Kernel Methods

Benjamin Recht

April 4, 2005

Gameplan

- Function Fitting
- Linear Regression
- Kernels and norms
- Nonlinear Regression
- Semi-supervised learning

Learning from Sparse Data

Suppose we want to find a functional mapping from one set X to another set Y but we are only given pairs of data points $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_L, y_L)$.

- What is the best $f : X \to Y$ that agrees with our data?
- What is the best f which generalizes to a new data point $\mathbf{x}_{new}$?
- What are efficient algorithms to approximate this best f with only knowledge of the data?

Risk Minimization

First we need to define a notion of “best.” Let us say that, pointwise, the cost for making an error is a function is given by a convex cost function $C(f(x), y)$.

$$f^* = \arg\min_f \int_{X,Y} C(f(\mathbf{x}), y) p(\mathbf{x}, y) d\mathbf{x} dy$$

would be the ideal function in this case. The cost function is called the *risk*.

Empirical Risk Minimization

However, if we don't know $p(\mathbf{x}, y)$ and can't estimate $p(\mathbf{x}, y)$ because density estimation is generically difficult, we can't find this f^*. However, we might be able to find

$$f_L^* = \arg\min_f \sum_{i=1}^{L} C(f(\mathbf{x}_i), y_i)$$

This cost function is called the *empirical risk*.

Hypothesis Spaces

In the limit of infinite data, the law of large numbers says that the empirical risk converges to the true risk exponentially. However, we are still left with the problem that there is no good way to search over the set of all functions. To fix this, we can restrict f to a specific class of functions $\mathcal{H}$ called the hypothesis space. Then we can try to find

$$f_L^* = \arg\min_{f \in \mathcal{H}} \sum_{i=1}^{L} C(f(\mathbf{x}_i), y_i)$$

Regularization

Even here, there are usually too many available f_L^* which fit the data in the hypothesis space. We get around this problem by introducing a measure of smoothness $\|f\|$ and try to search for a reasonably smooth function which fits the data

$$f_L^* = \arg\min_{f \in \mathcal{H}} \sum_{i=1}^{L} C(f(\mathbf{x}_i), y_i) + \lambda \|f\|^2$$

Function fitting

$$f_L^* = \arg\min_{f \in \mathcal{H}} \sum_{i=1}^{L} C(f(\mathbf{x}_i), y_i) + \lambda \|f\|^2$$

Our goal is to establish a class of functions $\mathcal{H}$ and smoothness measures $\|f\|$ for which

- We can compute f_L^* efficiently.
- The computation is robust to noise in the data.
- The functions f generalize well and are expressive enough to describe real world functional relationships.

Linear Regression

Consider the simple case of fitting the best $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$ for some vector $\mathbf{w}$. Set

$$C(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2$$

so that we are solving a least squares problem. Let the smoothness measure on f be the ordinary norm of w

$$\|f\|^2 = \|\mathbf{w}\|^2$$

Then the cost function is

$$\min_{w} \sum_{i=1}^{L} (\mathbf{w}^\top \mathbf{x}_i - y_i)^2 + \lambda \mathbf{w}^\top \mathbf{w}$$

Lemma 1 *For any matrix* $\mathbf{A}$ *and constant* λ

$$(\mathbf{A}\mathbf{A}^\top + \lambda \mathbb{1})^{-1}\mathbf{A} = \mathbf{A}(\mathbf{A}^\top\mathbf{A} + \lambda \mathbb{1})^{-1}$$

Proof

$$\begin{aligned}
(\mathbf{A}\mathbf{A}^\top + \lambda \mathbb{1})^{-1}\mathbf{A} &= \frac{1}{\lambda}\mathbf{A} - \frac{1}{\lambda^2}\mathbf{A}(\mathbb{1} + \frac{1}{\lambda}\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{A} \\
&= \frac{1}{\lambda}(\mathbf{A} - \mathbf{A}(\lambda\mathbb{1} + \mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{A} \\
&= \frac{1}{\lambda}\Big(\mathbf{A}(\lambda\mathbb{1} + \mathbf{A}^\top\mathbf{A})^{-1}(\lambda\mathbb{1} + \mathbf{A}^\top\mathbf{A}) \\
&\qquad - \mathbf{A}(\lambda\mathbb{1} + \mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{A}\Big) \\
&= \frac{1}{\lambda}\Big(\mathbf{A}(\lambda\mathbb{1} + \mathbf{A}^\top\mathbf{A})^{-1}\Big)\Big(\lambda\mathbb{1} + \mathbf{A}^\top\mathbf{A} - \mathbf{A}^\top\mathbf{A}\Big) \\
&= \mathbf{A}(\lambda\mathbb{1} + \mathbf{A}^\top\mathbf{A})^{-1}
\end{aligned}$$

■

Linear Regression: Solution

Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_L]$ be the data matrix, $\mathbf{G} = \mathbf{X}^\top \mathbf{X}$ be the Gram matrix, and $\mathbf{y}$ be the vector of y values. Then the goal is to find

$$\min_w \|\mathbf{X}^\top \mathbf{w} - \mathbf{y}\|^2 + \lambda \mathbf{w}^\top \mathbf{w}$$

Taking a derivative with respect to $\mathbf{w}$ and solving for $\mathbf{w}^*$ gives

$$\mathbf{w}^* = (\mathbf{X}\mathbf{X}^\top + \lambda \mathbb{1})^{-1} \mathbf{X}\mathbf{y} = \mathbf{X}(\mathbf{G} + \lambda \mathbb{1})^{-1} \mathbf{y}$$

Letting $\mathbf{c} = (\mathbf{G} + \lambda \mathbb{1})^{-1} \mathbf{y}$ we see that

$$f(\mathbf{x}) = \sum_{i=1}^{L} (c_i \mathbf{x}_i)^\top \mathbf{x} = \sum_{i=1}^{L} c_i (\mathbf{x}_i^\top \mathbf{x}_i)$$

Linear Regression: Comments

- All that is required is matrix inversion to solve.
- The solution is a linear combination of the data.
- Computing $f(\mathbf{x})$ only involved inner products of the data.
- Linear functions are not particularly expressive.

The Kernel Trick

Our trick for nonlinear regression is deceptively simple. We replace each data point $\mathbf{x}_i$ with a high (infinite) dimensional vector $\widehat{\mathbf{x}}_i$ and then solve the linear regression problem with the *lifted data*.

We can interpret this as creating a very long list of "features" of each data point and using the features to solve the linear regression problem.

- How do we lift the data?

- What does the resulting function look like?

Kernels

Let $\mathbf{K}$ be a function of two variables which is *positive definite*

$$\sum_{i=1}^{N} \sum_{j=1}^{N} c_i c_j \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

for all $\mathbf{x}_i$ and c_i. Such a function is called a positive definite kernel.

Lifting

By Mercer's theorem

$$K(\mathbf{x}_1, \mathbf{x}_2) = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}_1) \phi_i(\mathbf{x}_2)$$

If we lift $\mathbf{x}$ by the rule

$$\hat{\mathbf{x}} = [\sqrt{\lambda_1}\phi_1(\mathbf{x}), \sqrt{\lambda_2}\phi_2(\mathbf{x}), \ldots, \sqrt{\lambda_k}\phi_k(\mathbf{x}), \ldots]$$

we find that,

$$\langle \hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2 \rangle = \mathbf{K}(\mathbf{x}, \mathbf{y})$$

The functions $\phi(\mathbf{x})$ are the promised features. Fortunately, we do not need to compute them for most applications.

The Kernel Matrix

The matrix $\mathbf{K}$ with entries $K_{ij} = \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)$ is called the kernel matrix. It is the lifting of the Gram matrix. By the positivity of the kernel, we know it is positive semidefinite.

I will abuse notation and use $\mathbf{K}$ for the kernel matrix and $\mathbf{K}(\mathbf{x}_1, \mathbf{x}_2)$ for the kernel function.

Properties of the lifting

Let $\alpha = \sum_i c_i \hat{\mathbf{x}}_i$

- If $\|\alpha\|_K^2 := \alpha^\top \alpha$

$$\|\alpha\|_K^2 = \sum_{i,j} c_i c_j \hat{\mathbf{x}}_i^\top \hat{\mathbf{x}}_j = \mathbf{c}^\top \mathbf{K} \mathbf{c} \geq 0$$

- If $\mathbf{x}$ is another point, then

$$\alpha^\top \hat{\mathbf{x}} = \sum_i c_i \mathbf{K}(\mathbf{x}_i, \mathbf{x})$$

so we can interpret α as a function on the original space X.

Reproducing Kernel Hilbert Space

We can define a lifting as

$$\hat{\mathbf{x}} = \mathbf{K}(\mathbf{x}, \cdot)$$

And define a Hilbert space as all sums of the form $\sum_i c_i K(\mathbf{x}_i, \cdot)$ with inner product

$$\langle f, \mathbf{K}(\mathbf{x}_i, \cdot)\rangle = f(\mathbf{x}_i)$$

then we again have that

$$\langle \hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2 \rangle = \mathbf{K}(\mathbf{x}, \mathbf{y})$$

this approach is more rigorous and does not require the constraints of Mercer's theorem. But it requires a bit more machinery to understand.

The Kernel Trick

The kernel trick to tackle nonlinearities is to replace any $\mathbf{x}_1^\top \mathbf{x}_2$ with $\mathbf{K}(\mathbf{x}_1, \mathbf{x}_2)$. This is implicitly lifting the data into an RKHS, and while it looks like a hack is perfectly rigorous.

Nonlinear Regression

Replacing the Gram matrix with the kernel matrix, the solution to the linear regression problem with the lifted data will be

$$f(\mathbf{x}) = \sum_{i=1}^{L} c_i \mathbf{K}(\mathbf{x}_i, \mathbf{x})$$

where $\mathbf{c} = (\mathbf{K} + \lambda \mathbb{1})^{-1} \mathbf{y}^\top$.

Examples of Kernels

- Linear Kernel: $\mathbf{K}(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^\top \mathbf{x}_2$
- Polynomial Kernel: $\mathbf{K}(\mathbf{x}_1, \mathbf{x}_2) = (1 + \mathbf{x}_1^\top \mathbf{x}_2)^d$
- Fourier Coefficients: $\mathbf{K}(\mathbf{x}_1, \mathbf{x}_2) = \exp(i\mathbf{k}^\top(\mathbf{x}_1 - \mathbf{x}_2))$

Radial Basis Functions

A kernel which satisfies $\mathbf{K}(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{K}(\|\mathbf{x} - \mathbf{y}\|)$ is called a radial basis kernel.

- Gaussian RBF: $\mathbf{K}(\mathbf{x}_1, \mathbf{x}_2) = \exp(-C\|\mathbf{x}_1 - \mathbf{x}_2\|^2)$

- Inverse Multiquadric: $\mathbf{K}(\mathbf{x}_1, \mathbf{x}_2) = (\|\mathbf{x}_1 - \mathbf{x}_2\|^2 + C)^{-1/2}$

- FACT: The only RBF Kernels which are positive for all dimensions are of the form
$$\mathbf{K}(\mathbf{x}_1, \mathbf{x}_2) = \int \exp(-C\|\mathbf{x}_1 - \mathbf{x}_2\|^2) p(C) dC$$
where $p \geq 0$.

Representer Theorem

Theorem 2 *If you have any optimization problem of the form*

$$\begin{array}{ll} \min_f & F(f(\mathbf{x}_i)) + \lambda \|f\|_K^2 \\ \textit{s.t.} & G_j(f(\mathbf{x}_i)) \leq 0 \end{array}$$

then the solution is of the form

$$f(\mathbf{x}) = \sum_{i=1}^{L} c_i \mathbf{K}(\mathbf{x}_i, \mathbf{x})$$

for some $\mathbf{c}$.

This is the crucial theorem which makes the Kernel trick so powerful.

Proof of the Representer Theorem

Suppose the minimizer, $f(\mathbf{x})$, does not have the desired form. Then we can write $f(\mathbf{x}) = \sum_{i=1}^{L} c_i \mathbf{K}(\mathbf{x}_i, \mathbf{x}) + g(\mathbf{x})$ with

$$\langle g, \mathbf{K}(\mathbf{x}_i, \cdot)\rangle = 0$$

Then we have $f(\mathbf{x}_j) = \sum_{i=1}^{L} c_i \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)$ for all points in the data set.

But then $G_j(f(\mathbf{x}_i))) = G_j(\sum_{k=1}^{L} c_i \mathbf{K}(\mathbf{x}_k, \mathbf{x}_i)) \leq 0$. And we have

$$F(f(\mathbf{x}_i)) + \lambda \|f\|_K^2 \geq F(\sum_{k=1}^{L} c_k \mathbf{K}(\mathbf{x}_k, \mathbf{x}_j)) + \lambda \| \sum_{i=1}^{L} c_i \mathbf{K}(\mathbf{x}_i, \cdot)\|_K^2$$

which implies that $g = 0$.

Kernel Machines

Using the Representer Theorem, we see that we can use Kernels for a much wider class of function fitting problems that just regression. Generically we call the minimizing function of

$$\begin{array}{ll} \min_f & F(f(\mathbf{x}_i)) + \lambda \|f\|_K^2 \\ \text{s.t.} & G_j(f(\mathbf{x}_i)) \leq 0 \end{array}$$

a *Kernel Machine*

Classification

Given $\mathbf{x}$ determine if it belongs to class A or class B.

Examples:

- $\mathbf{x}$ is an image. Is it a face or not?
- $\mathbf{x}$ is an audio file. Is it a song by Madonna or not?
- $\mathbf{x}$ is a list of chemical properties of a wine. Is it white or red?

Regularized Least Squares Classification

One solution to the classification problem is just regression. Given a set of training examples $\mathbf{x}_1, \ldots, \mathbf{x}_N$ and class labels, let $f(\mathbf{x}_i) = 1$ for those in class A and $f(\mathbf{x}_i) = -1$ for those in class B. Then solve the nonlinear (or linear) regression problem to compute a function which gives positive values for class A and negative values for class B.

Support Vector Machines

If you are put off by the least squares cost, you could use the *Hinge Loss* cost function

$$C(f(x), y) = \text{max}(1 - f(x)y, 0)$$

which gives 0 for a correctly signed assignment and a linear penalty for misclassification.

The Kernel machine with this cost is called a *Support Vector Machine* (SVM).

Properties of SVMs

- Since the cost function is piecewise linear, we can write the solution as a quadratic program with linear constraints.

- It can be solved efficiently using a dual method called SMO.

- Because of the linearity of the constraints, some of the c_i will be zero so SVM cost induces a sparse solution.

- For most applications, RLSC and SVM perform similarly!

Semi-supervised Regression

- A learning problem is "semi-supervised" if some of the y values are withheld.
- Generally, this means that the unlabelled data can't help with the training data.
- If a prior on output values is added to the problem formulation, the unlabelled data helps the learning.

Semi-supervised Classification

Given a set of data points $\mathbf{x}_1, \ldots, \mathbf{x}_L$ and a label set $y_1, \ldots, y_K$ for the first K points we wish to find

$$\begin{array}{lll} \min_{f,\mathbf{z}} & \sum_{i=1}^{N} C(f(\mathbf{x}_i), z_i) + \lambda \|f\|^2 & \\ \text{s.t.} & z_i^2 = 1 & i = 1, \ldots, L \\ & z_k = l_k & k = 1, \ldots, K \end{array}$$

RLSK

In the least squares case this reduces to an integer program.

Let

$$\mathbf{Q} = (\mathbf{K} + \mathbb{1})^{-1} = \left[\begin{array}{cc} \mathbf{Q}_{11} & \mathbf{Q}_{12} \\ \mathbf{Q}_{12}^{\top} & \mathbf{Q}_{22} \end{array} \right]$$

where the partition 1 corresponds to the labelled points and partition 2 corresponds to the unlabelled points.

First solve for the missing labels

$$\begin{array}{ll} \min_{\mathbf{z}} & \mathbf{z}^{\top}\mathbf{Q}_{22}\mathbf{z} + 2\mathbf{l}^{\top}\mathbf{Q}_{12}\mathbf{z} \\ \text{s.t.} & z_i^2 = 1 \end{array}$$

After solving the quadratic program, use the output labels to solve for f using standard classification.

Clustering

In the absence of any labelled data, the classification problem becomes clustering. Here we solve the quadratic program

$$\begin{array}{ll} \text{min}_{\mathbf{z}} & \mathbf{z}^\top(\mathbf{K} + \lambda \mathbb{1})^{-1}\mathbf{z} \\ \text{s.t.} & z_i^2 \geq 1 \\ & \sum_{i=1}^N z_i = 0 \end{array}$$

Learning Manifolds

We can also vector valued functions by coupling the learning of each component. So you can solve, for example

$$\begin{array}{ll} \text{min} & \sum_{i=1}^{d} \left(\sum_{j=1}^{L} C(f_i(\mathbf{x}_l), y_i) + \|f_i\|_K^2 \right) \\ \text{s.t.} & \langle f_i, f_j \rangle = 0 \end{array}$$

This sort of semi-supervised regression is usually called “manifold learning,” even though it is just function fitting.

Kernel PCA

Using the Kernel matrix as a Gram matrix, we can extract principle components as eigenvectors

$$\mathbf{K}\mathbf{v} = \gamma\mathbf{v}$$

Recalling our derivation of the SVD, the principle component is then

$$\mathbf{V} = \sum_i \frac{v_i}{\gamma}\widehat{\mathbf{x}}_i$$

and for a test point $\mathbf{x}$, we can evaluate

$$\langle\mathbf{V}, \mathbf{x}\rangle = \sum_i \frac{v_i}{\gamma}\mathbf{K}(\mathbf{x}_i, \mathbf{x})$$

Other Priors

Other priors on the output that you can take advantage of

- Periodicity
- Monotonicity
- Markov Chain Dynamics

Problem 1a: Nonlinear Regression

Consider the standard regression problem

$$\min_f \sum_{i=1}^{L} C(f(\mathbf{x}_i), y_i) + \lambda \|f\|_K^2$$

plug in $f(\mathbf{x}) = \sum_{k=1}^{L} c_k K(\mathbf{x}_k, \mathbf{x})$ and derive a formula for the optimal f. Compare this to the form derived from linear regression.

Problem 1b: Nonlinear Regression

Let $g(u, v) = u \sin(2\pi(u + v)) + v \sin(3\pi(u - v))$.

- Randomly generate a set of 100 data points $\mathbf{x}_i$ in $[0, 1] \times [0, 1]$. Let $y_i = g(\mathbf{x}_i) + w_i$ where w_i is a random number with variance 0.1
- Solve the nonlinear regression problem with the linear, gaussian, and one other kernel from the list.
- Generate 10 test points, which kernel generates the best fit?

Problem 2: Multiclass Classification

Multiclass classification is classification with more than two labels. Download the ”wine” database, and train three classifiers

- class 1 vs class 2 and class 3
- class 2 vs class 1 and class 3
- class 3 vs class 1 and class 2

Try to get the best performance using leave-one out cross validation.

Problem 3: Choose your own...

- Using your data set for “eigensomething” pose a regression, classification or dimensionality reduction problem. Using your favortie kernel and algorihtm, train a kernel machine and validate it’s performance using leave-one out error.