Machine Learning
in
Natural Language

Semi-Supervised Learning and the EM Algorithm

Dan Roth
University of Illinois, Urbana-Champaign
danr@cs.uiuc.edu
http://L2R.cs.uiuc.edu/~danr

Semi-Supervised Learning

- Consider the problem of Prepositional Phrase Attachment.

- There are several ways to generate features. Given the limited representation, we can assume that all possible conjunctions of the 4 attributes are used. (15 feature in each example).
- See other possibilities in [Krymolovsky, Roth 98]
- Assume we will use naïve Bayes for learning to decise between [n,v]
- Examples are: $(x_1, x_2, \ldots x_n, [n,v])$

Using naïve Bayes

- To use naïve Bayes, we need to use the data to estimate:

$P(n)$ $P(v)$

$P(x_1|n)$ $P(x_1|v)$

$P(x_2|n)$ $P(x_2|v)$

……

$P(x_n|n)$ $P(x_n|v)$

- Then, given an example $(x_1,x_2,\ldots x_n,?)$, compare:

$$P_n(x)=P(n)\ P(x_1|n)\ P(x_2|n)\ldots\ P(x_n|n)$$

and

$$P_v(x)=P(v)\ P(x_1|v)\ P(x_2|v)\ldots\ P(x_n|v)$$

Using naïve Bayes

- After seeing 10 examples, we have:
- P(n) =0.5; P(v)=0.5

$P(x_1|n)=0.75; P(x_2|n)=0.5; P(x_3|n)=0.5; P(x_4|n)=0.5$

$P(x_1|v)=0.25; P(x_2|v)=0.25; P(x_3|v)=0.75; P(x_4|v)=0.5$

- Then, given an example (1000), we have:

$P_n(x)=0.5\ 0.75\ 0.5\ 0.5\ 0.5 = 3/64$

$P_v(x)=0.5\ 0.25\ 0.75\ 0.25\ 0.5=3/256$

Now, assume that in addition to the 10 labeled examples, we also have 100 unlabeled examples.

Using naïve Bayes

- For example, what can be done with (1000?) ?
- We can guess the label of the unlabeled example…

- But, can we use it to improve the classifier? (that is, the estimation of the probabilities?)
- We can assume the example x=(1000) is a
  - n example with probability $P_n(x)/(P_n(x) + P_v(x))$
  - v example with probability $P_v(x)/(P_n(x) + P_v(x))$

- Estimation of probabilities does not require work with integers!

Using Unlabeled Data

The discussion suggests several algorithms:

1. Use a threshold. Chose examples labeled with high confidence. Labeled them [n,v]. Retrain.

2. Use fractional examples. Label the examples with fractional labels [p of n, (1-p) of v]. Retrain.

Comments on Unlabeled Data

- Both algorithms suggested can be used iteratively.
- Both algorithms can be used with other classifiers, not only naïve Bayes. The only requirement – a robust confidence measure in the classification.
- E.g.: Brill, ACL'01: uses all three algorithms in SNoW for studies of these sort.
- There are other approaches to Semi-Supervised learning: See included papers (co-training; Yarowksy's Decision List algorithm)
- What happens if instead of 10 labeled examples we start with 0 labeled examples?
- Make a Guess; continue as above; a version of EM

EM

- EM is a class of algorithms that is used to estimate a probability distribution in the presence of missing attributes.
- Using it, requires an assumption on the underlying probability distribution.
- The algorithm can be very sensitive to this assumption and to the starting point (that is, the initial guess of parameters.
- In general, known to converge to a local maximum of the maximum likelihood function.
- Some more details.

Three Coin Example

- We observe a series of coin tosses generated in the following way:
- A person has three coins.
  - Coin 0: probability of Head is λ
  - Coin 1: probability of Head p
  - Coin 2: probability of Head q

- Consider the following coin-tossing scenarios:

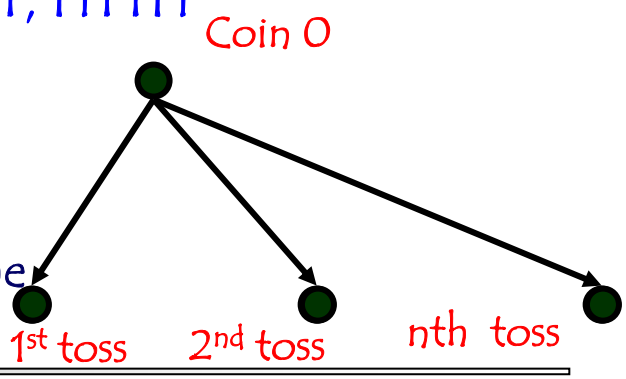
Estimation Problems

- Scenario I: Toss one of the coins six times.
  Observing HHHTHT
  Which coin is more likely to produce this sequence ?

- Scenario II: Toss coin 0. If Head – toss coin 1; o/w -- toss coin 2
  Observing the sequence HHHHT, THTHT, HHHHT, HHTTH
  produced by Coin 0 , Coin1 and Coin2
  Estimate most likely values for p, q (the probability of H in each coin) and the probability to use each of the coins (λ)

- Scenario III: Toss coin 0. If Head – toss coin 1; o/w -- toss coin 0
  Observing the sequence HHHT, HTHT, HHHT, HTTH
  produced by Coin 1 and/or Coin 2
  Estimate most likely values for p, q and λ

There is no known analytical solution to this problem.
That is, it is not known how to compute the values of the parameters so as to maximize the likelihood of the data.



Key Intuition

- If we knew which of the data points (HHHT), (HTHT), (HTTH) came from Coin1 and which from Coin2, there was no problem.
- Instead, use an iterative approach for estimating the parameters:
- Guess the probability that a given data point came from Coin 1/2
  Generate fictional labels, weighted according to this probability.
- Now, compute the most likely value of the parameters.
  [recall NB example]
- Compute the likelihood of the data given this model.
- Re-estimate the initial parameter setting: set them to maximize the likelihood of the data.

- This process can be iterated and can be shown to converge to a local maximum of the likelihood function

EM Algorithm (Coins) -I

- We will assume (for a minute) that we know the parameters $\tilde{p}, \tilde{q}, \tilde{\alpha}$ and use it to estimate which Coin it is (Problem 1)
- Then, we will use the estimation for the tossed Coin, to estimate the most likely parameters and so on…
- What is the probability that the ith data point came from Coin1 ?

$$P_1^i = P(\text{Coin1} \mid D^i) = \frac{P(D^i \mid \text{Coin1})\, P(\text{Coin1})}{P(D^i)} =$$

$$= \frac{\tilde{\alpha}\, \tilde{p}^h (1-\tilde{p})^{m-h}}{\tilde{\alpha}\, \tilde{p}^h (1-\tilde{p})^{m-h} + (1-\tilde{\alpha})\tilde{q}^h (1-\tilde{q})^{m-h}}$$

EM Algorithm (Coins) - II

- At this point we would like to compute the likelihood of the data, and find the parameters that maximize it.
- We will maximize the log likelihood of the data (n data points):

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

- But, one of the variables – the coin's name – is hidden.
- Which value do we plug in for it in order to compute the likelihood of the data?
- We marginalize over it. Or, equivalently, we think of the likelihood $\mathbf{logP(D^i \mid p, q, \alpha)}$ as a random variable that depends on the value of the coin in the i^{th} toss. Therefore, instead of maximizing the LL we will maximize the expectation of this random variable (over the coin's name).

EM Algorithm (Coins) - III

- We maximize the expectation of this random variable (over the coin name).

$$\mathbf{E[LL] = E[\sum_{i=1}^{n} \log P(D^i \mid p,q,\alpha)] = \sum_{i=1}^{n} E[\log P(D^i \mid p,q,\alpha)] =}$$

$$\mathbf{= \sum_{1}^{n} P_1^i \log P(1,D^i \mid p,q,\alpha) + (1-P_1^i) \log P(0,D^i \mid p,q,\alpha)}$$

- Due to the linearity of the expectation and the random variable:

$$\mathbf{\log P(D^i \mid p,q,\alpha) = \log P(1,D^i \mid p,q,\alpha) \quad w/p \quad P_1^i}$$

$$\mathbf{\log P(0,D^i \mid p,q,\alpha) \quad w/p \quad 1-P_1^i}$$

EM Algorithm (Coins) - IV

- Explicitly, we get:

$$E\left(\sum_i \log P(D^i \mid \tilde{p}, \tilde{q}, \tilde{\alpha})\right) =$$

$$= \sum_i P_1^i \log P(1, D^i \mid \tilde{p}, \tilde{q}, \tilde{\alpha}) + (1 - P_1^i) \log P(0, D^i \mid \tilde{p}, \tilde{q}, \tilde{\alpha}) =$$

$$= \sum_i P_1^i \log(\tilde{\alpha}\, \tilde{p}^{h_i} (1-\tilde{p})^{m-h_i}) + (1 - P_1^i) \log((1-\tilde{\alpha})\, \tilde{q}^{h_i} (1-\tilde{q})^{m-h_i}) =$$

$$= \sum_i P_1^i (\log \tilde{\alpha} + h_i \log \tilde{p} + (m - h_i) \log(1-\tilde{p})) +$$

$$(1 - P_1^i)(\log(1-\tilde{\alpha}) + h_i \log \tilde{q} + (m - h_i) \log(1-\tilde{q}))$$

EM Algorithm (Coins) - V

- Finally, to find the most likely parameters, we maximize the derivatives with respect to $\tilde{p}, \tilde{q}, \tilde{\alpha}$:

$$\frac{dE}{d\tilde{\alpha}} = \sum_{i=1}^{n} \frac{P_1^i}{\tilde{\alpha}} - \frac{1-P_1^i}{1-\tilde{\alpha}} = 0 \quad \Rightarrow \quad \tilde{\alpha} = \frac{\sum P_1^i}{n}$$

$$\frac{dE}{d\tilde{p}} = \sum_{i=1}^{n} P_1^i \left(\frac{h_i}{\tilde{p}} - \frac{m-h_i}{1-\tilde{p}}\right) = 0 \quad \Rightarrow \quad \tilde{p} = \frac{\sum P_1^i \frac{h_i}{m}}{\sum P_1^i}$$

$$\frac{dE}{d\tilde{q}} = \sum_{i=1}^{n} (1-P_1^i)\left(\frac{h_i}{\tilde{p}} - \frac{m-h_i}{1-\tilde{p}}\right) = 0 \quad \Rightarrow \quad \tilde{q} = \frac{\sum (1-P_1^i) \frac{h_i}{m}}{\sum (1-P_1^i)}$$

EM Summary (so far)

- EM is a general procedure for learning in the presence of unobserved variables.

- We have shown how to use it in order to estimate the most likely density function for a mixture of (Bernoulli) distributions.
- EM is an iterative algorithm that can be shown to converge to a local maximum of the likelihood function.

- It depends on assuming a family of probability distributions.
- In this sense, it is a family of algorithms. The update rules you will derive depend on the model assumed.
- It has been shown to be quite useful in practice, when the assumption made on the probability distribution are correct, but can fail otherwise.
- We now move to a general setting and the HMM case.