Regression and Least-Squares Classification

9.520 Class 05, 19 February 2003

Ryan Rifkin

Plan

- Regression using the square-loss
- Solving square-loss regression
- Computational approaches
- Advantages of a linear kernel
- Classification as regression
- The Rectangular Method

Regression

In regression, we are given a training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)$. Unlike in classification, the y_i are *real-valued*. The goal is to learn a function f to predict the y values associated with new observed x values.

Our Friend Regularization

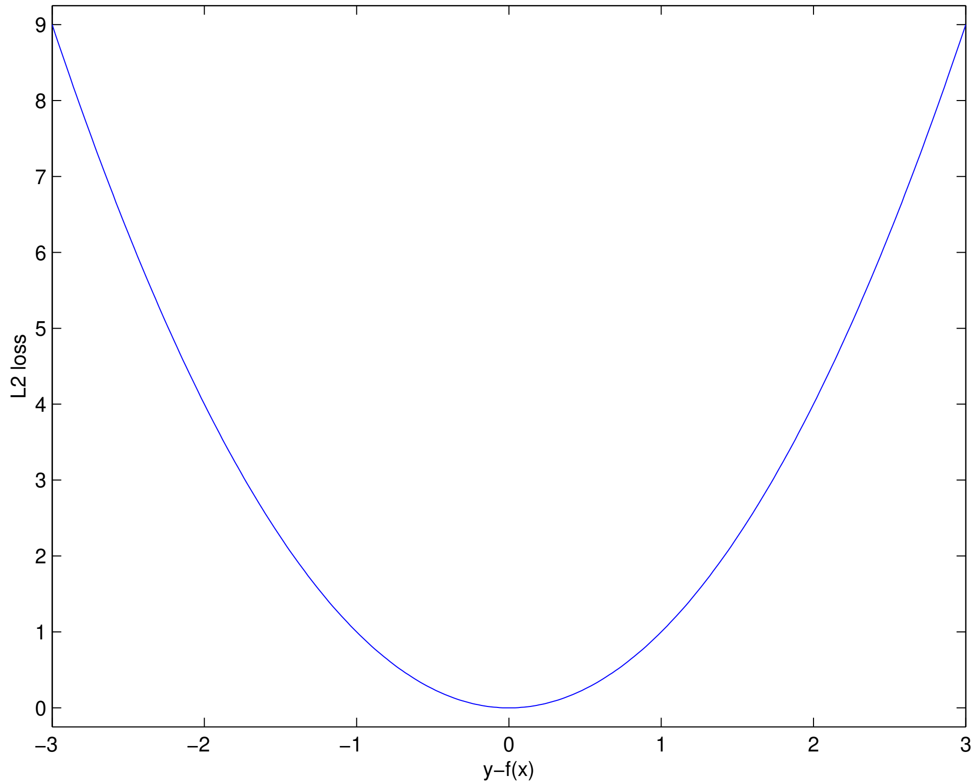
We pose our regression task as finding the function f that solves a Tikhonov regularization problem:

$$f = \arg\min_{f \in \mathcal{H}} \frac{1}{\ell} \sum_{i=1}^{\ell} V(f(x_i), y_i) + \lambda \|f\|_K^2$$

To fully specify the problem, we need to choose a loss function V and a kernel K.

The Square Loss

For regression, a natural choice of loss function is the square loss $V(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2$.



Substituting In The Square Loss

Using the square loss, our problem becomes

$$f = \arg\min_{f \in \mathcal{H}} \frac{1}{\ell} \sum_{i=1}^{\ell} (f(x_i) - y_i)^2 + \lambda \|f\|_K^2.$$

The Return of the Representer Theorem

Theorem. The solution to the Tikhonov regularization problem

$$\min_{f \in \mathcal{H}} \frac{1}{\ell} \sum_{i=1}^{\ell} V(y_i, f(x_i)) + \lambda \|f\|_K^2$$

can be written in the form

$$f(x) = \sum_{i=1}^{\ell} c_i K(x, x_i).$$

This theorem is exceedingly useful — it says that to solve the Tikhonov regularization problem, we need only find the best function of the form $f(x) = \sum_{i=1}^{\ell} c_i K(x, x_i)$. Put differently, all we have to do is find the c_i.

Applying the Representer Theorem, I

NOTATION ALERT!!! We will use the symbol K to refer to either the kernel function K **OR** the ℓ-by-ℓ matrix K defined via

$$K_{ij} \equiv K(x_i, x_j)$$

Using this definition, consider the output of our function

$$f(x) = \sum_{i=1}^{\ell} c_i K(x, x_i).$$

at the training point x_j:

$$\begin{aligned} f(x_j) &= \sum_{i=1}^{\ell} K(x_i, x_j) c_i \\ &= (Kc)_j \end{aligned}$$

Applying the Representer Theorem, I

Using this notation, we apply the Representer Theorem to our Tikhonov minimization problem, reformulating it as:

Using the square loss, our problem becomes

$$f = \min_{f \in \mathcal{H}} \frac{1}{\ell}(Kc - y)^2 + \lambda \|f\|_K^2.$$

Using the Norm of a "Represented" Function

Recall that if we have a function

$$f(x) = \sum_{i=1}^{\ell} c_i K(x_i, x),$$

then we have

$$||f||_K^2 = c^T K c.$$

Using the Norm of a "Represented" Function, II

Substituting in, our Tikhonov minimization problem is now entirely a problem of finding c:

$$f = \arg\min_{c \in \mathbb{R}^\ell} \frac{1}{\ell}(Kc - y)^2 + \lambda c^T K c.$$

Solving the Least Squares Problem, I

We are trying to minimize $g(c)$, where

$$g(c) = \frac{1}{\ell}(Kc - y)^2 + \lambda c^T K c$$

This is a **convex**, **differentiable** function of c, so we can minimize it simply by taking the derivative with respect to c, then setting this derivative to 0.

$$\frac{\partial g(c)}{\partial c} = \frac{2}{\ell} K(Kc - y) + 2\lambda K c.$$

Solving the Least Squares Problem, II

Setting the derivative to 0 and playing with some math,

$$
\begin{aligned}
& \frac{2}{\ell} K(Kc - y) + 2\lambda Kc = 0 \\
\rightarrow \quad & K(Kc) + \lambda \ell Kc = Ky \\
\text{“}\rightarrow\text{”} \quad & (K + \lambda \ell I)c = y \\
\rightarrow \quad & c = (K + \lambda \ell I)^{-1} y
\end{aligned}
$$

The matrix $K + \lambda \ell I$ is positive definite and will be well-conditioned if λ is not too small.

Least-Squares Regularized Regression

- The matrix $(K + \lambda \ell I)$ is guaranteed to be invertible if $\lambda > 0$. As $\lambda \to 0$, the regularized least-squares solution goes to the standard Gaussian least-squares solution which minimizes the empirical loss. As $\lambda \to \infty$, the solution goes to $f(\mathbf{x}) = 0$.
- In pratice, we don't actually invert $(K + \lambda \ell I)$, but instead use an algorithm for solving linear systems.
- In order to use this approach, we need to compute and store the entire kernel matrix K. This makes it impractical for use with very large training sets.

The Conjugate Gradient Algorithm

The conjugate gradient algorithm is a popular algorithm for solving positive definite linear systems. For the purposes of this class, we need to know that CG is an **iterative** algorithm. The major operation in CG is multiplying a vector v by the matrix A. Note that matrix A need not always be supplied explicitly, we just need some way to form a product Av.

For "ordinary" positive semidefinite systems, CG will be competitive with direct methods. CG can be much faster if there is a way to multiply by A quickly ...

Aside: A Linear Kernel

Suppose our kernel K is linear:

$$K(x, y) = x \cdot y.$$

Then our solution x can be written as

$$\begin{aligned} f(x) &= \sum c_i x_i \cdot x \\ &= (\sum c_i x_i) \cdot x \\ &\equiv w \cdot x, \end{aligned}$$

and we can apply our function to new examples in time d rather than time ℓd.

This is a general property of Tikhonov regularization with a linear kernel, not related to the use of the square loss.

Linear Regularized Least-Squares Regression

We can use the CG algorithm to get a huge savings for solving regularized least-squares regression with a linear kernel ($K(\mathbf{x_1}, \mathbf{x_2}) = \mathbf{x_1} \cdot \mathbf{x_2}$). With an arbitrary kernel, we must form a product Kv explicitly — we multiply a vector by K. With the linear kernel, we note that $K = AA^T$, where A is a matrix with the data points as row vectors. Using this:

$$\begin{aligned}(K + \lambda \ell I)v &= (AA^T + \lambda \ell I)v \\ &= A(A^T v) + \lambda \ell I v\end{aligned}$$

Cost Analysis of the Linear Approach

Suppose we have ℓ points in d dimensions. Forming the kernel matrix K explicitly takes $\ell^2 d$ time, and multiplying a vector by K takes ℓ^2 time.

If we use the linear representation, we pay nothing to form the kernel matrix, and multiplying a vector by K takes $2d\ell$ time.

If $d << \ell$, we save approximately a factor of $\frac{\ell}{2d}$ per iteration. The memory savings are even more important, as we cannot store the kernel matrix at all for large training sets, and if were to recompute the entries of the kernel matrix as needed, each iteration would cost $\ell^2 d$ time.

Sparse Training Data

Also note that if the training data are sparse (they consist of a large number of dimensions, but the majority of dimensions for each point are zero), the cost of multiplying a vector by K can be written as $2\bar{d}\ell$, where $\bar{d}$ is the **average** number of nonzero entries per data point.

This is often the case for applications relating to text, where the dimensions will correspond to the words in a "dictionary". There may be tens of thousands of words, but only a few hundred will appear in any given document.

Square-Loss Classification

There is nothing to stop us for using the above algorithm for **classification**. By doing so, we are essentially treating our classification problem as a regression problem with y values of 1 or -1.

Faster Nonlinear RLS

For a general nonlinear kernel, to solve the Tikhonov problem as defined, we must begin by computing the entire K matrix. We will see (soon) that this means that the SVM, which we will discuss in the next class, is preferable to RLS for large nonlinear problems.

Is there a way around this?

First Idea: Throw Away Data

Suppose that we throw away all but M of our data points, where $M << \ell$. Then we only need time $M^2 d$ to form our new, smaller kernel matrix, and we only need time M^2 for each iteration of CG. Great, isn't it?

Well, if we have too much data to begin with (say 1,000,000 points in 3 dimensions) this will work just fine. In general, we will lose accuracy.

Introducing The Rectangular Method

Suppose, instead of throwing away data, we restrict our hypothesis space further. Instead of allowing functions of the form

$$f(x) = \sum_{i=1}^{\ell} c_i K(x_i, x),$$

we only allow

$$f(x) = \sum_{i=1}^{M} c_i K(x_i, x),$$

where $M << \ell$. In other words, we only allow the first M points to have non-zero c_i. However, we still measure the loss at all ℓ points.

More On The Rectangular Method

If we define K_{MM} to be the kernel matrix on just the M points we're using to represent our function, and K_{ML} to be the kernel product between those M points and the remaining L points, we can derive (homework) that the minimization problem becomes:

$$(K_{ML}K_{LM} + K_{MM}\lambda\ell) = K_{ML}y,$$

which is again an M-by-M system.

Various authors have reported good results with this or with similar approaches, although the jury is still out (possible class project).