Statistical Techniques in Robotics (16-831, F14)   Lecture#12 (Tuesday October 07)

Convex Optimization and Subgradients; Regret Bounds

Lecturer: Drew Bagnell   *Scribes: Allie Del Giorno, Abhijeet Tallavajhula* [1]

1 More on regret

The bound on expected regret for the randomized weighted majority algorithm is

$$E[R] \leq \epsilon \sum_t L_t(e_t^*) + \frac{1}{\epsilon} \log N$$

Where e_t^* is the best performing expert in hindsight at time t. What is a good choice for ϵ? We'd like to minimize the upper bound (so we can send the average regret $\frac{R}{T}$ to 0 in the limit to prove no-regret, for instance). We can (and will) find the exact minimum, but let's look at what its relation to T (the number of data points we've seen) should look like. The upper bound is a sum of two terms. Suppose we choose $\epsilon \sim \mathcal{O}(T)$ to make the second disappear with rising T. The first term then grows as $\mathcal{O}(T^2)$, because $\sum_t L_t(e_t^*)$ grows at worst as $\mathcal{O}(T)$. If we instead choose $\epsilon \sim \mathcal{O}(\frac{1}{T})$, then the first term in the bound is $\mathcal{O}(1)$ but the second grows as $\mathcal{O}(T)$. $\epsilon \sim \mathcal{O}(\frac{1}{\sqrt{T}})$ is a good choice, because it sets the upper bound to $\mathcal{O}(\sqrt{T})$, which is sub-linear growth, and the average regret $\frac{\mathbb{E}[R]}{T}$ will go to 0.

The no-regret property does not imply that $\sum_t \frac{L_t(\text{algo})}{T} \to 0$. What about the converse? First we restrict ourselves to non-zero loss. If we know that $\sum_t \frac{L_t(\text{algo})}{T} \to 0$ for a particular sequence of data, that does not tell us that the regret for the algorithm goes to zero, because the no-regret definition is applied to performance on *all* sequences of data.

2 An application to online learning

Consider a mobile robot operating in the outdoors. We would like its perception system to accurately classify terrain. Terrain classification works better at close than far ranges. However, the requirement of getting close to a rock or bush to tell them apart leads to myopic behaviour. If we could detect terrain at a distance, we could built better plans for the robot. We want to take advantage of a perception system that works well up close to remember what terrain looks like from afar. Online learning can be applied to this problem of near-to-far self-supervised learning.

Say we want to distinguish rocks from bushes from range data. The hypothesis space is linear classifiers θ on features f,

$$\theta^T f \geq 0, Y = 1 \text{ (rock)}$$
$$\theta^T f < 0, Y = 0 \text{ (bush)}$$

Features on range data can be

[1]Some content adapted from previous scribes: Shervin Javdani

- shape (fit ellipsoid to points; get the ellipsoid's parameters as features),
- height (bushes are typically taller than rocks),
- spread,
- density/ sparsity (rockes result in dense returns),
- color (bushes are typically green!).

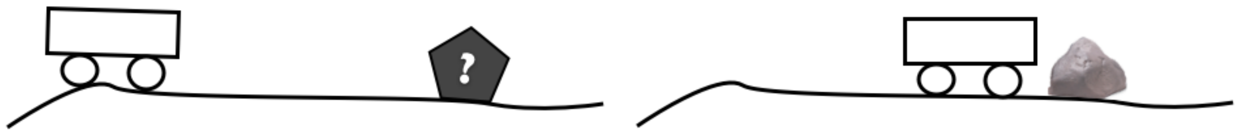

Figure 1: Self Supervised learning example. The robot must predict the label from far away and can acquire labels to figure out if it was correct when it gets closer to the object.

Let's use the randomized weighted majority algorithm. The experts make predictions from range data at a distance. Labels are received when the robot gets close, as shown in Fig. 1, and weights on the experts are updated. As experts we choose all possible classifiers. This is difficult as is because the space of θ is unbounded and continuous. Let the parameters be regularized by the condition $|\theta|_\infty \leq 1$, which defines an origin-centered box. Further, discretize space in the box at intervals of size a in each dimension. If $\theta \in \mathrm{R}^d$, this results in $(2/a)^d$ experts, as shown in Fig.2.

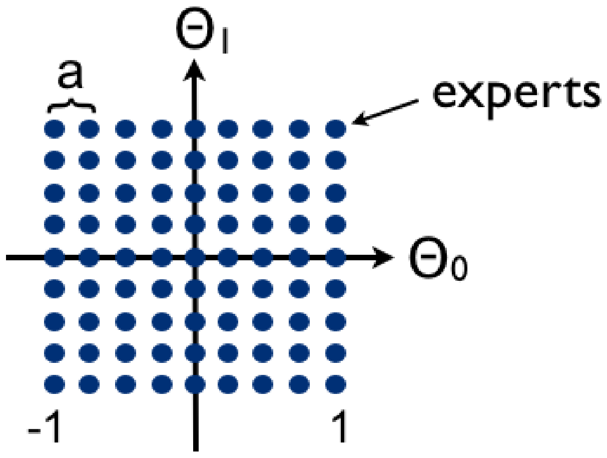

Figure 2: Discretized set of experts in two dimensions.

We choose the $0-1$ loss function. The regret bound looks like

$$\mathrm{E}[R] \leq \epsilon T + \frac{1}{\epsilon} \log N$$

The optimal learning rate is found by setting the derivative of the upper bound with ϵ to zero:

$$\begin{aligned}\frac{\partial R}{\partial \epsilon} = T - \frac{1}{\epsilon^2}\log N = 0\\ T = \frac{1}{\epsilon^2}\log N\\ \epsilon = \sqrt{\frac{\log N}{T}}\end{aligned}$$

So

$$\begin{aligned}R \leq \epsilon T + \frac{1}{\epsilon}\log N\\ R \leq 2\sqrt{T\log N}\\ \text{or, } R \leq 2\sqrt{Td\log\frac{2}{a}}\\ \text{or, } \frac{R}{T} \leq 2\sqrt{\frac{d}{T}\log\frac{2}{a}}\end{aligned}$$

which means that, as the dimension d of θ increases, the number of samples that need to be observed (T) to maintain the same value of the upper bound are $\mathcal{O}(d)$. Drew's rule of thumb is to wait for $T = 10d$ samples.

While it is nice that the number of samples scales linearly with the dimension of the features, other issues remain. One is that the distribution of terrain trained on depends on the planner. If the planner avoids rocks, the robot can get by with inaccurate rock classification. Another issue is that the number of experts is still exponential in d and storing and polling them is expensive. We address this by further limiting the experts in a way that leads to similar regret bounds, but lowers the computational requirements:

- The set of experts will be a convex set.
- The loss function l_t will be convex over the experts.

For the following, we will think of experts in the continuous context (it makes more sense).

3 Convexity

Convex sets Convex sets are set of points in which every point can "see" every other point (Figure 3). In other words, for every pair of points we choose, the set of points of the chord between them fall within the convex set. Formally, if a set C is convex,

$$a \in C, b \in C \Rightarrow \alpha a + (1-\alpha)b \in C.$$

Simple examples of convex sets: all space, straight lines half spaces, bowl spaces, and all p norms for $p \geq 1$ (Fig. 4[2]).

[2]Figure courtesy of `http://upload.wikimedia.org/wikipedia/commons/f/f9/Norm-vergleich.gif`

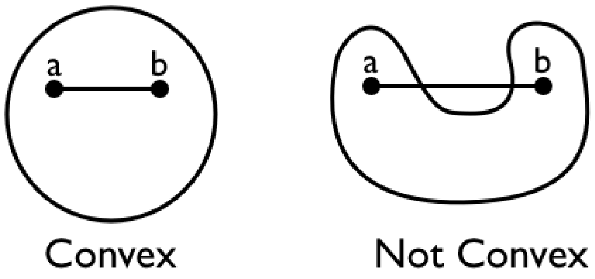

Figure 3: Convex and non-convex sets. A chord connecting any two points in the set must lie completely within the set.

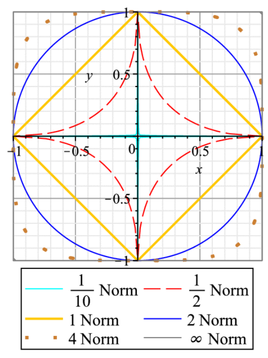

Figure 4: The p norms for $p \geq 1$ are convex sets.

Operators: Intersections and Projections Intersections preserve convexity: If C_1, C_2 convex, $C_1 \cap C_2$ is also convex. All convex sets can be described as intersections of half spaces.

Every convex set has a corresponding projection, an operator that takes a point P and maps it to a point in the set, P'. Projection of a point P onto the convex set is the point in the set P is closest to. Interestingly this projection P' is also closer to every other point in the convex set than P is (Fig. 5).

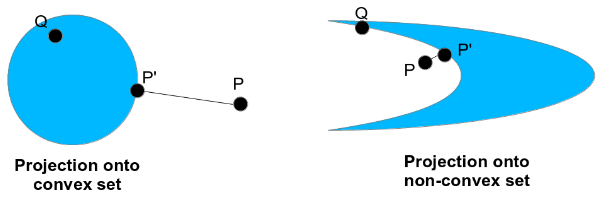

Figure 5: Any point P projected to a point P' on a convex set will get closer to any other point Q in that set. This does not hold for non-convex sets.

We're always going to think of a set of experts as lying on a convex set.

Convex functions Our loss function will also be convex. Let's discuss what it means for a function to be convex.

First we need to define an epigraph. The epigraph of a function $f(x)$ is the set of all points on or above $f(x)$. Formally:

$$\forall x, (x, y) \geq f(x)$$

If the epigraph is a convex set, then the function is convex.

There are other ways to characterize a convex function. These are sufficient but not necessary ways to prove convexity (in order of increasing generality):

- If the second derivative is greater than 0, the function is convex.
- If the first derivative is monotonic non-decreasing, the function is convex.
- If the subgradients (discussed below) exist on all points along a function, the function is convex.

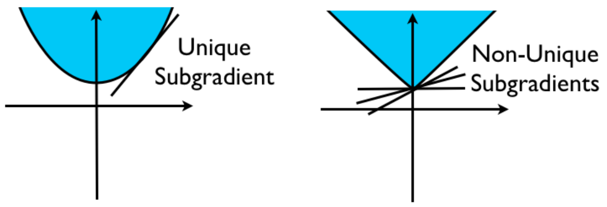

Figure 6: Two convex functions. The left function has a unique subgradient everywhere; the right function has several subgradients at 0. Notice even though the right function is non-differentiable at that point, we can still identify it as convex using subgradients.

Subgradients are a generalization of gradients for convex functions. In other words, we can still test convexity even if the function is not first- or second- order differentiable. A subgradient $\nabla f(x)$ at a point on $f(x)$ is defined as as any vector that is normal to $f(x)$, and remains below the rest of f. Formally:

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) \qquad \forall y \tag{1}$$

Why are we using convex loss functions? Convex have cool properties. For instance, if a function is convex, its local minimum is also its global minimum. (note the reverse is not true – not every function whose local and global minima match is convex). This type of property is great because certain local statements about the function (such as its minimum) imply global statements about the function over all points. That means we don't have to test a high-dimensional set of points and we require less computation to make statements with convex loss functions.