COS 511: Foundations of Machine Learning

Rob Schapire
Scribe: Douglas Thunen

Lecture #15
April 1, 2003

1 Linear Regression

General Framework:

Given a set of examples, $(\vec{x}_1, y_1), \ldots, (\vec{x}_m, y_m)$, where the $\vec{x}_i$s are vectors and the y_is are values, the goal is to find $\vec{w}$ such that $y_i \approx \vec{w} \cdot \vec{x}_i$ and $\Phi(\vec{w}) = \min \sum_i (y_i - \vec{w} \cdot \vec{x}_i)^2$.

In order to do this, we can build a matrix as shown in Equation 1 below:

$$\Phi(\vec{w}) = \left\| \begin{pmatrix} \vec{x}_1^T \\ \vec{x}_2^T \\ \vdots \\ \vec{x}_m^T \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \right\|^2 \tag{1}$$

and then minimize based on setting the gradient equal to zero, where each component of the gradient is calculated as

$$\tfrac{\partial \Phi}{\partial w_j} = 2 \sum (\vec{w} \cdot x_i - y_i) x_{i,j}$$

We can therefore write the gradient as a vector of all of the derivatives in terms of the matrix from Equation 1:

$$\nabla_{\vec{w}} \Phi = \begin{pmatrix} \frac{\partial \Phi}{\partial w_1} \\ \vdots \\ \frac{\partial \Phi}{\partial w_i} \end{pmatrix} = 2 \sum (\vec{w} \cdot \vec{x}_i - y_i) \vec{x}_i = 2M^T (M\vec{w} - b) \tag{2}$$

$$\text{where } M = \begin{pmatrix} \vec{x}_1^T \\ \vec{x}_2^T \\ \vdots \\ \vec{x}_m^T \end{pmatrix}, \vec{w} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{pmatrix}, \text{ and } b = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

Thus, setting the gradient equal to zero, we can calculate $\vec{w}$:

$$2M^T (M\vec{w} - b) = 0$$
$$M^T M \vec{w} = M^T b$$
$$\vec{w} = (M^T M)^{-1} M^T b$$

where the quantity $(M^T M)^{-1} M^T$ is known as the "pseudoinverse." (It can be computed whenever $M^T M$ is invertible, but even if $M^T M$ is not invertible, it is still possible to find $\vec{w}$ using some other method, *e.g.* gradient descent.)

While this is useful in that it describes the minimum loss function on the training set, we would also like to be able to prove bounds on new data. In order to address this problem, we next examine an online learning model.

2 Online Model

In the case of an online model, instead of working with a certain batch of examples, the algorithm receives one example at a time. The algorithm then needs to make a prediction about the next y value based on the observed example x.

The general idea in this case, then, is the following:

For t=1,2,. . . ,T:

1. Get $\vec{x}_t \in \mathbf{R}^n$
2. Predict $\hat{y}_t = \vec{w}_t \cdot \vec{x}_t$
3. Observe $y_t \in \mathbf{R}$

Where the goal is to minimize the loss function, L, relative to the best loss we could get from any vector $\vec{u}$, more specifically:

$$\text{minimize } L = \sum_t (\vec{w}_t \cdot \vec{x}_t - y_t)^2 \text{ relative to } L_{\vec{u}} = \sum_t (\vec{u} \cdot \vec{x}_t - y_t)^2 \text{ for ``best'' } \vec{u}$$

2.1 Real World Example - Echo Cancellation

An example of where this method is used (or at least used to be), is in echo cancellation in long distance telephone calls.

The problem is that a small amount of what one person says leaks through to the return connection, and is thus transmitted back to the original speaker and heard as an echo. The goal of echo cancellation, then, is to try to predict that echo so that the system can introduce the negative of the echo on the return wire in order to cancel it out.

A solution, in simplified terms, is to use a learning algorithm (Widrow-Hoff) that takes a small, say 100ms, window into the past, $\vec{x}_t$, and from that try to predict what would be heard at some point on the return, y_t. The idea is to determine a vector $\vec{w}$, where $\vec{w} \cdot \vec{x}_t$ is a good approximation of y_t.

2.2 Widrow-Hoff Algorithm (LMS or Least Mean Squares)

How is the weight vector $\vec{w}$ maintained and updated?

$$\vec{w}_1 = 0$$
$$\vec{w}_{t+1} = \vec{w}_t - \eta(\vec{w}_t \cdot \vec{x}_t - y_t)\vec{x}_t$$

where η is the learning rate and $0 < \eta < 1$

Vector $\vec{w}$ is thus updated by simply subtracting a constant (η) times the gradient from the previous value of $\vec{w}$. There are two motivations for this update rule:

1. (gradient descent) $\vec{w}_t$ is known, $\vec{x}_t$ and y_t have just been seen, and $\vec{w}$ must be adjusted. The loss is

$$L(\vec{w}_t, \vec{x}_t, y_t) = (\vec{w}_t \cdot \vec{x}_t - y_t)^2$$

As such, the update to $\vec{w}$ should move $\vec{w}$ so that $L \downarrow$. One method of doing this is to compute the gradient and move in its negative direction:

$$\nabla L(\vec{w}_t) = 2(\vec{w}_t \cdot \vec{x}_t - y_t)\vec{x}_t$$

and then take a small step in the direction that causes L to decrease.

2. (More general) There are two goals:

   (a) The loss, $L(\vec{w}_{t+1}, \vec{x}_t, \vec{y}_t)$ should be small, and thus $(\vec{w}_t \cdot \vec{x}_t - y_t)^2$ should also be small.

   (b) $\vec{w}_{t+1}$ should be close to $\vec{w}_t$, but then some measure of closeness is needed: $\|\vec{w}_{t+1} - \vec{w}_t\|^2$ should also be small.

   To meet these goals, we can take a linear combination of the above and minimize their sum:
   Find $\vec{w}_{t+1}$ to minimize $(\eta(\vec{w}_{t+1} \cdot \vec{x}_t - y_t)^2 + \|\vec{w}_{t+1} - \vec{w}_t\|^2)$
   Working through the minimization, we get

$$\vec{w}_{t+1} = \vec{w}_t - \eta(\vec{w}_{t+1} \cdot \vec{x}_t - y_t)\vec{x}_t$$

   It is then possible to solve for $\vec{w}_{t+1}$ or simply use the approximation $\vec{w}_t$ for $\vec{w}_{t+1}$ on the right side of the previous equation.

Given this update function, we can prove the following theorem:

Theorem 1: Assume $\|\vec{x}_t\|_2 \leq 1$, then

$$L \leq \min_{\vec{u}} \left[\tfrac{L_{\vec{u}}}{1-\eta} + \tfrac{\|\vec{u}\|_2^2}{\eta}\right]$$

Alternatively, if we divide through by T, the total number of time steps,

$$\tfrac{L}{T} \leq \tfrac{L_{\vec{u}}}{T}(1+\eta) + \tfrac{\|\vec{u}\|^2}{\eta T}$$

That is, the loss of the algorithm over the number of time steps converges to the loss of a best vector $\vec{u}$.

Note that the result is completely adversarial, *i.e.* it does not make any assumptions about the input.

Proof:
We must first choose a potential function, Φ:

$$\Phi_t = \|\vec{w}_t - \vec{u}\|_2^2, \textit{ i.e. } \text{a measure of how close we are to best vector } \vec{u}$$

Notation: $e_t = (\vec{w}_t \cdot \vec{x}_t - y_t)$, $g_t = (\vec{u} \cdot \vec{x}_t - y_t)$

Claim: $\Phi_{t+1} - \Phi_t \leq -\eta e_t^2 + \frac{\eta}{1-\eta} g_t^2$

Pf: $-\|\vec{u}\|^2 \leq \|\vec{w}_{t+1} - \vec{u}\|^2 - \|\vec{w}_1 - \vec{u}\|^2 = \Phi_{T+1} - \Phi_1$

$$= (\Phi_{T+1} - \Phi_T) + (\Phi_T - \Phi_{T-1}) + \ldots + (\Phi_2 - \Phi_1)$$

$$\leq \sum_t (-\eta e_t^2 + \tfrac{\eta}{1-\eta} g_t^2) = -\eta \sum_t e_t^2 + \tfrac{\eta}{1-\eta} \sum_t g_t^2$$

where $\sum_t e_t^2 = L$ and $\sum_t g_t^2 = L_{\vec{u}}$

We can then solve for L: $L \leq \frac{L_{\vec{u}}}{1-\eta} + \frac{\|\vec{u}\|^2}{\eta}$

Now it is necessary to compute the change in Φ: $\Phi_{t+1} - \Phi_t$

$$\|\vec{w}_{t+1} - \vec{u}\|^2 - \|\vec{w}_t - \vec{u}\|^2 = \|\vec{w}_t - \vec{u} - \Delta_t\|^2 - \|\vec{w}_t - \vec{u}\|^2$$

$$= \|\vec{w}_t - \vec{u}\|^2 - 2\Delta_t \cdot (\vec{w}_t - \vec{u}) + \|\Delta_t\|^2 - \|\vec{w}_t - \vec{u}\|^2$$

$$= \eta^2 e_t^2 \|\vec{x}_t\|^2 - 2\eta e_t \vec{x}_t \cdot (\vec{w}_t - \vec{u})$$

(since $\eta^2 e_t^2 \|\vec{x}_t\|^2 = \|\Delta_t\|^2$ and $\eta e_t \vec{x}_t = \Delta_t$)

$$\leq \eta^2 e_t^2 - 2\eta e_t(\vec{w}_t \cdot \vec{x}_t - y_t + y_t - \vec{u} \cdot \vec{x}_t)$$

$$= (\eta^2 - 2\eta) e_t^2 + 2\eta e_t g_t$$

We have a product $(2\eta e_t g_t)$, but we want squares:
Proposition: $ab \leq \frac{a^2+b^2}{2} \Leftrightarrow 0 \leq a^2 - 2ab + b^2 = (a-b)^2$

So we can choose $a = \frac{g_t}{c}, b = e_t c \Rightarrow a = \frac{g_t}{\sqrt{1-\eta}}, b = e_t\sqrt{1-\eta}$

Plugging in, then: $(\eta^2 - 2\eta)e_t^2 + 2\eta \left[\frac{g_t^2}{1-\eta} + e_t^2(1-\eta)\right] \frac{1}{2}$

And thus if true for any $\vec{u}$, obviously true for $\min \vec{u}$.

2.3 Comments

Where does this potential come from? How does this technique generalize?

To derive Widrow-Hoff, we found $\vec{w}_{t+1}$ to minimize $\eta(\vec{w}_{t+1} \cdot \vec{x}_t - y_t)^2 + \|\vec{w}_{t+1} - \vec{w}_t\|^2$. In general, we want to minimize:

$$\eta(\text{loss of } \vec{w}_{t+1} \text{ on } \vec{x}_t, y_t) + (\text{distance from } \vec{w}_t \text{ to } \vec{w}_{t+1})$$

So suppose loss is some function $L(\vec{w}_{t+1}, \vec{x}_t, y_t)$. We can compute the gradient and get:

$$\vec{w}_{t+1} = \vec{w}_t - \eta \nabla L(\vec{w}_{t+1}, \vec{x}_t, y_t)$$

which is problematic since $\vec{w}_{t+1}$ appears on both sides of the equation. We can, however, approximate $\vec{w}_{t+1}$ with $\vec{w}_t$ on the right hand side.

It is also possible to generalize these concepts by working with alternate distance functions, *e.g.* Relative Entropy:

$$\eta(\vec{w}_{t+1} \cdot \vec{x}_t - y_t)^2 + RE(\vec{w}_t \| \vec{w}_{t+1})$$

in which case the update function would be:

$$\vec{w}_{t+1,i} = \frac{\vec{w}_{t,i} e^{-\eta(\vec{w}_{t+1} \cdot \vec{x}_t - y_t) x_{t,i}}}{Z_t}$$

where Z_t is a normalization factor.

For this algorithm, using potential $\Phi = RE(\vec{u} \| \vec{w}_t)$, it is possible to prove the following bound on the total loss of the algorithm:

$$\begin{gathered} \textstyle\sum_t (\vec{w}_t \cdot \vec{x}_t - y_t) \leq \min_{\vec{u}} \left[a_\eta \textstyle\sum_t (\vec{u} \cdot \vec{x}_t - y_t)^2 + b_\eta \ln n \right] \\ \text{where } \|\vec{x}_t\|_\infty \leq 1 \text{ and } \|\vec{u}\|_1 = 1 \end{gathered}$$