17-740 Spring 2025

Algorithmic Foundations of Interactive Learning

Lecture 3: Information Theory and Maximum Entropy

Lecturer: Drew Bagnell
Scribe: Zora Wang, Lindia Tjuatja

3.1 Recap: Regret

Recall that the *regret* of some algorithm is the difference in loss l between the algorithm and the best expert $e^\star$ over all timesteps $t \in [T]$:

$$\text{Regret(T)} = \sum_{t}^{T} \ell_t(\text{alg}_t) - \ell_t(e^\star) \tag{3.1}$$

As we saw in the previous lecture, there exist algorithms (such as weighted majority) where the regret is sub-linear in T. We say an algorithm is *no-regret* when

$$\frac{\text{Regret}(T)}{T} \to 0. \tag{3.2}$$

For example, for weighted majority, $\text{Regret}(T) \leq \mathcal{O}(\log(N)\sqrt{T})$, where N is the number of experts we are selecting between.

3.2 Application: Linear Classifiers

We now explore how we can apply the idea of *learning from expert advice* to the standard machine learning problem of learning a linear classifier. Intuitively, we will partition the parameter space into a set of "experts", and attempt to compete with the best of these experts (i.e., the best classifier).

Suppose we have a linear classifier with parameters $\theta \in \Theta \subset \mathbb{R}^d$ with some input $x \in \mathbb{R}^d$:

$$f_\theta(x) = \begin{cases} \theta^T x \geq 0 : \text{Output } 1 \\ \theta^T x < 0 : \text{Output } -1 \end{cases} \tag{3.3}$$

As mentioned above, one way can frame this problem as learning from expert advice by partitioning Θ into a set of experts and running, say, multiplicative weights to pick a classifier.

Observe that from the definition of f_θ, only the direction and not the scale of θ matters. Thus, we can set Θ to be the unit sphere in d dimensions without loss of generality. If we split each of the d dimensions into b points, we get $\mathcal{O}(b^d)$ experts, roughly speaking. Recalling that the regret of WM scales logarithmically with N, it immediately follows that we pay linearly in the dimension, d. While that is nice, note that we need to do an *exponential* amount of work per iteration of WM because the number of experts still scales exponentially in d, making this a rather computationally inefficient approach.

3.3 Convexity

We now explore a few definitions related to convexity.

Definition 1 (Convex Function) *A function f is convex if, $\forall \alpha_1, \alpha_2 > 0$ s.t. $\alpha_1 + \alpha_2 = 1$ and $x_1, x_2 \in dom(f)$, we have:*

$$\sum_i^2 \alpha_i f(x_i) \geq f\left(\sum_i^2 \alpha_i x_i\right). \tag{3.4}$$

In words, this definition is saying that the line connecting two points on a convex function is on or above the function. Note that this also implies that all linear functions are convex – strong convexity excludes lines. We can generalize this observation via induction: given any 3 points on a convex function $(x_1, f(x_1)), (x_2, f(x_2)), (x_3, f(x_3))$, we can trace out a triangle by taking different linear combinations of the weights which lies above the function:

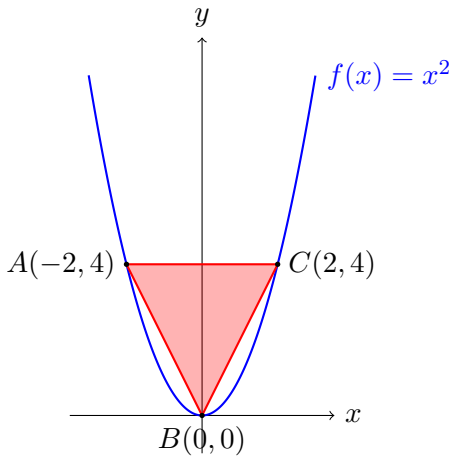

Figure 3.1: We can trace out the triangle in red with different linear combinations of points A, B, C (i.e. $\alpha_1, \alpha_2, \alpha_3 \geq 0$ with $\sum_i^3 \alpha_i = 1$. Convexity means this triangle lies above the function (other than at the 3 intersecting points).

The more general version of this above observation for an arbitrary number of points is known as Jensen's Inequality:

Lemma 1 *For a convex function f and probability distribution $p \in \Delta(dom(f))$, we have*

$$\mathbb{E}_p[f(x)] \geq f(E_p[f(x)]). \tag{3.5}$$

In words, the average value of a convex function lies at or above the evaluation of the function at the average input. [1]

3.4 Information Theory

Information theory is a theory of compressing, communicating, and quantifying information, like a message. We now outline some basic quantities in information theory.

3.4.1 Entropy, Cross-Entropy, and KL-Divergence

Consider a random variable X with sample space $\{a, b, c, d\}$. Let's say $P(X = a) = \frac{1}{2}$, $P(X = b) = \frac{1}{4}$, $P(X = c) = \frac{1}{8}$, and $P(X = d) = \frac{1}{8}$. Let's try to encode samples from

[1] If you forget which way Jensen's Inequality goes, a convenient heuristic is that it is the reverse of whatever you need to complete the proof.

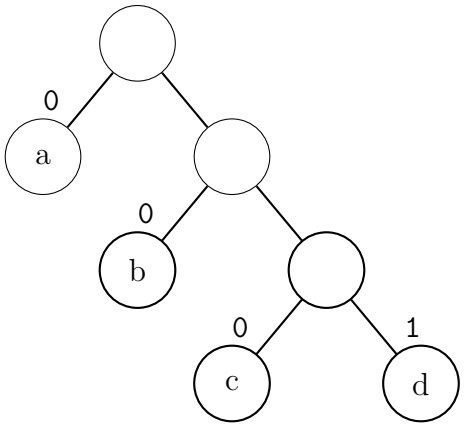

Figure 3.2: Prefix tree representation of optimal code of $P(X)$.

$P(X)$ in binary. Specifically, our goal will be to design a code (i.e. a mapping from the sample space to binary strings) to minimize the expected number of bits required to encode a sample from $P(X)$ is minimized.

Table 3.1 shows an optimal code for X. As one might expect, more probable values of X have shorter representations. Furthermore, this code is *instantaneously decodable*, i.e. we can, by reading the bits in sequence, tell when a character has completed without the need to insert explicit delimiters. This is why we don't use, say, 1 as the encoding of any value. This also means that we can write the code in a tree-structured manner, and is why it is sometimes referred to as a *prefix code*. Also observe that we set the length of the encoding of some member of our alphabet to be $\log_2(\frac{1}{P(X=\cdot)})$ – this value is sometimes refered to as the *information content* or *surprisal* of the outcome.

Table 3.1: Optimal prefix code for $P(X)$

Value	**Prob.**	**Code**	**Length**
a	1/2	`0`	1
b	1/4	`10`	2
c	1/8	`110`	3
d	1/8	`111`	3

To calculate the expected number of bits required to encode a sample using this optimal scheme, we can simply weight the information content of each out come by the probability of its occurrence. This value is known as *(Shannon) entropy*:

Definition 2 *Given a probability distribution $p \in \Delta(\mathcal{X})$, we can define the (Shannon) en-*

tropy of p as

$$\mathbb{H}(p) = \mathbb{E}_{x\sim p}[-\log_2(p(x)]. \tag{3.6}$$

Observe that entropy does not depend on permutations of the alphabet, nor the dimension of the points in the distribution. The *maximum entropy* distribution (i.e., the most uncertain) is the uniform distribution. Thus, for discrete outcome spaces, $\mathbb{H}(p) \leq \log_2(N)$, where N is the number of possible outcomes. We visualize the special case of a Bernoulli random variable below.

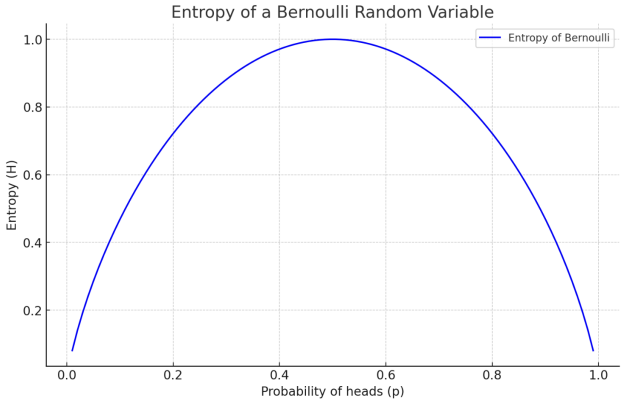

Figure 3.3: We plot the entropy of a **Bern**(p) random variable of as a function of p. Observe that entropy is maximized at $p = 0.5$ (i.e., a fair coin) with value $\log_2 2 = 1$.

Above, we coded $P(X)$ according to $P(X)$. What if we instead code some $Q(X)$ according to $P(X)$? We can write the expected length of our code by changing the distribution we're taking the expectation with respect to, a quantity known as *cross entropy*:

Definition 3 *Given distributions $P, Q \in \Delta(\mathcal{X})$, we can define the cross entropy from Q to P as*

$$\mathbb{H}(Q||P) = \mathbb{E}_{x\sim Q}[\ -\log_2 P(x)]. \tag{3.7}$$

Clearly, this is minimized when $P = Q$, recovering entropy.

The amount of suboptimality incurred by encoding with the wrong probability distribution (i.e., the difference between entropy and cross-entropy) is known as the *KL Divergence*:

Definition 4 *Given distributions $P, Q \in \Delta(\mathcal{X})$, we can define the cross entropy from Q to P as*

$$\mathbb{D}_{KL}(Q||P) = \mathbb{E}_{x\sim Q}\left[\log_2 \frac{Q(x)}{P(x)}\right] = \mathbb{H}(P||Q) - \mathbb{H}(Q). \tag{3.8}$$

Intuitively, this is a measure of how many extra bits we need to pay for picking the wrong encoding scheme (P instead of Q). The KL divergence is minimized to zero only $P = Q$ (and only when this is true). This is sometimes known as Gibbs' Inequality:

Lemma 2 *Given any distributions $P, Q \in \Delta(\mathcal{X})$, $\mathbb{D}_{KL}(P||Q) \geq 0$.*

Proof: We proceed via Jensen's:

$$-\mathbb{D}_{\mathrm{KL}}(P||Q) = -\sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right) \tag{3.9}$$

$$\leq \log\left(\sum_{x \in \mathcal{X}} P(x) \frac{Q(x)}{P(x)}\right) \tag{3.10}$$

$$= \log\left(\sum_{x \in \mathcal{X}} Q(x)\right) = \log(1) = 0. \tag{3.11}$$

Thus, $\mathbb{D}_{\mathrm{KL}}(P||Q) \geq 0$. ■

We can view learning a probability distribution as attempting to maximally compress some set of samples. More formally, we can view *maximum likelihood estimation* (MLE) as minimizing the KL Divergence from reality to our model. In this sense, MLE is an information-theoretic algorithm.

3.5 The Principle of Maximum Entropy

We now consider a problem that, while perhaps seeming somewhat abstract at first, will underlie many of the concepts we will explore in this course: *give partial information about some distribution (e.g. its mean), what is the right choice of distribution to pick out of all that match the known information?* The **Principle of Maximum Entropy** says to pick the highest entropy distribution that satisfies the given set of constraints. As we will discuss more, one justification for MaxEnt is *minimax optimality*: if an adversary were to pick another distribution that satisfies the given constraint, we'd suffer the least pain the worst case by picking the MaxEnt distribution. In this sense, we get a form of robustness.

Example 1: 8-Sided Die. Consider rolling a die with 8 faces. What is the optimal coding scheme for the outcome?

MaxEnt says to pick the distribution that maximizes entropy subject to the given information. Given we have no information, the MaxEnt distribution would be the uniform

distribution, which gives us an expected code length / entropy of 3. We won't prove this in class but, if an adversary were to view our code and then pick a true distribution that matched the given information, MaxEnt would prevent us from having a high KL divergence. Intuitively, we'll never put 0 probability mass on some outcome, or the adversary could make us output a code of infinite length by putting any weight on it.

Ok, that was a fairly boring example. Let's add in some known information. Using f to denote the number of dots on the top face, $\mathbb{E}_p[f]$ tells us how many dots we should expect, on average.

Example 2: Biased 8-sided Die. Let's say we have the same setting as above, but we now know $\mathbb{E}_p[f(x)] = 4.5$. In other words, the die is biased. The MaxEnt principle says we should chose

$$p^\star = \arg \max_{p \in \Delta(\mathcal{X})} \mathbb{H}(p) \tag{3.12}$$

$$\text{s.t. } \mathbb{E}_p[f(x)] = 4.5, \tag{3.13}$$

where $\Delta(\mathcal{X})$ is the probability simplex (i.e., the set of probability distributions) over $\mathcal{X}$. Given $\Delta(\mathcal{X})$ is a convex set, $\mathbb{H}$ is a convex function, and expectation / moment constraints are linear, this is a *convex optimization problem* (i.e., there is a unique optimal solution with an efficient algorithm to find it – the best you can hope for in machine learning!). We note in passing that maximizing entropy is equivalent to minimizing the KL divergence to the uniform distribution. One can naturally substitute this uniform distribution for a better informed prior, leading to the *principle of minimum cross-entropy* (MinRelEnt for short). We will revisit this point when we discuss RLHF.

We will explore how to solve this problem via forming the Lagrangian next time!