CS 446 Machine Learning Fall 2016 Nov 10, 2016

Semi-Supervised Learning

Professor: Dan Roth *Scribe: Ben Zhou, C. Cervantes*

Overview

- Semi-Supervised Learning
- Expectation Maximization
- K-Means Algorithm

1 Semi-Supervised Learning

Consider the problem of prepositional phrase attachment; we want to predict to which preceding phrase a prepositional phrase attaches, as in

Buy car *with money*.

Buy car *with wheels*.

In this setting, each sentence has four features (one for each word) and our goal is to predict whether to attach the preposition ("with") to the verb ("buy") or the noun ("car"). Let us further expand the feature space to include all conjunctions, resulting in 15 total features.

In the Naive Bayes setting, we want to predict **n**oun or **v**erb, given our features ($[x_1...x_{15}]$). Therefore, we must estimate

$$\begin{array}{ll} P(n) & P(v) \\ P(x_1|n) & P(x_1|v) \\ & ... \\ P(x_n|n) & P(x_n|v) \end{array}$$

Note that there are two values for each cell in the column (ex. $P(x_1|n=1)$ and $P(x_1|n=0)$).

Now, given an example $(x_1, x_2...x_n, ?)$, we compare

$P(n|x) = P(n)P(x_1|n)...P(x_n|n)$

$P(v|x) = P(v)P(x_1|v)...P(x_n|v)$.

and predict the label with the highest probability.

Consider that after seeing 10 examples, we have

$$
\begin{array}{ll}
P(n) = 0.5 & P(v) = 0.5 \\
P(x_1|n) = 0.75 & P(x_1|v) = 0.25 \\
P(x_2|n) = 0.5 & P(x_2|v) = 0.25 \\
P(x_3|n) = 0.5 & P(x_3|v) = 0.75 \\
P(x_4|n) = 0.5 & P(x_n|v) = 0.5
\end{array}
$$

If now we have an example of $x = [1000]$, we have

$P(n|x) = 0.5 * 0.75 * 0.5 * 0.5 * 0.5 = \frac{3}{64}$

$P(v|x) = 0.5 * 0.25 * 0.75 * 0.25 * 0.5 = \frac{3}{256}$

According to Naive Bayes, we then compare these two numbers and predict that the label for this example is n.

1.1 Incorporating Unlabeled Data

Consider that – in addition to the 10 labeled examples, above – we are given 100 unlabeled examples. Intuitively, we might expect that these new examples might help us train our model, even without labels, and we therefore must consider how best to use these unlabeled examples.

In this setting, our goal is to assign labels to the unlabeled examples and retrain our model, repeating the process until some convergence criteria. When considering methods to assign labels, we have several options.

Option 1
Make predictions on (some of) the unlabeled data, using a probability threshold to determine how many examples to use (ex. retain only those labelings with probability $> 80\%$). Ideally, the examples we keep (and then retrain on) are the highest confidence ones.

Option 2
Assume we have some similarity function over the data. We can further assume that similar examples have similar labels, which enables us to propagate labels; each unlabeled example that is sufficiently similar to a labeled example is assigned that label.

Option 3
Importantly, we can relax our predictions, saying instead that x has label n with probability

$$\frac{P_n(x)}{P_n(x) + P_v(x)}$$

and has label v with probability

$$\frac{P_v(x)}{P_n(x) + P_v(x)}$$

We can then train Naive Bayes as before, because *Naive Bayes does not require integers*. During training, probabilities can be estimated using fractional label counts.

2 Algorithms

In the previous section, Option 1 and Option 3 are quite similar, and suggest the following algorithms:

> Use a threshold, chose examples labeled with high confidence; label them and retrain
>
> Label the examples with fractional labels and then retrain

Both can be used iteratively and both can be used with other classifiers (there is no restriction to Naive Bayes). The only requirement is we must have a robust confidence measure in the classification.

These two approaches correspond to *Hard EM* and *Soft EM*, respectively; in the former, we make a hard labeling decision at each step – we commit to a label – but in the latter we only commit to a distribution.

There are other approaches to semi-supervised learning as well; co-training, bootstrapping, graph-based algorithms that invent some notion of similarity and propagate labels.

3 Expectation Maximization (Coins)

Expectation Maximization (EM) is a class of algorithms used to estimate probability distributions in the presence of missing attributes.

In order to run the algorithm, we must make an assumption about the underlying probability distribution. In the Naive Bayes example, above, we assumed that the two labels were binomial.

EM is sensitive to this initial assumption. Typically it is run multiple times under different initial conditions to account for this.

3.1 Three Coin Example

Consider we have three coins, such that the probability of a **h**ead is given as

> Coin 0: $P(h) = \alpha$
>
> Coin 1: $P(h) = p$

Coin 2: $P(h) = q$

Consider the following scenarios

Scenario 1
Toss one of the coins four times, observing: HHTH.

Question: Which coin is more more likely to produce this sequence?

Solution: Trivial; choose coin 0, 1, or 2 based on whether α, p, or q is closest to 0.75, respectively.

Scenario 2
Toss coin 0. If heads, toss coin 1; if tails, toss coin 2. We observe

HHHHT

THTHT

HHHHT

HHTTH

where the first toss in each sequence (bold) corresponds to coin 0, and the others correspond to coin 1 or 2.

Question: What are the most likely values for α, p, and q?

Solution: First calculate α (0.75), and then estimating p and q is trivial.

Scenario 3
Toss coin 0. If heads, toss coin 1; if tails, toss coin 2. We observe

?HHHT

?HTHT

?HHHT

?HTTH

Question: What are the most likely values for α, p, and q?

Solution: Since we no longer have the label for coin 0, there exists no known analytic solution (in the general case).

3.2 Intuition

In Scenario 3 from the previous section, if we knew which example came from which coin (as in Scenario 2), we could simply find the maximum likelihood estimates.

Since we lack such labels, however, we must use the following iterative approach to estimate the parameters.

1. Guess the probability that a given example came from coin 1 or 2, and generate fictional labels weighted according to that probability
2. Estimate the most likely value of the parameters
3. Compute the likelihood of the data, given these parameters
4. Re-estimate the parameters, maximizing the likelihood of the data

The core intuition here is that if we have parameters we can predict labels, and if we have labels we can estimate parameters. This iterative process can be shown to converge to a local maximum of the likelihood function.

3.3 Algorithm

Assume that we know the parameters $\tilde{p}, \tilde{q}, \tilde{\alpha}$ (we guess them to start). Let's further assume that we have n data points, in each there are m tosses and h heads.

The probability that the i^{th} data point come from coin 1 is given by

$$\begin{aligned} P_1^i &= P(coin\ 1|D^i) \\ &= \frac{P(D^i|coin\ 1)P(coin\ 1)}{P(D^j)} \\ &= \frac{\tilde{\alpha}\tilde{p}^h(1-\tilde{p})^{m-h}}{\tilde{\alpha}\tilde{p}^h(1-\tilde{p})^{m-h} + (1-\tilde{\alpha})\tilde{q}^h(1-\tilde{q})^{m-h}} \end{aligned} \tag{1}$$

This process – determining P_1^i – is known as the *Expectation Step*.

We must now compute the log likelihood of the data, and in so doing find parameters to maximize it.

$$LL = \sum_1^n \log P(D^i|p, q, \alpha)$$

However, we do not observe all the data. The label of the coin is hidden, and we thus must marginalize over the possible coins (labels). This results in

$$LL = \sum_{i=1}^n \log \sum_{y=0,1} P(D^i, y|p, q, \alpha)$$

where y is the label we don't see.

We cannot maximize this directly, however, because there is a summation inside the log. In EM, we circumvent this limitation by using the expectation of the log likelihood under the posterior distribution of the the latent variable y.

In this setting, we consider the expectation[1] under y to be given by

$$E_y[P(D^i|p,q,\alpha)] = \sum_{y=0,1} P(D^i|p,q,\alpha)P(y|D^i,p,q,\alpha)$$

In effect, we are considering the probability of the data D – given parameters p, q, and α – to be a random variable that depends on the value y of the coin on the i^{th} toss. This enables us maximize the expectation of this random variable, rather than maximizing the log likelihood directly.

To see why this is the case, consider Jensen's Inequality

$$\begin{aligned} LL &= \sum_{i=1}^{n} \log \sum_{y=0,1} P(D^i, y|p,q,\alpha) \\ &= \sum_{i=1}^{n} \log \sum_{y=0,1} P(D^i|p,q,\alpha)P(y|D^i,p,q,\alpha) \\ &= \sum_{i=1}^{n} \log E_y P(D^i|p,q,\alpha) \\ &\geq \sum_{i=1}^{n} E_y \log P(D^i|p,q,\alpha) \end{aligned} \tag{2}$$

The last line is of particular note, stating that the sum of the log of the expectation of the random variable provides an upper bound for the sum of the expectation of the log values of that random variable. Put another way, given random variable X, the inequality enables us to move the summation (implicit in computing expectation) outside of the log, since $\sum log(E[X]) \geq \sum E[log(X)]$.

Now we maximize an upper bound, rather than maximizing the true value, over

[1] Recall that the *expected value* of a random variable is the average value of that variable after repeated experiments

the coin's label, y.

$$\begin{aligned}
E[LL] &= E\left[\sum_{i=1}^{n} \log P(D^i|p,q,\alpha)\right] \\
&= \sum_{i=1}^{n} E\left[\log P(D^i|p,q,\alpha)\right] \\
&= \sum_{i=1}^{n} P_1^i \log P(D^i,1|p,q,\alpha) + (1-P_1^i)\log P(D^i,0|p,q,\alpha) \\
&= \sum_{i=1}^{n} P_1^i \log(\tilde{\alpha}\tilde{p}^{h_i}(1-\tilde{p})^{m-h_i}) + (1-P_1^i)\log((1-\tilde{\alpha})\tilde{q}^{h_i}(1-\tilde{q})^{m-h_i}) \\
&= \sum_{i=1}^{n} P_1^i(\log\tilde{\alpha} + h_i\log\tilde{p} + (m-h_1)\log(1-\tilde{p})) + \\
&\qquad (1-P_1^i)(\log(1-\tilde{\alpha}) + h_i\log\tilde{q} + (m-h_1)\log(1-\tilde{q}))
\end{aligned} \tag{3}$$

Finally, we can derive the most likely parameters by maximizing the derivatives with respect to $\tilde{p}, \tilde{q}, \tilde{\alpha}$.

$$\begin{aligned}
\frac{\partial E}{\partial\tilde{\alpha}} &= \sum_{i=1}^{n}\left(\frac{P_1^i}{\tilde{\alpha}} - \frac{1-P_1^i}{1-\tilde{\alpha}}\right) \\
\tilde{\alpha} &= \frac{1}{n}\sum_i P_1^i \\
\frac{\partial E}{\partial\tilde{p}} &= \sum_{i=1}^{n} P_1^i(\frac{h_i}{\tilde{p}} - \frac{m-h_i}{1-\tilde{p}}) \\
\tilde{p} &= \frac{\sum P_1^i\frac{h_i}{m}}{\sum P_1^i} \\
\frac{\partial E}{\partial\tilde{q}} &= \sum_{i=1}^{n}(1-P_1^i)(\frac{h_i}{\tilde{q}} - \frac{m-h_i}{1-\tilde{q}}) \\
\tilde{q} &= \frac{\sum(1-P_1^i)\frac{h_i}{m}}{\sum(1-P_1^i)}
\end{aligned} \tag{4}$$

Essentially, these values correspond to

α: average of the probabilities that the i^{th} example came from α

p: the number of heads in the i^{th} example, weighted by the probability that these came from coin 1

q: same as p, or the weighted sum of the observations

as before, we are fixing P_1^i as a constant in order to differentiate.

This procedure can be referred to as the *maximization step*.

4 EM Algorithm (General)

Assume two sets – $\mathcal{X}$ and $\mathcal{Y}$ – and a joint distribution $P(X, Y|\theta)$.

If we had fully observed data – (X_i, Y_i) pairs – then we can find the likelihood of the parameters according to

$$L(\theta) = \sum_i \log P(X_i, Y_i|\theta)$$

In the EM setting, however, we assume only partially observed data – X_i without the corresponding Y_i – as in the above example, where we saw the coin tosses without knowing which coin produced them.

Under this framework, we define the likelihood of our parameters θ as

$$L(\theta) = \sum_i \log P(X_i|\theta) = \sum_i \log \sum_{Y \in \mathcal{Y}} P(X_i, Y|\theta)$$

In EM, we find the best parameters by finding

$$\theta_{ML} = \underset{\theta}{\operatorname{argmax}} \sum_i \log \sum_{Y \in \mathcal{Y}} P(X_i, Y|\theta)$$

EM is a general purpose algorithm for finding the maximum likelihood estimates in latent variable models, iterating over expectation (E-step) and maximization (M-step). In the E-step, we fill in the latent variables using the posterior, and in the M-step we maximize the expected complete log likelihood with respect to the complete posterior distribution.

4.1 Setup

Let $D = (x_1, x_2 ... x_n)$ be the observed data, let z denote the hidden (latent) random variable[2], and let θ be the model parameters. Then

$$\begin{aligned} \theta^* &= \underset{\theta}{\operatorname{argmax}} P(x|\theta) \\ &= \underset{\theta}{\operatorname{argmax}} \sum_z P(x, z|\theta) \\ &= \underset{\theta}{\operatorname{argmax}} \sum_z (P(z|\theta) P(x|z, \theta)) \end{aligned} \tag{5}$$

We refer to this expression as the complete log likelihood.

[2]Random variable z could be a vector – that is, a set of latent variables – rather than a single value; EM is general enough to handle this setting as well

To derive the EM objective function, we re-write the complete log likelihood function by multiplying it by $\frac{q(z)}{q(z)}$ where $q(z)$ is an arbitrary distribution for the random variable z.

$$\begin{aligned}\log p(x|\theta) &= \log \sum_z p(x, z|\theta) \\ &= \log \sum_z p(z|\theta)p(x|z,\theta)q(z)/q(z) \\ &= \log E_q[p(z|\theta)p(x|z,\theta)/q(z)] \\ &\geq E_q \log[p(z|\theta)p(x|z,\theta)/q(z)]\end{aligned} \tag{6}$$

Recall that the last line is given by Jensen's Inequality, where we know – because log is a concave function – that the log of a function will always be greater than or equal to the average (expectation) of that function.

Now we get the objective:

$$L(\theta, q) = E_q[\log p(z|\theta)] + E_q[\log p(x|z,\theta)] - E_q[\log q(z)]$$

where the last component of the objective is an entropy component. It is also possible to write the objective function so that it includes KL-divergence (a distance function between distributions) between $q(z)$ and $p(z|x,\theta)$.

4.2 Procedure

We can now consider EM to be an iterative, gradient ascent algorithm, where q is the posterior function

$$q = p(z|x,\theta)$$

Thus, at time t (the t^{th} step), we have q^t and θ^t.

E-Step
Update posterior, q, while fixing θ^t

$$q^{t+1} = \underset{q}{\operatorname{argmax}}\, L(q, \theta^t) = p(z|x, \theta^{(t)})$$

M-Step
Update parameters to maximize the expected compete log-likelihood

$$\theta^{t+1} = \underset{\theta}{\operatorname{argmax}}\, L(q^{t+1}, \theta)$$

With the right q, we have:

$$\begin{aligned} L(\theta, q) &= E_q \log[p(z|\theta)p(x|z,\theta)/q(z)] \\ &= \sum_z p(z|x,\theta) \log[p(x,z|\theta/p(z|x,\theta)] \\ &= \sum_z p(z|x,\theta) \log[p(x,z|\theta)p(x|\theta)/p(z,x|\theta)] \\ &= \sum_z p(z|x,\theta) \log[p(x|\theta)] \\ &= \log[p(x|\theta)] \sum_z p(z|x,\theta) \\ &= \log[p(x|\theta)] \end{aligned} \tag{7}$$

The equation gives an intuition that the algorithm is doing the right thing because it maximizes the likelihood of the data given the parameters.

Other qs can be chosen to produce variations of EM. For example, rather than keeping the distribution, we could keep only a label instead. Doing so results in Hard EM, rather than the above (Soft EM).

In general, the EM procedure is given by

1. Initialize θ as θ_0
2. E-Step: use the current θ^t to find the value of q^{t+1} (the posterior, in soft EM)
3. M-Step: use the current q^t to find the maximum parameters θ^{t+1}
4. Repeat steps 2 and 3 until convergence

5 K-Means

Consider data points that are known to be sampled independently from a mixture of k normal distributions (clusters). K-means is a clustering algorithm for partitioning these points into groups – or clusters – corresponding to the true distributions.

Consider the points in Figure 1, where each point comes from a distribution, labeled as the mean of that distribution (μ_1 and μ_2 (we assume that each has the same standard deviation (σ).

In k-means, the general procedure is as follows

1. Start with a set of points, assuming that they come from k clusters
2. Choose k center points (clusters)

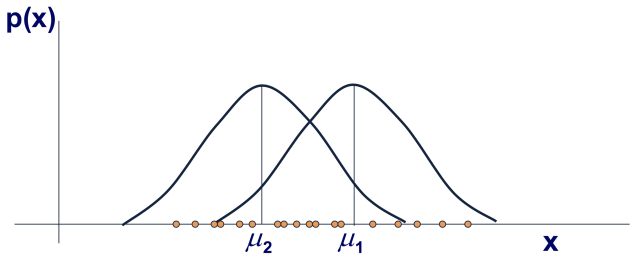

Figure 1: K-means example

3. Assign each of the points to one of the clusters
4. Recompute the center points (clusters) and repeat

This is a similar procedure to that of Hard-EM; that is, we choose a label for each example (make a hard assignment), recompute our parameters, and repeat iteratively. Therefore, we can view k-means as an EM algorithm, even to the point of relaxing the hard assignments.

5.1 Algorithm

First, if we know that all the data points are taken from a normal distribution with mean μ, finding its most likely value is easy.

$$p(x|\mu) = \frac{1}{\sqrt{2\pi\sigma^2}} exp[-\frac{1}{2\sigma^2}(x-\mu)^2]$$

Using the maximum likelihood, we know that the most likely value is the average.

With many data points $D = \{x_1, x_2 ... x_m\}$,

$$\ln(L(D|\mu)) = \ln(P(D|\mu)) = \sum_i -\frac{1}{2\sigma^2}(x_i - \mu)^2$$

Maximizing the log-likelihood is equivalent to minimizing

$$\mu_{ML} = argmin_\mu \sum_i (x_i - \mu)^2$$

By differentiating this, with respect to μ, we get that the minimal point. That is, the most likely mean is

$$\mu = \frac{1}{m} \sum_i x_i$$

or the average.

5.2 A mixture of distributions

As in the coins example, the problem is that data is sampled from a mixture of k different normal distributions, and we do not know, for a given data point x_i, from which distribution it was sampled.

Now the problem becomes: what is the probability that example x_i was sampled from distribution μ_j.

$$\begin{aligned} P_{ij} &= P(\mu_j|x_i) \\ &= \frac{P(x_i|\mu_j)P(\mu_j)}{P(x_i)} \\ &= \frac{\frac{1}{k}P(x=x_i|\mu=\mu_j)}{\sum_{n=1}^{k}\frac{1}{k}P(x=x_i|\mu=\mu_n} \\ &= \frac{exp[-\frac{1}{2\sigma^2}(x_i-\mu_j)^2]}{\sum_{n=1}^{k}exp[-\frac{1}{2\sigma^2}(x_i-\mu_n)^2]} \end{aligned} \tag{8}$$

Note that we guessed that there are k distributions (μ_j clusters).

Now we can introduce k binary hidden variables $z_{i1}, z_{i2}...z_{ik}$ such that $z_{ij} = 1$ *iff* x_i is sampled from the j^{th} distribution.

$E[z_{ij}] = 1*P(x_i \text{ was sampled from } \mu_j)+0*P(x_i \text{ was not sampled from } \mu_j) = P_{ij}$

Let $h = \sigma, \mu_1, ...\mu_k$, we have

$$p(y_i|h) = p(x_i, z_{i1}, ...z_{ik}|h) = \frac{1}{\sqrt{2\pi\sigma^2}}exp[-\frac{1}{\sqrt{2\sigma^2}}\sum_j z_{ij}(x_i-\mu_j)^2]$$

Note here that the introduction of z_{ij} simply enables us to compactly express the likelihood.

Given the above, we can now compute the expectation of the likelihood given $D = \{x_1, x_2...x_m\}$ and the hypothesis h,

$$\begin{aligned} E[\ln(P(Y|h))] &= E[\sum_{i=1}^{m} -\frac{1}{2\sigma^2}\sum_j z_{ij}(x_i-\mu_j)^2] \\ &= \sum_{i=1}^{m} -\frac{1}{2\sigma^2}\sum_j E[z_{ij}](x_i-\mu_j)^2 \end{aligned} \tag{9}$$

Then we want to maximize with respect to μ_j

$$Q(h|h\prime) = \sum_{i=1}^{m} -\frac{1}{2\sigma^2}\sum_j E[z_{ij}](x_i-\mu_j)^2$$

$$\frac{dQ}{d\mu_j} = C\sum_{i=1}^{m} E[z_{ij}](x_i - \mu_j) = 0$$

which yields

$$\mu_j = \frac{\sum_{i=1}^{m} E[z_{ij}]x_i}{\sum_{i=1}^{m} E[z_{ij}]}$$

6 Conclusion

EM is a family of algorithms that can be used as a way to estimate a mixture of probability distributions. We've shown two types of settings in which we can use EM to estimate the most likely density functions.

EM is very important because in many real-world cases we are missing variables or labeled data. Note, though, that it requires us to make assumptions about the distributions we're trying to learn. If you make good assumptions, EM will work quite well; if you don't, it wont.

Consider a dataset: $x_i \in \{0,1\}^{n+1}$

and a task: given $x_1 ... x_n$, predict x_0.

To learn this, we have two options

Parametric
Estimate the model using EM. Once a model is known, use it to make predictions. The limitation is that we cannot use EM directly without an additional assumption on the way data is generated (we must choose a family of probability distributions).

Non-Parametric
Learn x_0 directly as a function of the other variables. The limitation is that we do not know which function to try and learn (we must choose a hypothesis class).

Let's assume a variable z with k values $1 \leq z \leq k$ with probability $\alpha_z \sum_{1}^{k} \alpha_z = 1$.
Now that we know the value of z, let's call this value x_0. Now we choose values for each one of the other variables x_i according to probability P_i^z (0 otherwise); each x_i is independent from each other.

x_0 turns out to be a linear function of the other variables when $k = 2$. When k is known, the EM approach performs well. If an incorrect value is assumed, the estimation fails, then the linear methods performs better.