



Un/Semi-Supervised Learning: EM and K-Means

Dan Roth

danroth@seas.upenn.edu | <http://www.cis.upenn.edu/~danroth/> | 461C, 3401 Walnut

Slides were created by Dan Roth (for CIS519/419 at Penn or CS446 at UIUC), or by other authors who have made their ML slides available.

Administration (12/02/20)

Are we recording? YES!

Available on the web site

- Remember that all the lectures are available on the website **before the class**
 - Go over it and be prepared
 - A new set of written notes will accompany most lectures, with some more details, examples and, (when relevant) some code.
- **HW4 is out – NNs and Bayesian Learning**
 - Due 12/3 → 12/7 (Next Monday)
- **HW5 will be out tomorrow, 12/3. Due 12/10 (last day of the semester).**
 - It is mostly a summary of the material we covered this semester (with a focus on the second half) and will help you prepare for the exam.
 - No programming.
 - We will give an extension until a few days before the exam.
- Next week is the last week of classes – there will be **three** meetings (Monday, Wednesday, **Thursday**)
- **Projects**
 - You should have received an email from one of the TAs that will help guide your project
 - Progress reports are due today. Check the course's web site
- The **Final** is on 12/18; it is scheduled for 9am; we will try to give you a larger window of time.
- **Project Paper and Video Submission: 12/21**

Projects

- CIS 519 students need to do a team project: Read the [project descriptions](#) and follow the updates on the [Project webpage](#)
 - Teams will be of size 2-4
 - We will help grouping if needed
- There will be 3 options for projects.
 - Natural Language Processing (Text)
 - Computer Vision (Images)
 - Speech (Audio)
- In all cases, we will give you datasets and initial ideas
 - The problem will be multiclass classification problems
 - You will get annotated data only for some of the labels, but will also have to predict other labels
 - 0-zero shot learning; few-shot learning; transfer learning
- A detailed note will come out today.
- Timeline:
 - 11/11 Choose a project and team up
 - 11/23 Initial proposal describing what your team plans to do
 - 12/2 Progress report: 1 page. What you have done; plans; problems.
 - 12/21 Final paper + short video
- Try to make it interesting!

From NB to Logistic Regression

- Naïve Bayes and Logistic Regression:

– <https://colab.research.google.com/drive/1xWrXz0vs3PpTIBqWbJ9JYxAGui49sOra?usp=sharing#scrollTo=gG9pycPfUOrr>

Example

$$v_{NB} = \operatorname{argmax}_{v_j \in V} P(v_j) \prod_i P(x_i | v_j)$$

- Given: (Outlook=sunny; Temperature=cool; Humidity=high; Wind=strong)

$$P(\text{PlayTennis} = \text{yes}) = 9/14 = 0.64$$

$$P(\text{PlayTennis} = \text{no}) = 5/14 = 0.36$$

$$P(\text{outlook} = \text{sunny} | \text{yes}) = 2/9$$

$$P(\text{outlook} = \text{sunny} | \text{no}) = 3/5$$

$$P(\text{temp} = \text{cool} | \text{yes}) = 3/9$$

$$P(\text{temp} = \text{cool} | \text{no}) = 1/5$$

$$P(\text{humidity} = \text{hi} | \text{yes}) = 3/9$$

$$P(\text{humidity} = \text{hi} | \text{no}) = 4/5$$

$$P(\text{wind} = \text{strong} | \text{yes}) = 3/9$$

$$P(\text{wind} = \text{strong} | \text{no}) = 3/5$$

$$P(\text{yes}, \dots) \sim 0.0053$$

$$P(\text{no}, \dots) \sim 0.0206$$

$$P(\text{no} | \text{instance}) = 0.0206 / (0.0053 + 0.0206) = 0.795$$

What if we were asked about Outlook=OC ?

CIS 419/519 Fall'20

Recall: Naïve Bayes, Two Classes

- In the case of two classes we have:

$$\log \frac{P(v_j = 1|x)}{P(v_j = 0|x)} = \sum_i w_i x_i - b$$

- but since

$$P(v_j = 1|x) = 1 - P(v_j = 0|x)$$

- We showed that:

$$P(v_j = 1|x) = \frac{1}{1 + \exp(-\sum_i w_i x_i + b)}$$

- Which is simply the logistic (sigmoid) function used in the neural network representation.

94

Logistic Regression (3)

- Using the standard mapping to linear separators through the origin, we would like to minimize:

$$\min_{w,b} \sum_i^n \log P(y = +1 | x, w) = \min_{w,b} \sum_i^n \log[1 + \exp(-y_i(w^T x_i + b))]$$

- To get good generalization, it is common to add a regularization term, and the regularized logistics regression then becomes:

$$\min_w f(w) = \underbrace{\frac{1}{2} w^T w}_{\text{Regularization term}} + C \underbrace{\sum_i^m \log[1 + \exp(-y_i(w^T x_i))]}_{\text{Empirical loss}}$$

Where C is a user selected parameter that balances the two terms.

CIS 419/519 Fall'20

98

Summary: Basic Probability

- Product Rule: $P(A, B) = P(A|B)P(B) = P(B|A)P(A)$
- If A and B are independent:
 - $P(A, B) = P(A)P(B)$; $P(A|B) = P(A)$, $P(A|B, C) = P(A|C)$
- Sum Rule: $P(A \vee B) = P(A) + P(B) - P(A, B)$
- Bayes Rule: $P(A|B) = P(B|A) P(A)/P(B)$
- Total Probability:
 - If events A_1, A_2, \dots, A_n are mutually exclusive: $A_i \wedge A_j \ A_j = \Phi, \sum_i P(A_i) = 1$
 - $P(B) = \sum P(B, A_i) = \sum_i P(B|A_i) P(A_i)$
- Total Conditional Probability:
 - If events A_1, A_2, \dots, A_n are mutually exclusive: $A_i \wedge A_j \ A_j = \Phi, \sum_i P(A_i) = 1$
 - $P(B|C) = \sum P(B, A_i|C) = \sum_i P(B|A_i, C) P(A_i|C)$

Expectation of a Random Variable

- Let X be a random variable with arity k that takes the values $\{x_1, x_2, \dots, x_k\}$ with probabilities $\{p_1, p_2, \dots, p_k\}$, respectively, with $\sum_{i=1}^k p_i = 1$



- Then, the expectation of the random variable X is:

$$E[X] = \sum_{i=1}^k p(X = x_i)x_i = \sum_{i=1}^k p_i x_i$$

- Example: Let X be the outcome of tossing a fair 6-sided dice: $E[X] = 3.5$
- Important property:
 - Linearity: $E[X + Y] = E[X] + E[Y]$

Semi-Supervised Learning

- Consider the problem of Prepositional Phrase Attachment.
 - Buy car with money ; buy car with wheel
- Predict: noun (n) attachment or verb (v) attachment
 - Car with (n) or Buy with (v)
- There are several ways to generate features.
 - E.g., we can assume that all possible conjunctions of the 4 attributes are used. (15 feature in each example). Assume that we have k features
- Assume that we use naïve Bayes for learning to decide between $[n, v]$
 - Examples are: $(x_1, x_2, \dots, x_k, [n, v])$

Using Naïve Bayes

Assume we will use naïve Bayes for learning to decide between $[n, v]$; Examples are: $(x_1, x_2, \dots, x_k, [n, v])$

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

$$\begin{array}{cc} P(n) & P(v) \\ P(x_1|n) & P(x_1|v) \\ P(x_2|n) & P(x_2|v) \\ & \dots\dots \\ P(x_k|n) & P(x_k|v) \end{array}$$

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

$$P(n|x) \sim = P(n) P(x_1|n) P(x_2|n) \dots P(x_k|n)$$

and

$$P(v|x) \sim = P(v) P(x_1|v) P(x_2|v) \dots P(x_k|v)$$

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 $x = (1000)$, we have:
 $P_n(x) \sim = 0.5 \times 0.75 \times 0.5 \times 0.5 \times 0.5 = 3/64$
 $P_v(x) \sim = 0.5 \times 0.25 \times 0.75 \times 0.25 \times 0.5 = 3/256$
 - For $x=0101$ you will get: $P_n(x) \sim = 1/64 = 8/512$ $P_v(x) \sim = 9/512$
- Now, assume that in addition to the 10 labeled examples, we also have 100 unlabeled examples.
- Will that help?

Using Naïve Bayes

- For example, what can be done with the example (1000) ?
 - We have an estimate for its label...
 - But, can we use it to improve the classifier (that is, the estimation of the probabilities that we will use in the future)?
- Option 1: We can make predictions, and believe them
 - Or some of them (based on what?)
- Option 2: We can assume the example $x = (1000)$ is a
 - An n -labeled example with probability $\frac{P_n(x)}{P_n(x) + P_v(x)}$
 - A v -labeled example with probability $\frac{P_v(x)}{P_n(x) + P_v(x)}$
- Estimation of probabilities does not require working with integers!

What do we do once we have these labels?

We estimate the most likely parameters:
 $P(n), P(v); P(x_i|n), P(x_i|v)$
That is, we run NB again

Using Unlabeled Data

The discussion suggests several algorithms:

1. Use a threshold. Choose examples labeled with high confidence. Label them $[n, v]$. Retrain.
2. Use fractional examples. Label the examples with fractional labels $[p \text{ of } n, (1 - p) \text{ of } v]$. 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.
- There are other approaches to Semi-Supervised learning:
 - Most are conceptually similar: bootstrapping algorithms, self-training
 - Some are “graph-based” algorithms: assume “similar” examples have “similar labels”.
- 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.

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 α
 - 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 **four** times.

Observing *HHTH*

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

- **Scenario II:** Toss coin 0. If Head – toss coin 1; otherwise – toss coin 2

Observing the sequence *HHHT, THTT, HHHHT, HHTTH, THTTH*
produced by Coin 0, Coin1 and Coin2

Question: Estimate most likely values for p, q (the probability of H in each coin) and the probability to use each of the coins (α)

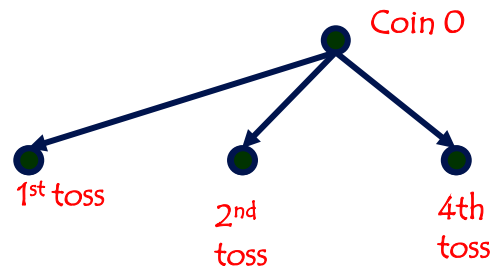
- (Mixing the Coin1 and Coin2 distributions)

- **Scenario III:** Toss coin 0. If Head – toss coin 1, o/w – toss coin 2

Observing the sequence *HHHT, HTHT, HHHT, HTTH, HTTH*
each 4 consecutive tosses are produced by Coin 1 or Coin 2

Question: Estimate most likely values for p, q and α

- (Coin0 is a hidden (latent) variable)



There is no known analytical solution to this problem (general setting).
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 (1)

- If we knew which of the data points $(HHHT)$, $(HTHT)$, $(HTTH)$ came from Coin 1 and which from Coin 2, there was no problem.
- Recall that the “simple” estimation is the **ML estimation**:
- Assume that you toss a $(p, 1 - p)$ coin m times and get k Heads $m - k$ Tails.

$$\log[P(D|p)] = \log[p^k(1 - p)^{m-k}] = k \log p + (m - k) \log(1 - p)$$

- To maximize, set the derivative w.r.t. p equal to 0:

$$d \frac{\log P(D|p)}{dp} = \frac{k}{p} - \frac{m - k}{1 - p} = 0$$

- Solving this for p , gives: $p = \frac{k}{m}$

Key Intuition (2)

- If we knew which of the data points $(HHHT)$, $(HTHT)$, $(HTTH)$ came from Coin 1 and which from Coin 2, 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 or 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.
(Labels \leftrightarrow Model Parameters) \leftrightarrow 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 this “label” estimation of the observed tosses, to estimate the **most likely** parameters
 - and so on...
- Notation: n data points; in each one: m tosses each, h_i heads in the i -th data point D^i
- What is the probability that the i -th data point, D^i , came from Coin 1 ?
- **STEP 1 (Expectation Step):** (Here $h = h_i$)

$$\begin{aligned} P_1^i &= P(\text{Coin1}|D^i) = \frac{P(D^i|\text{Coin1}) P(\text{Coin1})}{P(D^i)} = \\ &= \frac{\tilde{\alpha} \tilde{p}^h (1 - \tilde{p})^{m-h}}{\tilde{\alpha} \tilde{p}^h (1 - \tilde{p})^{m-h} + (1 - \tilde{\alpha}) \tilde{q}^h (1 - \tilde{q})^{m-h}} \end{aligned}$$

EM Algorithm (Coins) - II

- Now, 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)
 - $LL = \sum_1^n \log P(D^i | p, q, \alpha)$
- But, one of the variables – the coin's name - is hidden. We can marginalize:
 - $LL = \sum_{i=1}^n \log \sum_{y=0,1} P(D^i, y | p, q, \alpha)$
- However, the sum is inside the log, making ML solution difficult.

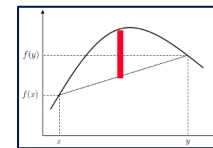
EM Algorithm (Coins) - II

- Instead of maximizing the LL we will maximize the expectation of the LL of the data (over the coin's name, y). $E[LL] = E[\sum_{i=1}^n \log P(D^i | p, q, \alpha)]$
- Explanation:
 - Since the variable y is not observed, we cannot use the complete-data log likelihood. Instead, we use the expectation of the complete-data log likelihood under the posterior distribution of y to approximate $\log P(D^i | p, q, \alpha)$ [see above]
 - We think of the likelihood $\log P(D^i | p, q, \alpha)$ as a random variable that depends on the value y 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). [Justified using Jensen's Inequality]

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

Where the inequality is due to Jensen's Inequality (concave case).

We maximize a lower bound on the Likelihood.

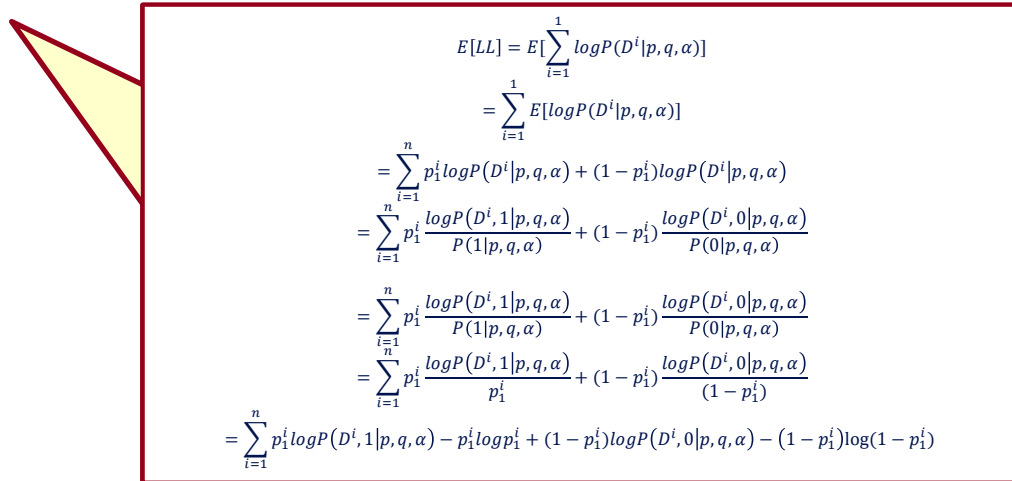


EM Algorithm (Coins) - III

- We maximize the expectation of this random variable (over the coin name).
- $$E[LL] = E[\sum_{i=1}^n \log P(D^i | p, q, \alpha)] = \sum_{i=1}^n E[\log P(D^i | p, q, \alpha)] = \text{(some math; see below)}$$

$$= \sum_{i=1}^n P_1^i \log P(D^i, 1 | p, q, \alpha) + (1 - P_1^i) \log P(D^i, 2 | p, q, \alpha)$$
~~$$- P_1^i \log P_1^i - (1 - P_1^i) \log(1 - P_1^i)$$~~

(Does not matter when we maximize)
- This is due to the linearity of the expectation and the random variable definition



$$\begin{aligned}
 E[LL] &= E\left[\sum_{i=1}^n \log P(D^i | p, q, \alpha)\right] \\
 &= \sum_{i=1}^n E[\log P(D^i | p, q, \alpha)] \\
 &= \sum_{i=1}^n p_1^i \log P(D^i | p, q, \alpha) + (1 - p_1^i) \log P(D^i | p, q, \alpha) \\
 &= \sum_{i=1}^n p_1^i \frac{\log P(D^i, 1 | p, q, \alpha)}{P(1 | p, q, \alpha)} + (1 - p_1^i) \frac{\log P(D^i, 0 | p, q, \alpha)}{P(0 | p, q, \alpha)} \\
 &= \sum_{i=1}^n p_1^i \frac{\log P(D^i, 1 | p, q, \alpha)}{P(1 | p, q, \alpha)} + (1 - p_1^i) \frac{\log P(D^i, 0 | p, q, \alpha)}{P(0 | p, q, \alpha)} \\
 &= \sum_{i=1}^n p_1^i \frac{\log P(D^i, 1 | p, q, \alpha)}{p_1^i} + (1 - p_1^i) \frac{\log P(D^i, 0 | p, q, \alpha)}{(1 - p_1^i)} \\
 &= \sum_{i=1}^n p_1^i \log P(D^i, 1 | p, q, \alpha) - p_1^i \log p_1^i + (1 - p_1^i) \log P(D^i, 0 | p, q, \alpha) - (1 - p_1^i) \log(1 - p_1^i)
 \end{aligned}$$

EM Algorithm (Coins) - IV

- Explicitly, we get:

$$\begin{aligned} E(\sum_i \log P(D^i | \tilde{p}, \tilde{q}, \tilde{\alpha})) &\simeq \\ &\simeq \sum_i P_1^i \log P(1, D^i | \tilde{p}, \tilde{q}, \tilde{\alpha}) + \sum_i (1 - P_1^i) \log P(0, D^i | \tilde{p}, \tilde{q}, \tilde{\alpha}) = \\ &= \sum_i P_1^i \log \tilde{\alpha} p^{h_i} (1 - p)^{m - h_i} + \sum_i (1 - P_1^i) \log (1 - \tilde{\alpha}) q^{h_i} (1 - q)^{m - h_i} = \\ &= \sum_i P_1^i (\log \tilde{\alpha} + h_i \log p + (m - h_i) \log (1 - p)) \\ &\quad + \sum_i (1 - P_1^i) (\log (1 - \tilde{\alpha}) + h_i \log q + (m - h_i) \log (1 - q)) \end{aligned}$$

EM Algorithm (Coins) - V

- Finally, to find the most likely parameters, we maximize the derivatives with respect to \tilde{p} , \tilde{q} , $\tilde{\alpha}$:
- **STEP 2: Maximization Step**
- (Sanity check: Think of the weighted fictional points)

Given old parameters we labeled the data. Now we compute the likelihood of the complete data (with the labels; as in the previous slide) and next we will find the new set of parameters that maximizes this likelihood.

When computing the derivatives, notice P_1^i here is a constant; it was computed using the current parameters in the E step

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

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

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

Summary: EM Algorithm (Coins)

- 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
STEP 1 (Expectation Step): (Here $h = h_i$)

Why does it work?

It can be shown that the likelihood of the observed data when computed with the set of parameters at time $t+1$, is larger than the likelihood of the observed data when computed with the set of parameters at time t . That is, the likelihood of the data grows in each round, reaching a local maxima.

$$P_1^i = P(\text{Coin1}|D^i) = \frac{P(D^i|\text{Coin1}) P(\text{Coin1})}{P(D^i)} = \frac{\tilde{\alpha} \tilde{p}^h (1 - \tilde{p})^{m-h}}{\tilde{\alpha} \tilde{p}^h (1 - \tilde{p})^{m-h} + (1 - \tilde{\alpha}) \tilde{q}^h (1 - \tilde{q})^{m-h}}$$

- STEP 2: Maximization Step**

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

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

- Iterate**

The General EM Procedure

- Initially, the parameter θ is set as θ_0
- In E step
 - We use the current parameter values θ^{old} to find the posterior distribution of the latent variables given by $p(Z|X, \theta^{\text{old}})$
 - Use $p(Z|X, \theta^{\text{old}})$ to compute the expectation of the complete-data log likelihood $\ln p(X, Z|\theta)$ under $p(Z|X, \theta^{\text{old}})$

$$Q(\theta, \theta^{\text{old}}) = \sum_Z p(Z|X, \theta^{\text{old}}) \ln p(X, Z|\theta) \quad \text{E}$$

- In M step, we need to compute θ^{new} which maximizes $Q(\theta, \theta^{\text{old}})$

$$\theta^{\text{new}} = \arg \max_{\theta} Q(\theta, \theta^{\text{old}}) \quad \text{M}$$

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 assumptions made on the probability distribution are correct, but can fail otherwise.

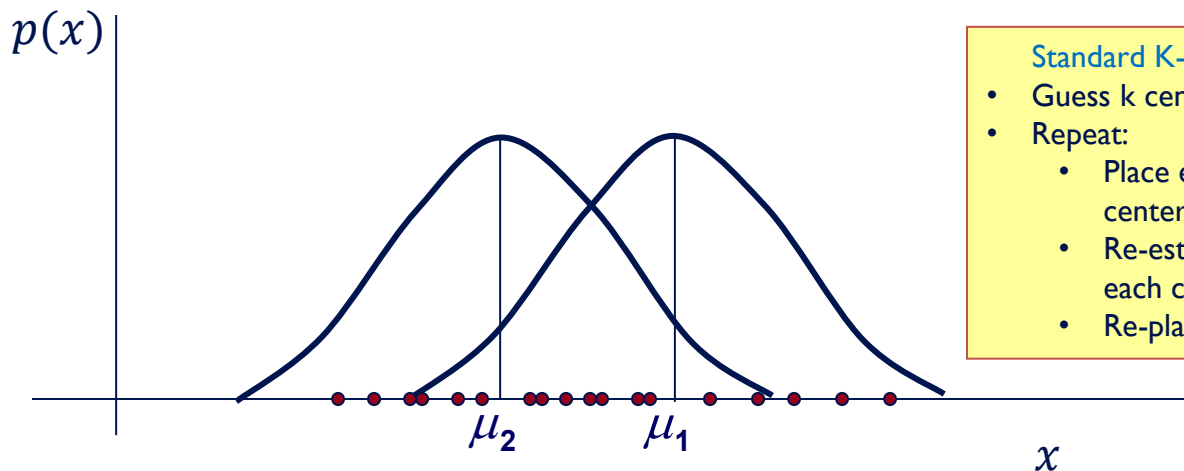
EM Summary (so far)

- EM is a general procedure for learning in the presence of unobserved variables.
- The (family of) probability distribution is known; the problem is to estimate its parameters
- In the presence of hidden variables, we can often think about it as a problem of a mixture of distributions – the participating distributions are known, we need to estimate:
 - Parameters of the distributions
 - The mixture policy
- Our previous example: Mixture of Bernoulli distributions

Example: K-Means Algorithm

K-Means is a **clustering** algorithm.

We are given data points, known to be sampled independently from a **mixture** of k Normal distributions, with means $\mu_i, i = 1, \dots, k$ and the same standard variation σ



Standard K-Means clustering:

- Guess k centers.
- Repeat:
 - Place each point in its center, based on distance.
 - Re-estimate centers for each cluster.
 - Re-place points

Example: K-Means Algorithm

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

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

- We get many data points, $D = \{x_1, \dots, x_m\}$

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

- Maximizing the log-likelihood is equivalent to minimizing:

$$\mu_{ML} = \operatorname{argmin}_{\mu} \sum_i (x_i - \mu)^2$$

- Calculate the derivative with respect to μ , we get that the
- minimal point, that is, the most likely mean is $\mu = \frac{1}{m} \sum_i x_i$

A mixture of Distributions

- As in the coin 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 \mathbf{x}_i , where is it sampled from.
- Assume that we observe data point \mathbf{x}_i ; what is the probability that it was sampled from the distribution μ_j ?

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

A Mixture of Distributions

- As in the coin example, the problem is that data is sampled from a mixture of k different normal distributions, and we do not know, for a given each data point \mathbf{x}_i , where is it sampled from.
- For a data point \mathbf{x}_i , define k binary hidden variables, $Z_{i1}, Z_{i2}, \dots, Z_{ik}$, s.t $Z_{ij} = 1$ iff \mathbf{x}_i is sampled from the j -th distribution.

$$E[z_{ij}] = 1 \cdot P(\mathbf{x}_i \text{ was sampled from } \mu_j) + 0 \cdot P(\mathbf{x}_i \text{ was not sampled from } \mu_j) = P_{ij}$$

$$E[Y] = \sum_{y_i} y_i P(Y = y_i)$$

$$E[X + Y] = E[X] + E[Y]$$

Example: K-Means Algorithms

- **Expectation:** (here: $h = \sigma, \mu_1, \mu_2, \dots, \mu_k$)

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

- Computing the likelihood given the observed data $D = \{x_1, \dots, x_m\}$ and the hypothesis h (w/o the constant coefficient)

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

Example: K-Means Algorithms

- **Maximization:** Maximizing

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

- with respect to u_j we get that:

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

- Which yields:

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

Given old parameters (h') we labeled the data. Now we compute the likelihood of the complete data (with the labels) and next we will find the new set of parameters (h) that maximizes this likelihood.

Summary: K-Means Algorithms

- Given a set $D = \{x_1, \dots, x_m\}$ of data points, **guess** initial parameters $\sigma, \mu_1, \mu_2, \dots, \mu_k$
- Compute (for all i, j)

$$p_{ij} = E[z_{ij}] = \frac{\exp[-\frac{1}{2\sigma^2}(x_i - \mu_j)^2]}{\sum_{n=1}^k \exp[-\frac{1}{2\sigma^2}(x_i - \mu_n)^2]}$$

- and a new set of means:

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

- repeat** to convergence

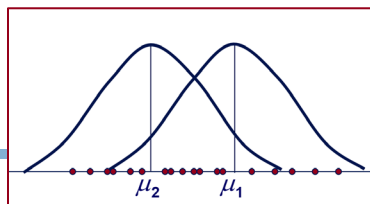
Notice that this algorithm will find the best k means in the sense of minimizing the sum of square distance.

Difference: now we place “fractional” points into clusters.
 p_{ij} is the fractional label

Recall: Standard K-Means clustering

- Guess k centers.
- Repeat:
 - Place each point in its center, based on distance.
 - Re-estimate centers for each cluster.
 - Re-place points

The **hard** EM algorithm
(threshold the distribution and keep the top option)



- We think about the probabilistic model generating the data
 - But, some of the data is not observed, making a direct estimation of the parameters more difficult
 - EM is used to estimate the model parameters in these cases

Summary: EM Algorithm (Coins)

- 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

STEP 1 (Expectation Step): (Here $h = h_i$)

$$P_1^i = P(\text{Coin1} | D^i) = \frac{P(D^i | \text{Coin1}) P(\text{Coin1})}{P(D^i)} = \frac{\tilde{\alpha} \tilde{p}^h (1 - \tilde{p})^{m-h}}{\tilde{\alpha} \tilde{p}^h (1 - \tilde{p})^{m-h} + (1 - \tilde{\alpha}) \tilde{q}^h (1 - \tilde{q})^{m-h}}$$

- STEP 2: Maximization Step

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

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

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

- Iterate

CIS 419/519 Fall'20

24

Summary: K-Means Algorithms

- Given a set $D = \{x_1, \dots, x_m\}$ of data points, **guess** initial parameters $\sigma, \mu_1, \mu_2, \dots, \mu_k$
- Compute (for all i, j)

$$p_{ij} = E[z_{ij}] = \frac{\exp[-\frac{1}{2\sigma^2}(x_i - \mu_j)^2]}{\sum_{n=1}^k \exp[-\frac{1}{2\sigma^2}(x_i - \mu_n)^2]}$$

Difference: now we place "fractional" points into clusters.
 p_{ij} is the fractional label

- and a **new set of means**:

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

- repeat** to convergence

Notice that this algorithm will find the best k means in the sense of minimizing the sum of square distance.

Recall: Standard K-Means clustering

- Guess k centers.
- Repeat:
 - Place each point in its center, based on distance.
 - Re-estimate centers for each cluster.
 - Re-place points

The hard EM algorithm (threshold the distribution and keep the top option)

CIS 419/519 Fall'20

— We have made some code available.

Summary: EM

- 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 probability distributions.
- EM is an iterative algorithm that can be shown to converge to a local maximum of the likelihood function. Thus, it might require many restarts.
- It depends on assuming a family of probability distributions.
- It has been shown to be quite useful in practice, when the assumptions made on the probability distribution are correct, but can fail otherwise.
- As examples, we have derived an important clustering algorithm, the k-means algorithm and have shown how to use it in order to estimate the most likely density function for a mixture of probability distributions.

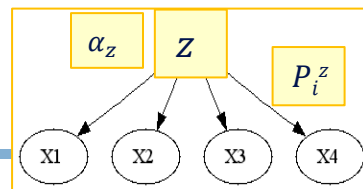
More Thoughts about EM

- **Training:** a sample of data points, $(x_0, x_1, \dots, x_n) \in \{0,1\}^{n+1}$
- **Task:** predict the value of x_0 , given assignments to all n variables.



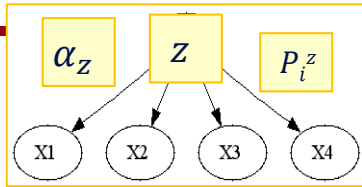
How would you model this as a learning problem? What learning approach would you use?

More Thoughts about EM



- Assume that a set $\mathbf{x}^i \in \{0,1\}^{n+1}$ of data points is generated as follows:
 - Postulate a hidden variable Z , with k values, $1 \leq z \leq k$ with probability α_z , $\sum_1^k \alpha_z = 1$
- Having randomly chosen a value z for the hidden variable, we choose the value x_i for each observable X_i to be 1 with probability p_i^z and 0 otherwise, $[i = 0, 1, 2, \dots, n]$
- **Training:** a sample of data points, $(x_0, x_1, \dots, x_n) \in \{0,1\}^{n+1}$
- **Task:** predict the value of x_0 , given assignments to all n variables.

More Thoughts about EM



Another important distinction to attend to is the fact that, once you estimated all the parameters with EM, you can answer many prediction problems e.g., $p(x_0, x_7, \dots, x_8 | x_1, x_2, \dots, x_n)$ while with Perceptron (say) you need to learn separate models for each prediction problem.

- Two options:
- **Parametric:** estimate the model using EM.
Once a model is known, use it to make predictions.
 - Problem: Cannot use EM directly without an additional assumption on the way data is generated.
- **Non-Parametric:** Learn x_0 directly as a function of the other variables.
 - Problem: which function to try and learn?
- x_0 turns out to be a linear function of the other variables, when $k = 2$ (what does it mean)?
- When k is known, the EM approach performs well; if an incorrect value is assumed the estimation fails; the linear methods performs better
[Grove & Roth 2001]