COS 511: Foundations of Machine Learning

Rob Schapire
Scribe: Melissa Carroll

Lecture #19
April 20, 2006

1 Review

When we left off last time, we established the following problem:

Given:
a space X where $|X| = N$
examples $x_1, ..., x_m \in \mathbf{X}$
$x_i \sim D$
features $f_1, ..., f_n$
$f_j : X \to \Re$
Our goal is to estimate D

The solution to this problem using *Maximum Entropy* is the distribution of maximum entropy from the set

$$P = \left\{ p : \forall j E_p[f_j] = \hat{E}_p[f_j] \right\}$$

The solution to this problem using *Maximum Likelihood* is the distribution of maximum likelihood from the set

$$Q = \left\{ q : q(x) = \frac{\exp(\sum_j x_j f_j(x))}{Z} \right\}$$

(all distributions $q \in Q$ are Gibbs distributions)

The Duality Theorem states that the following are equivalent and any one of these uniquely defines $q^\star$:

1. $q^\star = \arg\max_{p \in P} H(p)$ i.e. the Maximum Entropy approach
2. $q^\star = \arg\max_{q \in \overline{Q}} \sum_i \ln q(x_i)$ i.e. the Maximum Likelihood approach
3. $q^\star \in P \cap \overline{Q}$

2 Motivation

Now we ask, how do we actually find $q^\star$? What algorithm will we use? In coming up with an algorithm, #2 above (the *Maximum Likelihood* approach) will be the most useful. We will attempt to find the following:

$$\min_{q \in \overline{Q}} -\frac{1}{m} \sum_i \ln q(x_i)$$

More explicitly, we attempt to find a vector $\boldsymbol{\lambda}$ of parameter values which minimize the negative log likelihoood:

$$L(\boldsymbol{\lambda}) = -\frac{1}{m} \sum_i \ln q_\lambda(x_i) \tag{1}$$

where $q_\lambda(x)$ is a Gibbs distribution defined by $\boldsymbol{\lambda}$. Thus we have

$$q_\lambda(x) = \frac{e^{g_\lambda(x)}}{Z_\lambda} \tag{2}$$

where

$$g_\lambda(x) = \sum_j \lambda_j f_j(x).$$

Note that Eq. (1) is a convex function of $\boldsymbol{\lambda}$.

3 Algorithm

We now introduce an algorithm for minimizing Eq. (1) and prove the algorithm's convergence. Before doing so, however, let's consider how we might normally minimize a function. We could use basic calculus, finding the equation's derivatives, and setting the derivatives to 0. Unfortunately, for Eq. (1) there is no nice analytic solution when doing so. Instead, we will numerically minimize the function. The plan for the algorithm will be:

1. Start with a guess for the λ_j's
2. Iteratively adjust the guess

The outline of the algorithm is as follows:

choose $\boldsymbol{\lambda}_1$ arbitrarily
for $t = 1, 2, \cdots$ until convergence
    compute $\boldsymbol{\lambda}_{t+1}$ from $\boldsymbol{\lambda}_t$

We now will specify how to compute $\boldsymbol{\lambda}_{t+1}$ from $\boldsymbol{\lambda}_t$.

First, we'll declare some assumptions we need to make about the features. Note that the coefficients are arbitrary and, due to the normalization constant, we are free to scale or add constants to the features without altering the result. Therefore, we can scale all of the features so they are between 0 and 1.

Therefore, assume without loss of generality:

1. $$f_j : X \to [0, 1]$$
2. Now divide the features by $1/n$ so that:
$$\sum_j f_j(x) \leq 1$$
3. In fact, we can go further. Create a new feature f_0 such that
$$f_0 = 1 - \sum_j f_j$$
Doing so does not change the representation of the problem because the representation is still just a linear combination of the features. Adding this feature allows us to assume:
$$\sum_j f_j(x) = 1$$

Now we return to the question of how to compute $\boldsymbol{\lambda}_{t+1}$ from $\boldsymbol{\lambda}_t$.

We will use the following trick. First, we alter the notation slightly for convenience:

$$\boldsymbol{\lambda} = \boldsymbol{\lambda}_t$$

$$\boldsymbol{\lambda}' = \boldsymbol{\lambda}_{t+1}$$

We restate that we want to minimize $L(\boldsymbol{\lambda})$. Let

$$\Delta L = L(\boldsymbol{\lambda}') - L(\boldsymbol{\lambda}).$$

Therefore, we would like ΔL to be minimized.
The trick we will use will be to put an upper-bound on ΔL as an approximation and then minimize that approximation. We will show that by doing so on every step of the algorithm, our algorithm will converge to the actual minimum.

4 Derivation of Algorithm

First note that $\lambda'_j = \lambda_j$ plus some small adjustment, which we can formalize as:

$$\lambda'_j = \lambda_j \ + \alpha_j. \tag{3}$$

What is α_j? To answer that, let's try to compute ΔL and, in doing so, see at what points we need to make an approximation. Recall that

$$L(\boldsymbol{\lambda}) = \ln\left(\frac{e^{g_\lambda(x_i)}}{Z_\lambda}\right)$$

Plugging in for λ and λ' we have:

$$\Delta L = \frac{1}{m}\sum_i \left[\ln\left(\frac{e^{g_\lambda(x_i)}}{Z_\lambda}\right) - \ln\left(\frac{e^{g_{\lambda'}(x_i)}}{Z_{\lambda'}}\right)\right]$$

Note that we have logs of exponentials and we have $\ln Z_\lambda$ and $\ln Z_{\lambda'}$, both constants that do not depend on i. Therefore, we can simplify to:

$$\frac{1}{m}\sum_i (g_\lambda(x_i) - g_{\lambda'}(x_i)) + \ln\left(\frac{Z_{\lambda'}}{Z_\lambda}\right) \tag{4}$$

Remember that

$$g_\lambda(x) = \sum_j \lambda_j f_j(x). \tag{5}$$

Therefore, $(g_\lambda(x_i) - g_{\lambda'}(x_i))$ can be re-written as:

$$\sum_j (\lambda_j f_j(x_i) - \lambda'_j f_j(x_i)).$$

Recall from Eq. (3) that $\alpha_j = \lambda'_j - \lambda_j$.
Therefore,

$$(g_\lambda(x_i) - g_{\lambda'}(x_i)) = -\sum_j \alpha_j f_j(x_i).$$

Plugging this equation in above and noting that we can reverse the sum over i, j, the entire first term of Eq. (4) becomes:

$$-\frac{1}{m}\sum_{i,j}\alpha_j f_j(x_i)$$

and we can pull out the α_j, which is not dependent on i, to yield

$$= -\sum_j \alpha_j \left(\frac{1}{m}\sum_i f_j(x_i)\right).$$

Recall from the previous lecture that

$$\hat{E}[f_j] = \frac{1}{m}\sum_i f_j(x_i).$$

Therefore, we can re-write the first term as:

$$-\sum_j \alpha_j \hat{E}[f_j].$$

Now let's examine the second term. We will ignore the logs for now. Note that Z_λ is a normalizing constant so

$$\frac{Z_{\lambda'}}{Z_\lambda} = \frac{\sum_{x\in X}\exp\left(\sum_j \lambda'_j f_j(x)\right)}{Z_\lambda}.$$

Replacing λ'_j using Eq. (3):

$$= \frac{\sum_x \exp\left(\sum_j \lambda_j f_j(x) + \alpha_j f_j(x)\right)}{Z_\lambda}$$

Substituting in Eq. (5), we can re-write this as:

$$= \sum_x \frac{e^{g_\lambda(x)}}{Z_\lambda}\exp\left(\sum_j \alpha_j f_j(x)\right)$$

Recall from Eq. (2) that

$$q_\lambda(x) = \frac{e^{g_\lambda(x)}}{Z_\lambda}$$

Therefore, substituing above, we have

$$\frac{Z_{\lambda'}}{Z_\lambda} = \sum_x q_\lambda(x)\exp\left(\sum_j \alpha_j f_j(x)\right)$$

However we still have the α's we're trying to optimize embedded in a "nasty" exponential. To get around this problem, recall the approach we've seen many times before: upper bound an exponential by a linear. Also, recall our third assumption, that $\sum_j f_j(x) = 1$. Therefore, above, we simply have an exponential raised to the average of the α_js.
Recall from previous lectures that an exponential is convex and we have the following upper-bound:

$$f(avg) \leq avg(f)$$

We can use this convexity to upper-bound the above equation and give us an approximate optimization as follows. We will upper-bound $\exp(\sum_j \alpha_j f_j(x))$ with $\sum_j f_j(x)\exp(\alpha_j)$, leading to the following upper-bound:

$$\frac{Z_{\lambda'}}{Z_\lambda} \leq \sum_x q_\lambda(x) \sum_j f_j(x) e^{\alpha_j}$$

Again reversing the sum,

$$= \sum_j e^{\alpha_j} \sum_x q_\lambda(x) f_j(x)$$

Note, though, that $\sum_x q_\lambda(x) f_j(x) = E_{q_\lambda}[f_j]$.

Putting the whole thing together, we have shown that:

$$\Delta L \leq -\sum_j \alpha_j \hat{E}[f_j] + \ln\left(\frac{Z_{\lambda'}}{Z_\lambda}\right)$$

and that:

$$\frac{Z_{\lambda'}}{Z_\lambda} \leq \sum_j e^{\alpha_j} E_{q_\lambda}[f_j]$$

Therefore:

$$\Delta L \leq -\sum_j \alpha_j \hat{E}[f_j] + \ln\left(\sum_j e^{\alpha_j} E_{q_\lambda}[f_j]\right). \tag{6}$$

Eq. (6) *can* be optimized simply by differentiating and setting the derivative equal to 0, which we will now do. First, let:

$$\hat{E}[f_j] = \hat{E}_j$$

$$E_{q_\lambda}[f_j] = E_j$$

Taking the derivative with respect to α_j yields:

$$\frac{\partial}{\partial_{\alpha_j}} = -\hat{E}_j + \frac{E_j e^{\alpha_j}}{\sum_j E_j e^{\alpha_j}}. \tag{7}$$

We note that setting Eq. (7) equal to 0 would be much easier if not for the denominator in the second term. Conveniently, we can remove the denominator by noting that if we find a solution for the α_js and add a constant, the result would also be a solution. That is, let α'_j be a solution to the above (found by setting Eq. (7) to 0). Then $\alpha_j = \alpha'_j + C$ is also a solution, for any constant C. We can choose C so that $\sum_j E_j e^{\alpha_j} = 1$. Thus, setting Eq. (7) $= 0$ and solving, we obtain:

$$\alpha_j = \ln\left(\frac{\hat{E}_j}{E_j}\right)$$

and we have arrived at a method for approximately minimizing ΔL.

5 Iterative Scaling Algorithm

Now we plug the method arrived at above into our previous algorithm:

choose $\boldsymbol{\lambda}_1$
for $t = 1, 2, \cdots$ until convergence
    //compute $\boldsymbol{\lambda}_{t+1}$ from $\boldsymbol{\lambda}_t$
    $\lambda_{t+1,j} = \lambda_{t,j} + \ln\left(\frac{\hat{E}[f_j]}{E_{q_{\lambda_t}}[f_j]}\right)$

This algorithm is known as the **Iterative Scaling Algorithm**.

By plugging in previous equations we derived, it can be shown that this algorithm can also be thought of in the following alternative way, in terms of a distribution over samples instead of α's.
Let $p_t = q_{\lambda_t}$.

$$p_{t+1}(x) \propto p_t(x) \prod_j \left(\frac{\hat{E}[f_j]}{E_{p_t}[f_j]}\right)^{f_j(x)} \tag{8}$$

This alternative formulation is consistent with intuition. We are trying to reach a point at which $E_{p_t}[f_j] = \hat{E}[f_j]$. If $\hat{E}[f_j] > E_{p_t}[f_j]$, we have underestimated the expected value of feature j over the samples and would therefore like to increase the distribution weight of those samples with high values for j more than for those with low values. For feature j, the quotient within the product in Eq. (8) will be > 1, so the weight adjustment component corresponding to j will be proportional to the value of f_j for each sample. Alternatively, if $E_{p_t}[f_j] < \hat{E}[f_j]$, the weight adjustment component corresponding to j will be inversely proportional to the value of f_j for each sample, contributing a negative adjustment amount for examples with larger f_j values. In addition, the magnitude of the adjustment is also dependent on how different the empirical and true expected values are.

6 Proof of Convergence

Now that we have the Iterative Scaling Algorithm, how do we prove it works, i.e. that it converges? After all, we made an approximation when we upper-bounded ΔL. Thus we need to apply a general technique for proving the convergence of algorithms that perform such approximations.

6.1 Theorem

The distributions p_t converge to the Maximum Entropy/Maximum Likelihood solution $q\star$. Formally,

$$p_t \to q\star$$

6.2 Proof

First, we will note that to prove L is converging to a minimum, it is not sufficient to show that L is strictly decreasing. Now, for the proof, we will use the **Method of Auxiliary Functions**.

We define an *auxiliary function* A here as a *helper function* that maps probability distributions $\rightarrow \Re$ and has 3 properties:

1. A is continuous.
2. A upper-bounds the change in the negative log likelihood function and is never positive, i.e.
$$\Delta L = L(\boldsymbol{\lambda}_{t+1}) - L(\boldsymbol{\lambda}_t) \leq A(q_{\lambda_t}) \leq 0$$
3. $A(p) = 0 \Rightarrow p \in P$ (if it is 0, it implies p is in P)

It turns out that showing that there exists an A as defined above implies the Iterative Scaling Algorithm converges to $q\star$. Why?
Say A exists. First, we know that $L \geq 0$ and $L(\boldsymbol{\lambda}_t)$ is always decreasing. Together, these imply
$$L(\boldsymbol{\lambda}_{t+1}) - L(\boldsymbol{\lambda}_t) \rightarrow 0$$
In addition, note by property (2) above that $A(p_t)$ is squeezed between $L(\boldsymbol{\lambda}_{t+1}) - L(\boldsymbol{\lambda}_t)$ and 0, where $p_t = q_{\lambda_t}$. Since the difference between them is $\rightarrow 0$, this implies
$$A(p_t) \rightarrow 0$$
Suppose that p_t converges with limit p, i.e. $p_t \rightarrow p$. Why does this imply that p is optimal? Given what we just showed and that A is continuous by property (1),
$$A(p) = \lim_{t \rightarrow \infty} A(p_t) = 0$$
which implies that $p \in P$ by property (3). Also, since each $p_t \in Q$ is a Gibbs distribution, and since p is the limit of points in Q, we have that
$$p \in \overline{Q}$$
and therefore
$$p \in P, p \in \overline{Q} \Rightarrow p \in P \cap \overline{Q} \Rightarrow p = q\star$$
by the Duality Theorem.

This argument assumes that the p_t's have a limit, a fact which we need a little bit of analysis or topology to prove. Although we skipped over this in class, for those who are interested, here is how this can be proved. Suppose the sequence of p_t's does not converge to $q^\star$. Then there must exist a neighborhood R around $q^\star$ such that an infinite number of p_t's lie outside of R. The p_t's lie in the space of all probability distributions over the finite set X. This is a compact space. Therefore, the infinite subset of p_t's outside of R must have a subsequence which converges to some point p (this is a property of compactness). By the same argument given above (slightly modified), p must be equal to $q^\star$, a contradiction since all of the points are outside of the neighborhood R around $q^\star$. Therefore, the p_t's converge to $q^\star$.

Now, how do we find A? Our A should be an upper-bound on ΔL. We already derived an upper-bound on ΔL and derived a choice for $\alpha_j = \ln\left(\frac{\hat{E}_j}{E_j}\right)$, so we can plug in α_j and we'll have an upper-bound on the change in L.
$$\Delta L \leq -\sum_j \hat{E}_j \ln\left(\frac{\hat{E}_j}{E_j}\right) + \ln\left(\sum_j \hat{E}_j\right).$$

First, note that

$$\sum_j \hat{E}_j = \sum_j \hat{E}[f_j].$$

By linearity of expectations,

$$= \hat{E}\left[\sum_j f_j\right]$$

and we said

$$\sum_j f_j = 1$$

so

$$\sum_j \hat{E}_j = 1$$

and the second term above goes away altogether. Therefore, we have:

$$\Delta L \leq -\sum_j \hat{E}_j \ln\left(\frac{\hat{E}_j}{E_j}\right)$$

By a similar argument to the one above, $\sum_j E_j = 1$. Thus, the E_j's and $\hat{E}_j$'s are distributions over features so we just have Relative Entropy! That is,

$$-\sum_j \hat{E}_j \ln\left(\frac{\hat{E}_j}{E_j}\right) = -RE\left(\hat{E}_j || E_j\right)$$

Therefore, we can use negative Relative Entropy as our A. So define

$$A(p) = -RE\left(\hat{E}\left[f_j\right] || E_p\left[f_j\right]\right)$$

Now we check the properties we specified above.

- Is A continuous? Yes, RE is continuous.
- $\Delta L \leq A(q_{\lambda_t})$? Yes, we just proved that.
- $A(p_t) \leq 0$? Yes, $RE \geq 0$, so $-RE \leq 0$.
- $A = 0 \Rightarrow p \in P$? Yes, $RE = 0 \Rightarrow \hat{E}\left[f_j\right] = E_p\left[f_j\right]$ is a property of RE.

Q.E.D.

7 General comments

1. The above proof was written for when we have a distribution over examples; however it is common to have labeled data x, y and the goal of estimating $Pr[y|x]$. It turns out that one can apply similar ideas in this case, essentially trying to maximize the entropy of $Y|X$ given constraints derived from data. This problem is called **Logistic Regression**, an "oldie but goodie." Therefore, Logistic Regression is just a special case of Maximum Entropy.

2. If the true probability distribution is in the class of distributions you are searching over, Maximum Likelihood will eventually converge to the true probability; however, Maximum Likelihood can behave very badly if the true distribution is not in that class. For example, consider the following scenario:

   - a distribution over $0, 1 = \left\{ \begin{array}{ll} 1 & \text{with probability } p \\ 0 & \text{with probability } 1-p \end{array} \right\}$ i.e. a simple Bernoulli distribution, for instance the probability of a coin flip returning heads.
   - say the model class you are considering is, for some reason, $= \{0.01, 1\}$, e.g. heads with probability 0.01 or 1 but no other possibilities.
   - say the true distribution $p = 0.98$.
   - intuitively, 1 is the better estimated distribution, but Maximum Likelihood will return 0.01 as the estimated distribution as the number of examples becomes larger. Why? Let's consider the expected log loss.

     $$-E[\ln q] = -0.98 \ln q - 0.02 \ln(1-q).$$

     Since $0.02 \ln(1-1) = 0.02 \ln 0$, the second term becomes ∞. Thus:

     $$= \left\{ \begin{array}{ll} < \infty & \text{if } q = 0.01 \\ \infty & \text{if } q = 1 \end{array} \right\}$$

     So Maximum Likelihood will naturally prefer the first alternative.

   Here we see an important caveat: Maximum Likelihood behaves badly when dealing with probabilities very close to 0 or 1.

8 Introduction to Online Log Loss and Universal Compression

We'll now introduce our next topic. Previously, we talked about the connection between Maximum Likelihood and minimizing log loss and how minimizing log loss is $\approx$ RE; however, what if you would like to do log loss in online learning? For example, imagine you are betting on horses at the track. You want to estimate the probability of each horse winning and translate these estimations into bets corresponding to the probability distribution over horses. How do you estimate these probabilities? You may want to use the expert advice setting, in which the experts are estimating the probabilities of each horse winning each race. Before each race, you would combine the probability estimates of the experts into a single distribution. One horse will win the race and then you'll move on and repeat the expert advice pooling for the next race. As in our bit expert setting, you'll want to learn which expert is giving the best predictions and, as before, you will want your predictions to be not much worse than the best expert, i.e. the one giving the best predictions in hindsight, without making probabilitistic assumptions; however the predictions are now probability distributions, not single bits.

Let's formalize the problem.

for $t = 1, 2, \cdots, T$
  each expert i chooses a distribution $p_{t,i}$,
    corresponding to expert i's prediction at time t,
    over the space X of possible outcomes
  the master/learner combines the $p_{t,i}$'s into a distribution q_t
  observe $x_t \in X$
  suffer a loss $-\ln q_t(x_t)$

The goal is, naturally, to suffer a low cumulative loss, i.e. a loss that is worse than the loss of the best expert by only a small amount.

$$-\sum_t \ln q_t(x_t) \leq \min_i \left[-\sum_t \ln p_{t,i}(x_t) \right] + \text{a small amount}$$

It turns out, as we will see next time, that solving this problem is closely related to performing compression. Stay tuned!