COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Frank Xiao

Lecture # 18
April 11, 2013

1 The Widrow-Hoff Algorithm

Last lecture we talked about the Widrow-Hoff algorithm, which we include below for completeness:

Algorithm 1: Widrow-Hoff

initialize $\boldsymbol{w}_1 = \mathbf{0}$;
for $t = 1$ **to** T **do**
  get $\boldsymbol{x}_t \in \mathbb{R}^n$;
  predict $\hat{y}_t = \boldsymbol{w}_t \cdot \boldsymbol{x}_t$;
  observe y_t;
  incur loss of $(\hat{y}_t - y_t)^2$;
  update $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta(\boldsymbol{w}_t \cdot \boldsymbol{x}_t - y_t)\boldsymbol{x}_t$;
end

We define the loss of algorithm $\mathcal{A}$ to be $L_A = \sum_{t=1}^{T}(\hat{y}_t - y_t)^2$, and the loss of any vector $\boldsymbol{u} \in \mathbb{R}^n$ to be $L_u = \sum_{t=1}^{T}(\boldsymbol{u} \cdot \boldsymbol{x}_t - y_t)^2$. We left off last time wanting to upper bound the loss of Widrow-Hoff in terms of the loss of the best vector in hindsight, which we do in the following theorem.

Theorem 1.1 *Assume that for all rounds t we have $\|\boldsymbol{x}_t\|_2^2 \leq 1$, then we have*

$$L_{WH} \leq \min_{\boldsymbol{u} \in \mathbb{R}^n} \left[\frac{L_{\boldsymbol{u}}}{1-\eta} + \frac{\|u\|_2^2}{\eta} \right], \tag{1}$$

where L_{WH} denotes the loss of the Widrow-Hoff algorithm.

Proof Let $\boldsymbol{u} \in \mathbb{R}^n$ be an arbitrary vector. We define a potential $\Phi_t = \|\boldsymbol{w}_t - \boldsymbol{u}\|_2^2$, and also define the following three quantities:

$$\begin{aligned} l_t &= \hat{y}_t - y_t = \boldsymbol{w}_t \cdot \boldsymbol{x}_t - y_t \\ g_t &= \boldsymbol{u} \cdot \boldsymbol{x}_t - y_t \\ \boldsymbol{\Delta}_t &= \eta(\boldsymbol{w}_t \cdot \boldsymbol{x}_t - y_t)\boldsymbol{x}_t = \eta l_t \boldsymbol{x}_t, \end{aligned}$$

so that l_t^2 denotes the learner's loss at round t, g_t^2 is $\boldsymbol{u}$'s loss at round t, and $\boldsymbol{\Delta}_t$ is the update to the weight vector.

Now we need the following result:

Claim 1.2

$$\Phi_{t+1} - \Phi_t \leq -\eta l_t^2 + \frac{\eta}{1-\eta} g_t^2. \tag{2}$$

We'll prove the claim later, but for now suppose that it's true. Then we can prove the bound in (1) by first making the following observation:

$$-\|\boldsymbol{u}\|_2^2 = -\Phi_1 \leq \Phi_{T+1} - \Phi_1, \tag{3}$$

where the equality comes from the fact that we initialize $\boldsymbol{w}_1 = \mathbf{0}$, and the inequality is due to the potential Φ_t being non-negative.

Now we rewrite the rightmost term of (3) as a telescoping sum, and get:

$$\begin{aligned}
-\|\boldsymbol{u}\|_2^2 &\leq \sum_{t=1}^{T} (\Phi_{t+1} - \Phi_t) \\
&\leq \sum_{t=1}^{T} (-\eta l_t^2 + \frac{\eta}{1-\eta} g_t^2) && \text{using (2)} \\
&= -\eta L_{WH} + \frac{\eta}{1-\eta} L_u \Longrightarrow \\
L_{WH} &\leq \frac{1}{1-\eta} L_u + \frac{\|\boldsymbol{u}\|_2^2}{\eta}.
\end{aligned}$$

Since $\boldsymbol{u}$ was arbitrary, the above inequality in particular must hold for the best hindsight vector, so that we get the inequality in (1).

Now what's left is to prove the claim in (2).

Proof

$$\begin{aligned}
\Phi_{t+1} - \Phi_t &= \|\boldsymbol{w}_{t+1} - \boldsymbol{u}\|_2^2 - \|\boldsymbol{w}_t - \boldsymbol{u}\|_2^2 && \text{definition of potential} \\
&= \|\boldsymbol{w}_t - \boldsymbol{u} - \boldsymbol{\Delta}_t\|_2^2 - \|\boldsymbol{w}_t - \boldsymbol{u}\|_2^2 \\
&= \|\boldsymbol{w}_t - \boldsymbol{u}\|_2^2 - 2(\boldsymbol{w}_t - \boldsymbol{u}) \cdot \boldsymbol{\Delta}_t + \|\boldsymbol{\Delta}_t\|_2^2 - \|\boldsymbol{w}_t - \boldsymbol{u}\|_2^2 && \text{since } \|\boldsymbol{a} - \boldsymbol{b}\|_2^2 = \|\boldsymbol{a}\|_2^2 - 2\boldsymbol{a} \cdot \boldsymbol{b} + \|\boldsymbol{b}\|_2^2 \\
&= -2\eta l_t \boldsymbol{x}_t \cdot (\boldsymbol{w}_t - \boldsymbol{u}) + \eta^2 l_t^2 \|\boldsymbol{x}_t\|_2^2 \\
&\leq -2\eta l_t (\boldsymbol{w}_t \cdot \boldsymbol{x}_t - \boldsymbol{u} \cdot \boldsymbol{x}_t) + \eta^2 l_t^2 && \text{since } \|\boldsymbol{x}_t\|_2^2 \leq 1 \\
&= -2\eta l_t [(\boldsymbol{w}_t \cdot \boldsymbol{x}_t - y_t) - (\boldsymbol{u} \cdot \boldsymbol{x}_t - y_t)] + \eta^2 l_t^2 && \text{subtracting and adding a } y_t \\
&= -2\eta l_t (l_t - g_t) + \eta^2 l_t^2 \\
&= -2\eta l_t^2 + 2\eta l_t g_t + \eta^2 l_t^2 \\
&\leq -2\eta l_t^2 + 2\eta \left(\frac{l_t^2(1-\eta) + \frac{g_t^2}{1-\eta}}{2} \right) + \eta^2 l_t^2 && \text{by AM-GM} \\
&= -\eta l_t^2 + \frac{\eta}{1-\eta} g_t^2,
\end{aligned}$$

where we used the arithmetic mean-geometric mean inequality (AM-GM), which states that for any set of non-negative real numbers, the arithmetic mean of the set is greater than or equal to the geometric mean of the set. For two non-negative reals a and b, the inequality is $\sqrt{ab} \leq (a+b)/2$. In our case we set $a = l_t^2(1-\eta)$ and $g_t^2/(1-\eta)$. This completes the proof of Theorem 1.1.

We can look at the average loss per time step by dividing both sides of (1) by the total number of rounds T, to get

$$\frac{L_{WH}}{T} \leq \min_{\boldsymbol{u}} \left[\frac{1}{1-\eta} \cdot \frac{L_{\boldsymbol{u}}}{T} + \frac{\|\boldsymbol{u}\|_2^2}{\eta T} \right]$$

As T gets large, the term $\|\boldsymbol{u}\|_2^2/(\eta T)$ goes to 0; and if the step-size η is very small, the first term on the right hand side gets close to $\min_{\boldsymbol{u}} L_{\boldsymbol{u}}/T$, which is the average loss of the best hindsight vector. This means that the Widrow-Hoff algorithm is performing almost as well as the best hindsight vector as the number of rounds gets large.

2 Families of Online Algorithms

In the previous lecture, we motivated the update to the weight vector $\boldsymbol{w}_t$ by describing two objectives of the learning algorithm: minimize the loss of $\boldsymbol{w}_{t+1}$ on the point $(\boldsymbol{x}_t, y_t)$, and minimize the "distance" between $\boldsymbol{w}_{t+1}$ and $\boldsymbol{w}_t$. To simultaneously reflect these two goals, we formulated the problem as an optimization problem:

$$\min \eta L(\boldsymbol{w}_{t+1}, \boldsymbol{x}_t, y_t) + d(\boldsymbol{w}_{t+1}, \boldsymbol{w}_t) \tag{4}$$

where $L(\boldsymbol{w}, \boldsymbol{x}, y)$ is the loss of using weight vector $\boldsymbol{w}$ on the point $(\boldsymbol{x}, y)$, $d(\boldsymbol{p}, \boldsymbol{q})$ is some distance measurement between $\boldsymbol{p}$ and $\boldsymbol{q}$, and η is a real number that captures the relative importance of these two different goals.

2.1 Gradient Descent Algorithms

We have some freedom in choosing the loss and distance functions, depending on the particular problem we're working with. If we use the Euclidean norm as our distance measurement (so $d(\boldsymbol{p}, \boldsymbol{q}) = \|\boldsymbol{p} - \boldsymbol{q}\|_2^2$), then when we solve the optimization problem in (4), we get the following update equation

$$\begin{aligned} \boldsymbol{w}_{t+1} &= \boldsymbol{w}_t - \eta \nabla_{\boldsymbol{w}} L(\boldsymbol{w}_{t+1}, \boldsymbol{x}_t, y_t) \\ &\approx \boldsymbol{w}_t - \eta \nabla_{\boldsymbol{w}} L(\boldsymbol{w}_t, \boldsymbol{x}_t, y_t), \end{aligned} \tag{5}$$

where in the second line we use $\boldsymbol{w}_t$ as an approximation for $\boldsymbol{w}_{t+1}$, since when we run the algorithm we can't use $\boldsymbol{w}_{t+1}$ in calculating $\boldsymbol{w}_{t+1}$. This update equation describes the gradient descent family of algorithms. For Widrow-Hoff, in addition to specifying the distance measurement as being the Euclidean norm, we also set the loss function $L(\boldsymbol{w}, \boldsymbol{x}, y) = (\boldsymbol{w} \cdot \boldsymbol{x} - y)^2$, and so the update equation became $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta(\boldsymbol{w}_t \cdot \boldsymbol{x}_t - y_t)\boldsymbol{x}_t$.

2.2 Exponentiated Gradient (EG) Algorithms

Now suppose the loss function is the square loss $L(\boldsymbol{w}, \boldsymbol{x}, y) = (\boldsymbol{w} \cdot \boldsymbol{x} - y)^2$, and the distance is $RE(\boldsymbol{p}||\boldsymbol{q})$, where now we assume that $\boldsymbol{p}$ and $\boldsymbol{q}$ are probability distributions on n points, or in other words they lie in the standard $(n-1)$-simplex, denoted by Δ^{n-1}. Letting $\boldsymbol{a}(i)$ denote the ith coordinate of $\boldsymbol{a}$, the resulting update step when we solve (4) is given by

$$Z_t = \sum_{j=1}^{n} \boldsymbol{w}_t(j) \exp(-\eta(\boldsymbol{w}_t \cdot \boldsymbol{x}_t - y_t)\boldsymbol{x}_t(j))$$

$$\begin{aligned} \boldsymbol{w}_{t+1}(i) &= \frac{\boldsymbol{w}_t(i)\exp(-\eta(\boldsymbol{w}_{t+1}\cdot\boldsymbol{x}_t - y_t)\boldsymbol{x}_t(i))}{Z_t} \\ &\approx \frac{\boldsymbol{w}_t(i)\exp(-\eta(\boldsymbol{w}_t\cdot\boldsymbol{x}_t - y_t)\boldsymbol{x}_t(i))}{Z_t}, \end{aligned} \tag{6}$$

where Z_t is a normalization factor to ensure that the updated vector $\boldsymbol{w}_{t+1}$ lies in Δ^{n-1}. This update equation describes the exponentiated gradient family of algorithms.

It's possible to derive the following bound for EG, which we state without proof.

Theorem 2.1 *Suppose $\|\boldsymbol{x}_t\|_\infty \leq 1$ for all rounds t, then we have*

$$L_{EG} \leq \min_{\|\boldsymbol{u}\|_1=1} [a_\eta L_{\boldsymbol{u}} + b_\eta \ln N]$$

where L_{EG} denotes the loss of EG, and a_η and b_η are some constants.

2.3 Recap of Previous Algorithms

The update in (5) is an additive one, while in (6) it's multiplicative. Besides gradient descent, other algorithms we've covered that use an additive update include SVMs and the perceptron algorithm. When we studied these algorithms, we assumed a (L_2, L_2) bound on the norms of the prediction vectors $\boldsymbol{x}_t$ and best hindsight vector $\boldsymbol{u}$, respectively. Besides EG, we also learned about AdaBoost and the Winnow algorithm which used multiplicative updates. Here we assumed a (L_∞, L_1) bound.

3 Using an Online Algorithm in a Batch Setting

It turns out that online algorithms can be modified slightly for use in some batch learning settings. To show this, we look at the specific problem of linear regression (though the techniques involved are more general and can be used to deal with other problem types as well).

In the linear regression setting, we assume the training and test examples are i.i.d. from a fixed distribution D, and we're given a training set $S = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m))$, with $S \sim D^m$. We then define the risk $R_{\boldsymbol{v}}$ of a prediction vector $\boldsymbol{v}$ as

$$R_{\boldsymbol{v}} = \mathrm{E}_D[(\boldsymbol{v}\cdot\boldsymbol{x} - y)^2],$$

where the expectation E_D is with respect to the random sampling of the test example $(\boldsymbol{x}, y)$.

We'd like an algorithm that after training on S outputs a vector $\boldsymbol{v}$ with low risk. One approach is to take the Widrow-Hoff algorithm, and pass it the examples $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)$ (in that order), which would then generate the weight vectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{m+1}$. A natural choice is to then just output the last weight vector $\boldsymbol{w}_{m+1}$. This is often what's done in practice, but here we'll look at the case where the output is an average of the first m vectors, so $\boldsymbol{v} = \frac{1}{m}\sum_{t=1}^m \boldsymbol{w}_t$ (we leave out $\boldsymbol{w}_{m+1}$). In some experiments this average has had good performance, but we also choose to study this behavior since it makes the analysis easier, and we're able to prove the following bound on the risk.

Theorem 3.1 *Let $\boldsymbol{v} = \frac{1}{m}\sum_{t=1}^{m} \boldsymbol{w}_t$ be the output vector of running the modified Widrow-Hoff algorithm above. Then we have*

$$\mathrm{E}_S[R_{\boldsymbol{v}}] \leq \min_{\boldsymbol{u}\in\mathbb{R}^n} \left[\frac{R_{\boldsymbol{u}}}{1-\eta} + \frac{\|\boldsymbol{u}\|_2^2}{\eta m} \right],$$

where E_S refers to the expectation with respect to the random sample set $S \sim D^m$.

Proof The proof relies on the following three observations:

$$\begin{aligned}
(\boldsymbol{v}\cdot\boldsymbol{x} - y)^2 &= \left[\left(\frac{1}{m}\sum_{t=1}^{m} \boldsymbol{w}_t \right) \cdot \boldsymbol{x} - y \right]^2 \\
&= \left[\frac{1}{m}\sum_{t=1}^{m} (\boldsymbol{w}_t\cdot\boldsymbol{x} - y) \right]^2 \\
&\leq \frac{1}{m}\sum_{t=1}^{m} (\boldsymbol{w}_t\cdot\boldsymbol{x} - y)^2 && \text{since } x^2 \text{ is convex} && (7)\\
\mathrm{E}_{S\times D}[(\boldsymbol{u}\cdot\boldsymbol{x}_t - y_t)^2] &= \mathrm{E}_{S\times D}[(\boldsymbol{u}\cdot\boldsymbol{x} - y)^2] && \forall \boldsymbol{u}\in\mathbb{R}^n && (8)\\
\mathrm{E}_{S\times D}[(\boldsymbol{w}_t\cdot\boldsymbol{x}_t - y_t)^2] &= \mathrm{E}_{S\times D}[(\boldsymbol{w}_t\cdot\boldsymbol{x} - y)^2], && && (9)
\end{aligned}$$

where $\mathrm{E}_{S\times D}$ in (8) and (9) refers to the expectation with respect to the random sampling of S and the test example $(\boldsymbol{x}, y)$. The equality in (8) is because of the i.i.d. assumption about the training and test examples. The equality in (9) is because in generating $\boldsymbol{w}_t$, the Widrow-Hoff algorithm won't have seen $\boldsymbol{x}_t$ yet, so the i.i.d. property means the expectations are the same.

Now we can prove the theorem as follows:

$$\begin{aligned}
\mathrm{E}_S[R_{\boldsymbol{v}}] &= \mathrm{E}[(\boldsymbol{v}\cdot\boldsymbol{x} - y)^2] \\
&\leq \mathrm{E}\left[\frac{1}{m}\sum_{t=1}^{m} (\boldsymbol{w}_t\cdot\boldsymbol{x} - y)^2 \right] && \text{using (7)} \\
&= \frac{1}{m}\sum_{t=1}^{m} \mathrm{E}[(\boldsymbol{w}_t\cdot\boldsymbol{x} - y)^2] \\
&= \frac{1}{m}\sum_{t=1}^{m} \mathrm{E}[(\boldsymbol{w}_t\cdot\boldsymbol{x}_t - y_t)^2] && \text{using (9)} \\
&= \frac{1}{m}\mathrm{E}\left[\sum_{t=1}^{m} (\boldsymbol{w}_t\cdot\boldsymbol{x}_t - y_t)^2 \right] \\
&\leq \frac{1}{m}\mathrm{E}\left[\frac{\sum_{t=1}^{m} (\boldsymbol{u}\cdot\boldsymbol{x}_t - y_t)^2}{1-\eta} + \frac{\|\boldsymbol{u}\|_2^2}{\eta} \right] && \text{from Theorem 1.1} \\
&= \frac{1}{m}\left[\frac{\sum_{t=1}^{m} \mathrm{E}[(\boldsymbol{u}\cdot\boldsymbol{x} - y_t)^2]}{1-\eta} + \frac{\|\boldsymbol{u}\|_2^2}{\eta} \right] && \text{using (8)} \\
&= \frac{R_{\boldsymbol{u}}}{1-\eta} + \frac{\|\boldsymbol{u}\|_2^2}{\eta m},
\end{aligned}$$

which completes the proof.

4 Learning the Distribution

In the batch setting, we assume that the training and test examples are all i.i.d. from some fixed, but unknown distribution D. So far in class, we've studied algorithms for performing classification/regression without any assumptions about D. Next time we'll change focus by looking at ways to try to learn about and model the underlying distribution of the data.