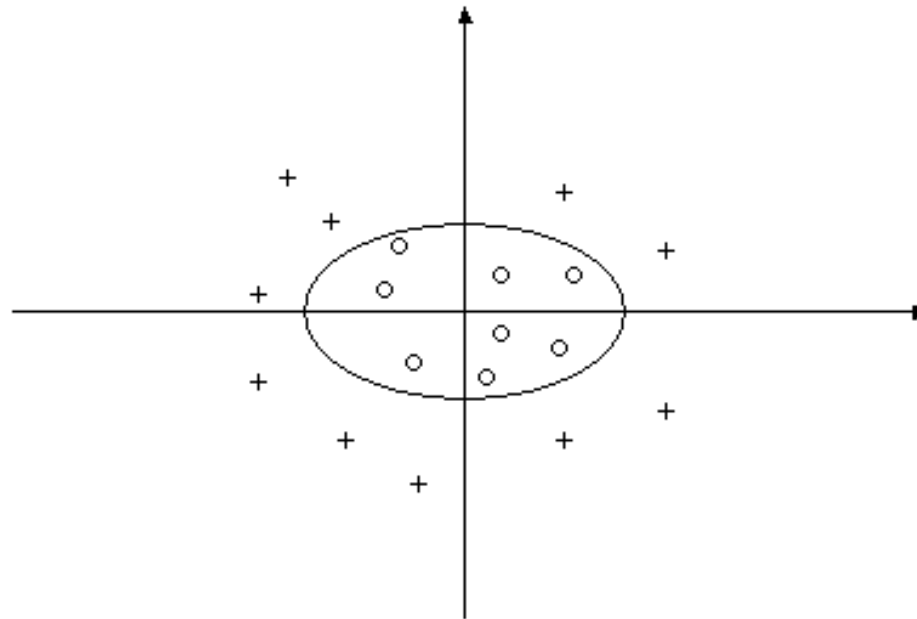
1. How to make more complex models using kernels
2. Theory motivating kernels

# More Complex Models, Using Kernels

Why kernels, part I. “Because we can get richer models, yet leave the methods the same.”

# Generalized Linear Models

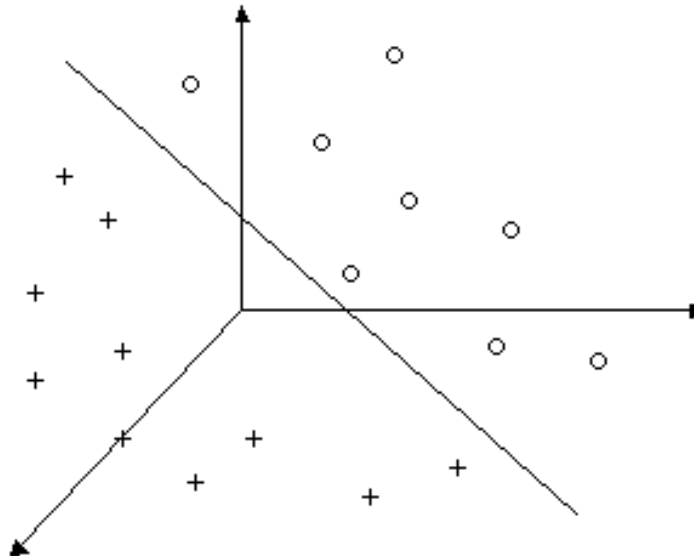
Suppose we have data  $\{(x, y)\}_i$  where each  $x \in \mathcal{X} = \mathbb{R}^2$  is a vector  $(x_1, x_2)$  like the following. Then the classes cannot be separated by a linear decision boundary.



# Generalized Linear Models

Now let's make a transformed dataset  $\{(z, y)\}_i$  where

$$z = \phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2). \quad (1)$$



# Generalized Linear Models

Thus  $\phi$  is a map from  $\mathcal{X} = \mathbb{R}^2$  to  $\mathcal{Z} = \mathbb{R}^3$ . In the new space, the data are linearly separable.

So here a linear classifier in a higher-dimensional space corresponds to a nonlinear classifier in the original space.

Thus we get to leave our learning method exactly as it was.

# Generalized Linear Models

We can do this with any linear model, including for example linear regression, where this is called *generalized linear regression*, to effectively get a more powerful model class.

However, there are drawbacks to this. We don't know in advance which features need to be constructed. Thus we might want to consider all possible products of the features, for example. But even considering all possible quadruplets of features, if  $D=256$ , yields 183,181,376 features in the transformed space.

# Kernel Trick

Now suppose we have a model that can be represented in terms of only dot products between points,  $\langle x, \tilde{x} \rangle$ . Now notice that the inner product in  $\mathcal{Z}$  can be written

$$\langle z, \tilde{z} \rangle = \langle \phi(x), \phi(\tilde{x}) \rangle \quad (2)$$

$$= x_1^2 \tilde{x}_1^2 + 2x_1 \tilde{x}_1 x_2 \tilde{x}_2 + x_2^2 \tilde{x}_2^2 \quad (3)$$

$$= (\langle x, \tilde{x} \rangle)^2 \quad (4)$$

$$\equiv K(x, \tilde{x}). \quad (5)$$

Thus we can compute  $\langle z, \tilde{z} \rangle$  without ever actually computing  $z_i = \phi(x_i)$ .

# Kernel Trick

In summary, this so-called *kernel trick* involves finding a mapping  $\phi : \mathcal{X} \rightarrow \mathcal{Z}$  and a learning method such that

- $\mathcal{Z}$  has higher dimension than  $\mathcal{X}$ , effectively leading to a richer set of models
- The method's algorithm only requires computing inner products, as far as how the data appear in the method
- There is a kernel function  $K$  such that  $\langle \phi(x), \phi(\tilde{x}) \rangle = K(x, \tilde{x})$ .
- Everywhere the term  $\langle x, \tilde{x} \rangle$  appears in the algorithm, replace it with  $K(x, \tilde{x})$ .

This is also called *kernelizing* the original method.

# Kernel Trick

In fact, we never need to explicitly construct the mapping  $\phi$  at all. We only need to specify a kernel  $K(x, \tilde{x})$  that corresponds to  $\langle \phi(x)\phi(\tilde{x}) \rangle$  for some  $\phi$ .

One question of interest, then, is the following: Given a function of two variables  $K(x, \tilde{x})$ , does there exist a function  $\phi(x)$  such that  $K(x, \tilde{x}) = \langle \phi(x)\phi(\tilde{x}) \rangle$ ?

# Mercer's Theorem

*Mercer's Theorem* says, roughly, that if  $K$  is positive definite, *i.e.*

$$\int \int K(x, \tilde{x}) g(x) g(\tilde{x}) dx d\tilde{x} \geq 0 \quad (6)$$

for square integrable functions  $g$ , *i.e.*

$$\int g^2(x) dx < \infty \quad (7)$$

then such a  $\phi$  exists for  $K$ .

# Mercer Kernels

Some commonly used kernels having this property, or *Mercer kernels*, are:

$$\text{Gaussian } K(x, \tilde{x}) = \exp \left\{ -\|x - \tilde{x}\|^2 / 2a^2 \right\} \quad (8)$$

$$\text{polynomial } K(x, \tilde{x}) = (\langle x, \tilde{x} \rangle + a)^b \quad (9)$$

$$\text{sigmoid } K(x, \tilde{x}) = \tanh (a \langle x, \tilde{x} \rangle + b) \quad (10)$$

There are also Mercer kernels for comparing non-vector objects, like strings and graphs.

The *kernel matrix* or *Gram matrix*  $\mathbf{K}$  is the  $N \times N$  matrix having entries  $\mathbf{K}_{ij} = K(x_i, x_j)$ .

# Kernelizing the SVM

The support vector machine is a method whose algorithm contains the data only in terms of their dot products with each other. Thus we can replace the objective function

$$\sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{i'=1}^N \alpha_i \alpha_{i'} y_i y_{i'} \langle x_i x_{i'} \rangle \quad (11)$$

with

$$\sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{i'=1}^N \alpha_i \alpha_{i'} y_i y_{i'} K(x_i, x_{i'}). \quad (12)$$

Before the discriminant function was  $g(x) = \beta^T x + \beta_0$ . Now it is  $g(x) = \beta^T \phi(x) + \beta_0$ .

# Theory Motivating Kernels

Why kernels, part II. “Because everything boils down to kernels.”

# Regularization Theory

Consider minimizing a regularized loss function:

$$\min_{f \in \mathcal{F}} \left[ \sum_{i=1}^N L(y_i, f(x_i)) + \lambda J(f) \right] \quad (13)$$

where  $J(f)$  is a penalty functional, and  $\mathcal{F}$  is the space of functions on which  $J(f)$  is defined.

# Regularization Theory

Consider a general class of penalty functionals having the form

$$J(f) = \int_{\mathbb{R}^D} \frac{|\check{f}(s)|^2}{\check{K}(s)} ds \quad (14)$$

where  $\check{f}$  denotes the Fourier transform of  $f$ , and  $\check{K}$  is some positive function which falls off to zero as  $\|s\| \rightarrow \infty$ . The idea is that  $1/\check{K}$  increases the penalty for high-frequency components of  $f$ .

# Regularization Theory

It can be shown under certain assumptions that the solutions have the form

$$f(x) = \sum_{i=1}^N \alpha_i K(\|x - x_i\|) + \sum_{j=1}^M \beta_j \psi_j(x) \quad (15)$$

where the  $\psi_j$  span the null space of the penalty functional  $J$ , and  $K$  is the inverse Fourier transform of  $\check{K}$ .

Interestingly:

- While the minimization is over an infinite space, the solution is finite-dimensional.
- The solution has the form of a weighted sum of kernel functions.

# Reproducing Kernel Hilbert Spaces

Consider a subclass of this kind of regularization problem, corresponding to certain kernel functions  $K$ . The penalty functional  $J$  is defined in terms of the kernel also.

The corresponding space of functions  $\mathcal{F}_K$  is called a *reproducing kernel Hilbert space*.

Suppose that  $K$  has an eigen-expansion

$$K(x, \tilde{x}) = \sum_{j=1}^{\infty} \gamma_j \psi_j(x) \psi_j(\tilde{x}) \quad (16)$$

with  $\gamma_j \geq 0$ ,  $\sum_{j=1}^{\infty} \gamma_j^2 < \infty$ .

# Reproducing Kernel Hilbert Spaces

Elements of  $\mathcal{F}_K$  have an expansion in terms of these eigen-functions,

$$f(x) = \sum_{j=1}^{\infty} \beta_j \psi_j(x) \quad (17)$$

with the constraint that

$$\|f\|_{\mathcal{F}_K}^2 \equiv \sum_{j=1}^{\infty} \beta_j^2 / \gamma_j < \infty \quad (18)$$

where  $\|f\|_{\mathcal{F}_K}$  is called the norm induced by  $K$ . The penalty functional for the space  $\mathcal{F}_K$  is defined to be the squared norm  $J(f) = \|f\|_{\mathcal{F}_K}^2$ . This penalizes functions with large eigenvalues more.

# Reproducing Kernel Hilbert Spaces

Thus we are talking about a special case of regularized loss minimization:

$$\min_{f \in \mathcal{F}} \left[ \sum_{i=1}^N L(y_i, f(x_i)) + \lambda \|f\|_{\mathcal{F}_K}^2 \right] \quad (19)$$

$$= \min_{\{\beta_j\}} \left[ \sum_{i=1}^N L \left( y_i, \sum_{j=1}^{\infty} \beta_j \psi_j(x) \right) + \lambda \sum_{j=1}^{\infty} \beta_j^2 / \gamma_j \right]. \quad (20)$$

It can be shown that the solution is finite-dimensional, having the form

$$f(x) = \sum_{i=1}^N \alpha_i K(x, x_i). \quad (21)$$

# RKHS: Representing and Reproducing

The kernel as a function of its first argument

$$\phi_i(x) \equiv K(x, x_i) \quad (22)$$

is called the *representer of evaluation* at  $x_i$  in  $\mathcal{F}_K$ , since for  $f \in \mathcal{F}_K$ , it is easily seen that

$$\langle K(\cdot, x_i), f \rangle_{\mathcal{F}_K} = f(x_i). \quad (23)$$

Similarly,

$$\langle K(\cdot, x_i), K(\cdot, x_{i'}) \rangle_{\mathcal{F}_K} = K(x_i, x_{i'}), \quad (24)$$

which is called the *reproducing* property of  $\mathcal{F}_K$ .

# RKHS: Minimization Problem

It is a consequence that

$$J(f) = \sum_{i=1}^N \sum_{i'=1}^N K(x_i, x_{i'}) \alpha_i \alpha_{i'} \quad (25)$$

for  $f(x) = \sum_{i=1}^N \alpha_i K(x, x_i)$ .

Our minimization problem then has the form

$$\min_{\alpha} L(\mathbf{y}, \mathbf{K}\alpha) + \lambda \alpha^T \mathbf{K}\alpha. \quad (26)$$

# RKHS: SVM Example

The support vector machine's optimization problem is equivalent to

$$\min_{\alpha} \left[ \sum_{i=1}^N (1 - y_i g(x_i))_+ + \lambda \|\beta\|^2 \right] \quad (27)$$

$$= \min_{\alpha} \left[ \sum_{i=1}^N (1 - y_i g(x_i))_+ + \lambda \alpha^T \mathbf{K} \alpha \right]. \quad (28)$$

We can kernelize linear regression, obtaining something called *kriging*, which can be reinterpreted slightly in a Bayesian way as something called *Gaussian process regression*.

# Main Things You Should Know

- What the kernel trick (or kernelization) is
- What the point of the kernel trick is

# Quiz

1. (T/F) The kernel trick can be applied to any machine learning method.
2. (T/F) Any similarity function is a kernel.