Introduction to Machine Learning **Fall Semester, 2013**

Recitation 9: December 15

Lecturer: Mariano Schain *Scribe: ym*

9.1 Adaboost - Continue

We will finish something that we did not have a chance to complete last week.

We would like to consider a very simple class of weak learners, where their predictions are independent. Our goal has two folds: We like to see that the complexity bound we get (in terms of the required number of iterations to achieve a certain accuracy) is very similar to AdaBoost, showing that the number of iteration in AdaBoost cannot be reduced. Also, in this simple setting the intuitive majority rule would do the work, in contrast to the general case where we needed to continuously modify the distribution.

Model Each time t we get a new weak learner h_t, such that for any valid (x, y) we have $\Pr[h_t(x) = y] = 0.5+\gamma$. Note that there is no need to have a distribution D, since we require it to hold for each x independently.

We will define, given the h_t, a majority classifier,

$$H(x) = sign(\sum_{t=1}^{T} h_t(x))$$

Our goal is to derive a bound on the error of H as a function of γ and T.

An error of H, i.e., $H(x_i) \neq y_i$, whenever the number of correct classification by the T hypotheses h_t is less than $T/2$. Our stochastic assumption implies that we can compute the error for each x, independently. The error bound would be derived using a Chernoff bound. Recall, that given n random variables $X_1, , \ldots, X_n$ which are i.i.d., we have that

$$\Pr[\sum_{i=1}^{n} X_i > \mu + \lambda] < e^{-\lambda^2/n} \quad \text{and} \quad \Pr[\sum_{i=1}^{n} X_i < \mu - \lambda] < e^{-\lambda^2/n}$$

where $\mu = E[\sum_{i=1}^{n} X_i] = nE[X_i]$.

In our case, the expected number of correct predictions is $\mu = T(0.5 + \gamma) = T/2 + \gamma T$. Let X_t be a random variable indicating whether $h_t(x_i)$ is correct, i.e., $X_t = I_{h_t(x_i)=y_i}$. This implies that,

$$\Pr[H(x_i) \neq y_i] = \Pr[\sum_{t=1}^{T} X_t \leq T/2] = \Pr[\sum_{t=1}^{T} X_t \leq \mu - \gamma T] < e^{-\gamma^2 T}$$

9.2 Linear Regression

Our basic model is a linear function, i.e., $h_\theta(x) = \theta^t x$, where θ are the parameters we like to learn for the linear function. The examples are (x_i, y_i). We can model this by a matrix

$$X = \begin{pmatrix} \cdots & x_1 & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & x_m & \cdots \end{pmatrix} = \begin{pmatrix} \vdots & & \vdots \\ X^1 & \cdots & X^d \\ \vdots & & \vdots \end{pmatrix}$$

where X^j are the values of attribute j. The labels are

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}$$

We would like to find the parameter θ that minimizes the loss, namely

$$\begin{aligned} \min_\theta \sum_{i=1}^m loss(h_\theta, (x_i, y_i)) &= \min_\theta \sum_{i=1}^m (h_\theta(x_i) - y_i))^2 && \text{assume square loss} \\ &= \min_\theta \sum_{i=1}^m (\theta^t x_i - y_i))^2 && \text{linear predictions} \\ &= \min_\theta \|X\theta^t - Y\|_2^2 \end{aligned}$$

Alternatively, we have that

$$X\theta^t = \theta_1 X^1 + \cdots + \theta_d X^d$$

For example, if $x_1 = (1, 2)$, $x_2 = (3, 4)$, $x_3 = (5, 6)$, let $\theta = (a, b)$.

$$X\theta^t = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a + 2b \\ 3a + 4b \\ 5a + 6b \end{pmatrix} = a \begin{pmatrix} 1 \\ 3 \\ 5 \end{pmatrix} + b \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix}$$

Therefore, we are looking for the linear combination (defined by θ) of the columns of X that is closest to y.

9.2.1 feature normalization

Unlike the nearest neighbor (NN) we will show that feature normalization (more precisely, scaling) does not influence the regression.

Let θ^* be the value of θ which minimized the square error. The minimizer can be viewed as the projection of y to the space span by $\{X^j\}_{j=1,\ldots,d}$. Recall that we have $X^j(Y - X\theta^*) = 0$.

Assume we scale each coordinate j by some constant a_j (therby, 'normalizing' the feature j). This implies that the new feature vector for j is $\tilde{X}^j = a_j X^j$. We are looking for the solution of the new system, denoted by $\tilde{\theta}^*$.

$$\tilde{\theta}^* = \arg\min_{\tilde{\theta}} \|\tilde{X}\tilde{\theta}^t - Y\|_2^2$$

Note that

$$\tilde{X}\tilde{\theta}^t = a_1\tilde{\theta}_1 X^1 + \cdots + a_d\tilde{\theta}_d X^d$$

This again is a linear combination of the columns of X and therefore we will find the same combination closest to y as before, implying that $\tilde{\theta}_i^* = a_i\theta_i^*$. Since we have no restriction on θ essentially we recover the same solution and have the same predictions (although the specific values of θ^* will be scaled accordingly - by the same scaling applied to the features).

Challenge Question: What will happen in Ridge and Lasso regression?

9.2.2 Logistic regression

Assume we need to predict a Boolean outcome $y \in \{0, 1\}$. While this is a classical classification setting, we can still use regression. The problem is that the linear function of the regression can map values to be larger than 1 or below 0. we like to map any real number z to a value in $[0, 1]$. One such function is $g(z) = \frac{1}{1+e^{-z}}$.

We will now give a maximum likelihood interpretation and use it to learn θ.

$$h_\theta(x) = \Pr[y = 1|x; \theta] = g(\theta^t x)$$

We will learn θ using ML model. The probabilities in the model will be

$$\Pr[y|x; \theta] = (h_\theta(x))^y(1 - h_\theta(x))^{1-y}$$

The likelihood function would be

$$L(\theta|X, Y) = \Pr[Y|X; \theta] = \prod_{i=1}^{m}(h_\theta(x_i))^{y_i}(1 - h_\theta(x_i))^{1-y_i}$$

The log likelihood is

$$\ell(\theta|X, Y) = \sum_{i=1}^{m} y_i \log(h_\theta(x_i)) + (1 - y_i)\log(1 - h_\theta(x_i))$$

We now present an online model for this. Consider sample (x_i, y_i). Let

$$F(\theta) = y_i \log(h_\theta(x_i)) + (1 - y_i) \log(1 - h_\theta(x_i))$$

Maximizing over θ we consider the gradient in the direction of the j^{th} feature:

$$\frac{dF(\theta)}{d\theta_j} = \frac{y_i}{h_\theta(x_i)} g'(\theta^t x_i) x_j - \frac{1 - y_i}{1 - h_\theta(x_i)} g'(\theta^t x_i) x_j$$

Computing the derivative of g we have

$$g'(z) = \frac{e^{-z}}{(1 + e^{-z})^2} = \frac{1}{1 + e^{-z}} \cdot \frac{e^{-z}}{1 + e^{-z}} = g(z)(1 - g(z))$$

Therefore we have

$$\begin{aligned}
\frac{dF}{d\theta_j} =& \frac{y_i}{g(\theta^t x_i)} g(\theta^t x_i)(1 - g(\theta^t x_i) x_{ij} - \frac{1 - y_i}{1 - g(\theta^t x_i)} g(\theta^t x_i)(1 - g(\theta^t x_i)) x_{ij} \\
=& y_i (1 - g(\theta^t x_i) x_{ij} - (1 - y_i) g(\theta^t x_i) x_{ij} \\
=& (y_i - g(\theta^t x_i)) x_{ij}
\end{aligned}$$

The update of θ at the k^{th}i iteration is (moving at the direction of the gradient, since we maximize the log-likelyhood),

$$\theta^{k+1} = \theta^k + \alpha(y_k - h_{\theta^k}(x_k)) x_k$$

Note that this has the same form as the update for linear regression (however the updates are essentially different sine the underlying h_θ are different).