E0 370 Statistical Learning Theory **Lecture 17 (Oct 15, 2013)**

Online Regression (and Classification)

Instructor: Shivani Agarwal *Scribe: Harikrishna Narasimhan*

1 Introduction

In our previous lectures, we had focused on the batch setting, where one is given a 'batch' of training examples, and the goal is to learn a function that yields minimum error on future examples. In this lecture, we shall start with the online learning setting, where learning takes place in a sequence of trials: on each trial, the learner must make a prediction or take an action, resulting in a potential loss, and the goal is to update the prediction model at the end of each trial so as to minimize the total loss over a sequence of trials; such setting arise in a variety of real-world applications, including prediction problems (e.g. forecasting the weather the next day) and decision/allocation problems (e.g. investing in different stocks or mutual funds).

In this lecture, we shall focus on online supervised learning problems, where in each trial, the learner receives an example, predicts its label, following which the true label is revealed to him and a corresponding loss is incurred; as noted above, the goal is to minimize the total loss over a sequence of trials. As part of last week's reading exercise, you were asked to review two classic algorithms for online classification, namely the perceptron and winnow algorithms (see Lecture 18 of the 2011 offering of this course). In this lecture, we will start with two algorithms for a general online regression/classification problem. In the next lecture, we shall study online learning from experts, a framework that can be useful for both online supervised learning problems and online decision/allocation problems; following this, we shall discuss algorithms for a general online convex optimization problem; we will then look at how online learning algorithms and their analyses can be transported back into the batch setting.

An online regression/classification setting can be described as follows:

Online Regression/Classification
For $t = 1 \dots T$ – Receive $x^t \in \mathcal{X}$ – Predict $\hat{y}^t \in \mathbb{R}$ – Receive true label $y^t \in \mathcal{Y}$ – Incur loss $\ell(y^t, \hat{y}^t)$

Let $S = ((x^1, y^1), \dots, (x^T, y^T)) \in (\mathcal{X} \times \mathcal{Y})^T$ be the sequence of examples received in T trials. The cumulative loss of an online algorithm $\mathcal{A}$ over the trial sequence S is given by

$$L_S^\ell[\mathcal{A}] = \sum_{t=1}^{T} \ell(y^t, \hat{y}^t).$$

Let $\mathcal{F} \subseteq \mathbb{R}^{\mathcal{X}}$. The loss of a fixed function $f \in \mathcal{F}$ over S is given by

$$L_S^\ell[f] = \sum_{t=1}^{T} \ell(y^t, f(x^t)).$$

We would like to compare the cumulative loss of algorithm $\mathcal{A}$ with that of an offline (batch) algorithm which on hindsight selects a function from $\mathcal{F}$ with minimum cumulative loss on S. More specifically, the performance of algorithm $\mathcal{A}$ is measured in terms of its regret with respect to $\mathcal{F}$:

$$L_S^\ell[\mathcal{A}] - \inf_{f \in \mathcal{F}} L_S^\ell[f].$$

Our goal is to design algorithms with low regret over any trial sequence (satisfying certain properties); in particular, we would like to design algorithms with worst-case regret bound guarantees. Note that the analysis here is worst-case, rather than probabilistic as in the earlier batch settings.

In the following, we shall describe two algorithms for online regression/classification, namely gradient descent (GD) and exponentiated gradient (EG) algorithms, and discuss regret bounds for both algorithms; we shall then see that both these algorithms are instantiations of a larger family of online learning algorithms.

2 Gradient Descent Algorithm

The basic gradient descent algorithm works with a Euclidean instance space $\mathcal{X} \subseteq \mathbb{R}^n$ and learns a linear function (represented by a weight vector) on $\mathcal{X}$. The algorithm assumes a loss function $\ell : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}_+$ that is subdifferentiable in its second argument[1], and is outlined below:

Gradient Descent

Loss $\ell : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}_+$ (subdifferentiable in its 2nd argument)
Initialize: $\mathbf{w}^1 = (0, \ldots, 0) \in \mathbb{R}^n$
Parameter: $\eta > 0$
For $t = 1 \ldots T$
- Receive $\mathbf{x}^t \in \mathcal{X} \subseteq \mathbb{R}^n$
- Predict $\hat{y}^t = \mathbf{w}^t \cdot \mathbf{x}^t$
- Receive true label $y^t \in \mathcal{Y}$
- Incur loss $\ell(y^t, \hat{y}^t)$
- $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \eta \ell'_{y^t}(\hat{y}^t)\mathbf{x}^t$

---

where in the last step, $\ell_y(\hat{y}) = \ell(y, \hat{y})$, and $\ell'_y(\hat{y})$ denotes any subderivative of ℓ_y at $\hat{y}$.

At each trial, the above algorithm performs a simple additive update, where a negative subgradient of the loss function ℓ, scaled by a constant η (often known as the learning rate), is added to the current weight vector. We shall now prove a regret bound for the gradient descent algorithm.

Theorem 2.1 (Cesa-Bianchi, 1999)**.** Let $S = ((\mathbf{x}^1, y^1), \ldots, (\mathbf{x}^T, y^T)) \in (\mathbb{R}^n \times \mathcal{Y})^T$ and let $\ell : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}_+$ be a loss function subdifferentiable in its second argument. Let $R_2 = \max\{||\mathbf{x}^t||_2 : \ t \in [T]\}$ and let $Z = \max\{|\ell'_{y^t}(\hat{y}^t)| : \ t \in [T]\}$. Let $U > 0$. Then

$$L_S^\ell[\mathrm{GD}(\eta)] - \inf_{\mathbf{u} \in \mathbb{R}^n : ||\mathbf{u}||_2 \leq U} L_S^\ell[\mathbf{u}] \ \leq \ \frac{1}{2}\Big(\frac{U^2}{\eta} + \eta R_2^2 Z^2 T\Big).$$

Moreover, setting $\eta^* = \frac{U}{R_2 Z \sqrt{T}}$ to minimize the right-hand side of the above bound (assuming R_2, Z, and T, or upper bounds on them, are known in advance), we have

$$L_S^\ell[\mathrm{GD}(\eta^*)] - \inf_{\mathbf{u} \in \mathbb{R}^n : ||\mathbf{u}||_2 \leq U} L_S^\ell[\mathbf{u}] \ \leq \ U R_2 Z \sqrt{T}.$$

Proof. Let $\mathbf{u} \in \mathbb{R}^n$ with $\|\mathbf{u}\|_2 \leq U$. For any $t \in [T]$,

$$\begin{aligned}
\ell(y^t, \hat{y}^t) - \ell(y^t, \mathbf{u} \cdot \mathbf{x}^t) &\leq (\hat{y}^t - \mathbf{u} \cdot \mathbf{x}^t)\ \ell'_{y^t}(\hat{y}^t) \ \ \text{(by subdifferentiability of } \ell) \\
&= (\mathbf{w}^t \cdot \mathbf{x}^t - \mathbf{u} \cdot \mathbf{x}^t)\ \ell'_{y^t}(\hat{y}^t) \\
&= \frac{1}{2\eta}\Big(\|\mathbf{u} - \mathbf{w}^t\|_2^2 - \|\mathbf{u} - \mathbf{w}^{t+1}\|_2^2 + \|\mathbf{w}^t - \mathbf{w}^{t+1}\|_2^2\Big) \ \ \text{(by definition of } \mathbf{w}^{t+1}) \quad (1) \\
&= \frac{1}{2\eta}\Big(\|\mathbf{u} - \mathbf{w}^t\|_2^2 - \|\mathbf{u} - \mathbf{w}^{t+1}\|_2^2 + \eta^2 \big(\ell'_{y^t}(\hat{y}^t)\big)^2 \|\mathbf{x}^t\|_2^2\Big) \\
&\leq \frac{1}{2\eta}\Big(\|\mathbf{u} - \mathbf{w}^t\|_2^2 - \|\mathbf{u} - \mathbf{w}^{t+1}\|_2^2 + \eta^2 Z^2 R_2^2\Big) \ \ \text{(by definition of } Z \text{ and } R_2).
\end{aligned}$$

[1] A loss function $\ell : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}_+$ is said to be subdifferentiable (in its second argument) if $\forall y \in \mathcal{Y}, \hat{y} \in \mathbb{R}$, $\exists \ell'_y(\hat{y}) \in \mathbb{R}$, s.t. $\ell(y, \hat{y}') - \ell(y, \hat{y}) \geq (\hat{y}' - \hat{y})\ell'_y(\hat{y}),\ \forall \hat{y}' \in \mathbb{R}$.

When we sum over $t = 1, \ldots, T$, the first two terms in the above expression form a telescopic summation, thus yielding

$$\begin{aligned} L_S^\ell[\mathrm{GD}(\eta)] - L_S^\ell[\mathbf{u}] &\leq \frac{1}{2\eta}\Big(\|\mathbf{u} - \mathbf{w}^1\|_2^2 - \|\mathbf{u} - \mathbf{w}^{T+1}\|_2^2 + \eta^2 Z^2 R_2^2 T\Big) \\ &\leq \frac{1}{2\eta}\Big(\|\mathbf{u} - \mathbf{w}^1\|_2^2 + \eta^2 Z^2 R_2^2 T\Big) \\ &\leq \frac{1}{2\eta}\Big(U^2 + \eta^2 Z^2 R_2^2 T\Big) \quad (\text{since } \mathbf{w}^1 = \mathbf{0}^n \text{ and } \|\mathbf{u}\|_2 \leq U) \\ &= \frac{1}{2}\Big(\frac{U^2}{\eta} + \eta Z^2 R_2^2 T\Big). \end{aligned} \tag{2}$$

It can be verified that the right-hand side of the above bound is minimum when

$$\eta^* = \frac{U}{R_2 Z \sqrt{T}}.$$

Substituting this back into Eq. (2) gives the desired result. □

Note that the regret bound stated in the above theorem has a *sublinear* dependence on T; thus the average (per-trial) regret for the gradient descent algorithm (over T trials) approaches 0 as T approaches ∞.

We shall now see that the perceptron algorithm for online classification is a special case of the gradient descent algorithm, albeit with a conservative update step (updates performed only on trials where the predicted label is different from the true label). Let $\mathcal{Y} = \{\pm 1\}$ and the loss function under consideration be the hinge loss $\ell_{\text{hinge}} : \{\pm 1\} \times \mathbb{R} \to \mathbb{R}_+$ defined as $\ell_{\text{hinge}}(y, \hat{y}) = (1 - y\hat{y})_+$. In a given mistake trial ($y^t \neq \hat{y}^t$), if $y^t = 1$, and $\hat{y}^t < 0$, we have

$$\left.\frac{\partial \ell_{\text{hinge}}(y^t, \hat{y})}{\partial \hat{y}}\right|_{\hat{y}=\hat{y}^t} = -1,$$

while if $y^t = -1$, and $\hat{y}^t > 0$, we have

$$\left.\frac{\partial \ell_{\text{hinge}}(y^t, \hat{y})}{\partial \hat{y}}\right|_{\hat{y}=\hat{y}^t} = 1.$$

Clearly, when $\eta = 1$, the gradient descent update step for the hinge loss (performed conservatively) is the same as the perceptron update step.

3 Exponentiated Gradient Algorithm

The exponentiated gradient descent algorithm (Kinvinen & Warmuth, 1997) also works with a Euclidean instance space $\mathcal{X} \subseteq \mathbb{R}^n$ and learns a linear function on $\mathcal{X}$:

Exponentiated Gradient

Loss $\ell : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}_+$ (subdifferentiable in its 2nd argument)
Initialize: $\mathbf{w}^1 = \left(\frac{1}{n}, \ldots, \frac{1}{n}\right) \in \mathbb{R}^n$
Parameter: $\eta > 0$
For $t = 1 \ldots T$
- Receive $\mathbf{x}^t \in \mathcal{X} \subseteq \mathbb{R}^n$
- Predict $\hat{y}^t = \mathbf{w}^t \cdot \mathbf{x}^t$
- Receive true label $y^t \in \mathcal{Y}$
- Incur loss $\ell(y^t, \hat{y}^t)$
- $\forall i \in [n]$:

$$w_i^{t+1} \leftarrow \frac{w_i^t e^{-\eta \ell'_{y^t}(\hat{y}^t) x_i^t}}{Z_t},$$

where $Z_t = \sum_{i=1}^n w_i^t e^{-\eta \ell'_{y^t}(\hat{y}^t) x_i^t}$.

Notice that unlike the gradient descent algorithm, the exponentiated gradient algorithm uses a multiplicative update in each trial. Define $\Delta_n = \{\mathbf{w} \in \mathbb{R}^n : \ w_i \geq 0, \sum_{i=1}^n w_i = 1\}$ as the n-dimensional probability simplex. We then have the following regret bound for the above algorithm.

Theorem 3.1 (Cesa-Bianchi, 1999)**.** Let $S = ((\mathbf{x}^1, y^1), \ldots, (\mathbf{x}^T, y^T)) \in (\mathbb{R}^n \times \mathcal{Y})^T$. Let $\ell : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}_+$ be a loss function subdifferentiable in its second argument. Let $R_\infty = \max\{\|\mathbf{x}^t\|_\infty : \ t \in [T]\}$ and let $Z = \max\{|\ell'_{y^t}(\hat{y}^t)| : \ t \in [T]\}$. Then,

$$L_S^\ell[\mathrm{EG}(\eta)] - \inf_{\mathbf{u} \in \Delta_n} L_S^\ell[\mathbf{u}] \ \leq \ \frac{\ln(n)}{\eta} + \frac{\eta R_\infty^2 Z^2 T}{2}.$$

Moreover, setting $\eta^* = \frac{\sqrt{2\ln(n)}}{R_\infty Z \sqrt{T}}$ to minimize the right-hand side of the above bound (assuming R_∞, Z, and T, or upper bounds on them, are known in advance), we have

$$L_S^\ell[\mathrm{EG}(\eta^*)] - \inf_{\mathbf{u} \in \Delta_n} L_S^\ell[\mathbf{u}] \ \leq \ R_\infty Z \sqrt{2T \ln(n)}.$$

Since the above regret bound is sublinear in T, the average (per-trial) regret for the exponentiated gradient algorithm goes to 0 as $T \to \infty$.

Just as the gradient descent algorithm applied (with conservative update) to the hinge loss yields the perceptron algorithm, it can be verified that the exponentiated gradient algorithm applied (with conservative updates) to the hinge loss results in the winnow algorithm.

4 Generalizations of GD and EG Algorithms

We shall now see that both the gradient descent and exponentiated gradient algorithms can be viewed as special cases of a larger family of gradient-based online learning algorithms. We start with a class of online learning algorithms, called 'follow-the-regularized-leader' algorithms, where the update on each trial t, involves a (regularized) batch optimization problem over all examples received so far:

$$\mathbf{w}^{t+1} \leftarrow \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^n} \left\{ \sum_{s=1}^{t} \ell(y^s, \mathbf{w} \cdot \mathbf{x}^s) + \frac{1}{\eta} F(\mathbf{w}) \right\},$$

where $F : \mathbb{R}^n \to \mathbb{R}$ is a strictly convex function (serving as a regularizer) and as before $\eta > 0$ is a parameter of the algorithm. For our purpose, we shall consider an approximate version of this update rule, where the loss function $\ell(y^s, \mathbf{w} \cdot \mathbf{x}^s)$ at trial s is replaced by its first-order Taylor approximation at $\mathbf{w}^s$, i.e., $\ell(y^s, \mathbf{w} \cdot \mathbf{x}^s)$ is replaced by $\ell(y^s, \mathbf{w}^s \cdot \mathbf{x}^s) + (\mathbf{w} - \mathbf{w}^s) \cdot \ell'_{y^s}(\hat{y}^s)\mathbf{x}^s$:

$$\mathbf{w}^{t+1} \leftarrow \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^n} \left\{ \sum_{s=1}^{t} \left(\ell(y^s, \mathbf{w}^s \cdot \mathbf{x}^s) + (\mathbf{w} - \mathbf{w}^s) \cdot \ell'_{y^s}(\hat{y}^s)\mathbf{x}^s \right) + \frac{1}{\eta} F(\mathbf{w}) \right\}. \tag{A}$$

Since the gradient of the above objective (w.r.t. $\mathbf{w}$) is 0 at $\mathbf{w}^{t+1}$, we have

$$\sum_{s=1}^{t} \ell'_{y^s}(\hat{y}^s)\mathbf{x}^s + \frac{1}{\eta} \nabla F(\mathbf{w}^{t+1}) = 0,$$

which can be rewritten as

$$\begin{aligned} \nabla F(\mathbf{w}^{t+1}) &= -\eta \sum_{s=1}^{t} \ell'_{y^s}(\hat{y}^s)\mathbf{x}^s \\ &= -\eta \sum_{s=1}^{t-1} \ell'_{y^s}(\hat{y}^s)\mathbf{x}^s - \eta \ell'_{y^t}(\hat{y}^t)\mathbf{x}^t \\ &= \nabla F(\mathbf{w}^t) - \eta \ell'_{y^t}(\hat{y}^t)\mathbf{x}^t. \end{aligned} \tag{3}$$

Note that by setting $F(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$ in the above equation, we obtain the gradient descent update; similarly, it can be verified that if optimization problem in (A) is constrained to the probability simplex Δ_n, and $F(\mathbf{w}) = \sum_{i=1}^n w_i \ln(w_i)$ (negative entropy), we obtain the exponentiated gradient update.

We shall next see that the approximate follow-the-regularized-leader update rule discussed above is in fact equivalent to the following update rule computed on a single example:

$$\mathbf{w}^{t+1} \leftarrow \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^n} \left\{ \ell(y^t, \mathbf{w}^t \cdot \mathbf{x}^t) + \frac{1}{\eta} B_F(\mathbf{w}, \mathbf{w}^t) \right\}, \tag{B}$$

where $B_F : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_+$ is the Bregman divergence associated with F, defined as $B_F(\mathbf{w}, \mathbf{w}') = F(\mathbf{w}) - F(\mathbf{w}') - (\mathbf{w} - \mathbf{w}') \cdot \nabla F(\mathbf{w}')$ for any $\mathbf{w}, \mathbf{w}' \in \mathbb{R}^n$. By setting the gradient of the above optimization objective (w.r.t. $\mathbf{w}$) to 0, we obtain the same condition for optimality as the previous update rule (see Eq. (3)), clearly implying that both the update rules are equivalent.

The above update rule (B) can be generalized to any convex set $\Omega \subseteq \mathbb{R}^n$, and strictly convex function $F : \Omega \to \mathbb{R}$, with the corresponding optimization problem constrained to Ω; thus we have a family of gradient-based algorithms, of which the gradient descent algorithm ($\Omega = \mathbb{R}^n$ and $F(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$) and the exponentiated gradient algorithm ($\Omega = \Delta_n$ and $F(\mathbf{w}) = \sum_{i=1}^n w_i \ln(w_i)$) are special cases. By generalizing the proof technique used in Theorem 2.1 for deriving a regret bound for the gradient descent algorithm, one can derive a regret bound for any algorithm in the above family, provided, in the step in Eq. (1), where one gets $\frac{1}{\eta}\big(B_F(\mathbf{u}, \mathbf{w}^t) - B_F(\mathbf{u}, \mathbf{w}^{t+1}) + B_F(\mathbf{w}^t, \mathbf{w}^{t+1})\big)$, the Bregman divergence term between weight vectors in consecutive trials, $B_F(\mathbf{w}^t, \mathbf{w}^{t+1})$, can be suitably upper bounded; for the exponentiated gradient algorithm, this step involves upper bounding the KL-divergence between probability vectors in consecutive trials, yielding the regret bound in Theorem 3.1.

A particularly interesting special case of the above family of algorithms is when $\Omega = \mathbb{R}^n$ and $F(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_q^2$, resulting in the so-called p-norm algorithms, where $2 \le p \le \infty$, $\frac{1}{p} + \frac{1}{q} = 1$ (Grove et al. 2001; Kinvinen & Warmuth, 2001). It has been shown that by selecting $p = O(\ln(n))$, one can obtain an algorithm with a regret bound that has similar logarithmic dependence on the number of dimensions n as in the case of the winnow or exponentiated gradient algorithms, without the need for tuning the learning parameter η based on quantities that are typically not known in advance (Gentile, 2003).

Exercise. Show that $F(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$ is strictly convex for any norm $\|\cdot\|$ on $\mathbb{R}^n$.

5 Next Lecture

In the next lecture, we shall study online learning from experts, a framework that can be useful for both online supervised learning problems and online decision/allocation problems.

References

[1] Nicolò Cesa-Bianchi. Analysis of two gradient-based algorithms for on-line regression. *J. Comput. Syst. Sci.*, 59(3):392–411, 1999.

[2] Claudio Gentile. The robustness of the p-norm algorithms. volume 53, pages 265–299. Kluwer Academic Publishers, 2003.

[3] Adam J. Grove, Nick Littlestone, and Dale Schuurmans. General convergence results for linear discriminant updates. *Machine Learning*, 43(3):173–210, 2001.

[4] J. Kivinen and M. Warmuth. Exponentiated Gradient versus Gradient Descent for Linear Predictors. *Information and Computation*, 132(1):1–63, 1997.

[5] J. Kivinen and M. K. Warmuth. Relative loss bounds for multidimensional regression problems. *Machine Learning*, 45(3):301–329, December 2001.