CS/CNS/EE 253: Advanced Topics in Machine Learning
Topic: Online Convex Optimization and Online SVM **Lecturer:** Daniel Golovin
Scribe: Xiaodi Hou **Date:** Jan 13, 2010

4.1 Online Convex Optimization

Definition 4.1.1 *In Euclidian space, a set $\mathcal{C}$ is said to be* convex, *if $\forall x, y \in \mathcal{C}$, and $t \in [0,1]$, $z = (1-t)x + ty$ is in $\mathcal{C}$.*

Definition 4.1.2 *A function $f : \mathcal{D} \to \mathbb{R}$ is called* convex, *if $\forall x, y \in \mathcal{D}$, and $t \in [0,1]$, $f((1-t)x + y) \leq (1-t)f(x) + tf(y)$.*

Let the feasible set $\mathcal{X} \subseteq \mathbb{R}^n$ be a convex set. We have T convex cost functions: $c^1, c^2, \ldots, c^T$, where each of the functions is defined as $c^i : \mathcal{X} \to [0,1]$.

Theorem 4.1.3 (Zinkevich '03 [1]) *Zinkevich [1] proposed an algorithm for online convex optimization:*

1. *Choose $\mathbf{x}^1$ arbitrarily in $\mathcal{X}$*
2. *Update $\mathbf{x}^{t+1} = Proj_{\mathcal{X}}\big(\mathbf{x}^t - \eta_t \cdot \nabla c^t(\mathbf{x}^t)\big)$*
   where η_t is a non-increasing function of t. Common choices are $\eta_t = \frac{1}{\sqrt{t}}$, or $\eta_t = \frac{1}{t}$.

Using $\eta_t = 1/\sqrt{t}$ the regret of this online algorithm is bounded by:

$$\sum_{t=1}^{T} c^t(\mathbf{x}^t) - c^t(\mathbf{z}^t) \leq \frac{D^2}{2}\sqrt{T} + G^2\sqrt{T} + 2D \cdot L(\mathbf{z}^1, \mathbf{z}^2, \ldots, \mathbf{z}^T)\sqrt{T},$$

where $D = \max_{\mathbf{x},\mathbf{y} \in \mathcal{X}} \|\mathbf{x} - \mathbf{y}\|_2$ is the radius of the set; $\forall t, \forall \mathbf{x} \in \mathcal{X}, \|\nabla c^t(\mathbf{x})\|_2 \leq G$ is the upper bound of the gradient, and L is the total length of the drift, from $\mathbf{z}^1$ to $\mathbf{z}^T$, i.e., $L(z^1, \ldots, z^T) := \sum_{i=1}^{T-1} \|z^{i+1} - z^i\|_2$.

One example of how we can use this algorithm is as an alternative to the Hedge algorithm, in the case where we have n experts. For this, we construct a dimension for each expert, so that our feasible region lies in $\mathbb{R}^n$. More specifically, we have:

$$\mathcal{X} = \Big\{\mathbf{x} : \mathbf{x}_i \in [0,1]; \sum_{i=1}^{n} \mathbf{x}_i = 1\Big\}.$$

A feasible vector $\mathbf{x}$ then encodes a distribution over experts, where exactly one expert is chosen, and expert i is chosen with probability $\mathbf{x}_i$.

An example of the feasible region is shown in Fig. 4.1.1.

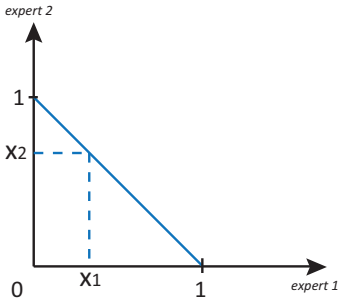

Figure 4.1.1: An example of the feasible region $\mathcal{X}$ in 2D space.

The projection operation can be very complex for an arbitrary convex set $\mathcal{X}$. Ideally we want to find the projection:

$$\text{Proj}(\mathbf{y}) = \underset{\mathbf{x}\in\mathcal{X}}{\text{argmin}} \|\mathbf{y} - \mathbf{x}\|_2. \tag{4.1.1}$$

4.2 Support Vector Machine

In this section, we will switch some of the previous notations. The data points are denoted as $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^T \in \mathbb{R}^n$; the labels y^i are binary variables: $y^1, y^2, \ldots, y^T; \quad y^i \in \{-1, 1\}$. A linear classifier can be considered as a hyperplane with normal vector $\mathbf{w} \in \mathbb{R}^n$ and offset b. The classification of $\mathbf{x}^i$ is determined by the hyperplane:

$$\tilde{y}^i = \text{sign}(\mathbf{w} \cdot \mathbf{x}^i + b). \tag{4.2.2}$$

4.2.1 Eliminating b by augmenting one dimension

Eq. 4.2.2 can be expressed in a simpler way, by augmenting $\mathbf{x}$ and $\mathbf{w}$. Let $\mathbf{x}^+ = [x^1, x^2, \ldots, x^n, 1] \in \mathbb{R}^{n+1}$, and $\mathbf{w}^+ = [w^1, w^2, \ldots, w^n, b]$, therefore:

$$\tilde{y} = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b) = \text{sign}(\mathbf{w}^+ \cdot \mathbf{x}^+).$$

For efficiency, we substitute $\mathbf{x}$ and $\mathbf{w}$ with the augmented vectors $\mathbf{x}^+$ and $\mathbf{w}^+$.

4.2.2 Hinge loss

The objective of a linear classifier is to find the hyperplane that "optimally" separates the positive samples from negative ones. In SVM, such optimality is defined as maximizing the margins, or minimizing the hinge loss.

$$\mathbf{w}^\star = \underset{\mathbf{w}}{\text{argmin}} \sum_{t+1}^{T} \text{hinge}(\mathbf{x}^t, y^t, \mathbf{w}), \quad s.t. \|\mathbf{w}\|_2 \leq \lambda. \tag{4.2.3}$$

where the hinge function is defined as:

$$\text{hinge}(\mathbf{x}, y, \mathbf{w}) \equiv \max\big(0, 1 - y(\mathbf{x} \cdot \mathbf{w})\big) \tag{4.2.4}$$

The hinge function is the least convex upper-bound of the $0-1$ loss function. Both functions are drawn in Fig. 4.2.2.

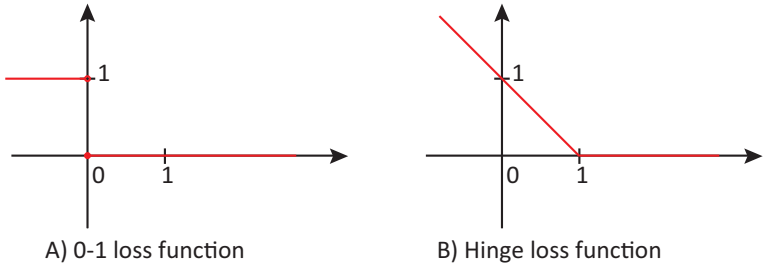

Figure 4.2.2: Figure A: the 0-1 loss function. Figure B: the hinge loss function.

4.2.3 Online SVM

Given the data points $\mathbf{x}^1, \mathbf{x}^2, \ldots \mathbf{x}^T \in \mathbb{R}^{n+1}$, and the corresponding labels $y^1, y^2, \ldots, y^T \in \{-1, 1\}$, the feasible set of the hyperplane $\mathcal{W} = \{\mathbf{w} : \|\mathbf{w}\|_2 \leq \lambda\}$, and the loss function as hinge function, we have the algorithm for training SVM in an online fashion:

1. Pick $\mathbf{w}^1 \in \mathcal{W}$ arbitrarily.
2. For $t = 1, 2, \ldots T$, the incurred loss is $c^t(\mathbf{w}^t) \equiv \text{hinge}(\mathbf{x}^t, y^t, \mathbf{w}^t)$.
3. Step forward on the gradient direction: $\hat{\mathbf{w}}^{t+1} = \mathbf{w}^t - \eta_t \nabla c^t(\mathbf{w}^t)$.
4. Finally, project $\hat{\mathbf{w}}^{t+1}$ back to the feasible set: $\mathbf{w}^{t+1} = \text{Proj}_{\mathcal{W}}(\hat{\mathbf{w}}^{t+1})$.

We note that Eq. 4.2.3 is not differentiable. To overcome this problem, we use a "subgradient" in leiu of the gradient.

4.2.3.1 Subgradient

Let $c : I \to \mathbb{R}$ be a convex function defined on an open interval of the real line. As shown in Fig.4.2.3, c is not differentiable at x_0. A subgradient of c at x_0 is any vector v such that:

$$\forall x : c(x) - c(x_0) \geq v \cdot (x - x_0).$$

The subgradient is not unique. In general, the set of subgradients of c at x_0 is a convex set. One way to think about a subgradient v of c at x_0 is that it defines a linear lower bound for c that equals it at x_0, namely,

$$\ell_{v,x_0}(x) := c(x_0) + v \cdot (x - x_0).$$

For the hinge loss function, we can pick a subgradient $\mathbf{v}^t$ at $\mathbf{w}^t$ as following:

$$\begin{array}{rl} \mathbf{0}, & \text{if } y^t(\mathbf{w}^t \cdot \mathbf{x}^t) \geq 1 \\ -y^t\mathbf{x}^t, & \text{if } y^t(\mathbf{w}^t \cdot \mathbf{x}^t) < 1. \end{array}$$

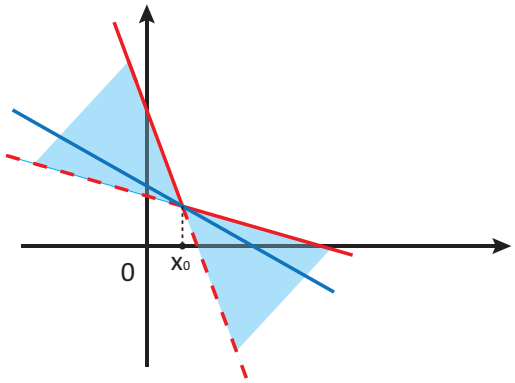

Figure 4.2.3: A convex function and its subgradient. Red solid line is the function $f(x)$. The subgradient of f at x_0 is the derivative of any blue line in the blue region that passes through x_0.

4.2.3.2 Projection

For a feasible set $\mathcal{W} = \{\mathbf{w} : \|\mathbf{w}\|_2 \leq \lambda\}$, the projection from $\hat{\mathbf{w}} \notin \mathcal{W}$ to its nearest point in $\mathcal{W}$ can be done by multiplying $\hat{\mathbf{w}}$ with a scalar:

$$\mathbf{w}^{t+1} = \text{Proj}(\hat{\mathbf{w}}^{t+1}) = \frac{\hat{\mathbf{w}}^{t+1} \cdot \lambda}{\|\hat{\mathbf{w}}^{t+1}\|}. \tag{4.2.5}$$

Of course, if $\hat{\mathbf{w}} \in \mathcal{W}$ then $\text{Proj}(\hat{\mathbf{w}}) = \hat{\mathbf{w}}$.

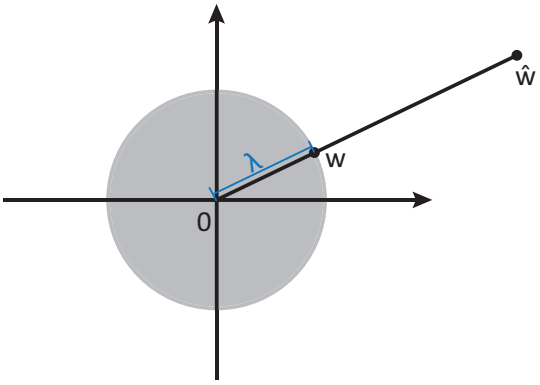

Figure 4.2.4: An illustration of the projection. Gray disk is the feasible set $\mathcal{W}$ with radius λ. $\hat{\mathbf{w}}$ is projected onto $\mathcal{W}$ to have $\mathbf{w}$.

4.3 Parallel Online SVM

In a recent paper [2], Zinkevich et al. proposed a parallel algorithm for Online SVM. In this scenario, the gradient is computed in a asynchronous way. At round t, the fetched gradient $\nabla c^{t-\tau}(w^{t-\tau})$ is the result at τ^{th} previous round. Zinkevich et al. proved that the online learning with delayed updates converges well. Therefore the parallel online learning can be achieved:

1. Choose $\mathbf{w}^1$ arbitrarily in $\mathcal{W}$
2. Update $\mathbf{w}^{t+1} = \text{Proj}_{\mathcal{W}}\big(\mathbf{w}^t - \eta_t \cdot \nabla c^{t-\tau}(\mathbf{w}^{t-\tau})\big)$
   where $\eta_t \approx \dfrac{1}{\sqrt{t}}$, or $\eta_t = \dfrac{1}{t}$ are common choices.

References

[1] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 928–936, 2003.

[2] Martin Zinkevich, Alex Smola, and John Langford. Slow learners are fast. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 2331–2339. 2009.