CIS 519/419
Applied Machine Learning
www.seas.upenn.edu/~cis519

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), Eric Eaton for CIS519/419 at Penn, or from other authors who have made their ML slides available.

Administration (1)

- **Surveys:**
  - **Please do it.**
  - **If 80% of the students complete it, we'll give extra credit!**

- **Projects:**

- Come to my office hours at least once to discuss the project.
- **Posters** for the projects will be presented on the last meeting of the class, December 10, 12:00-1:30.
- **Final reports** will only be due after the Final exam, on December 18
  - Specific instructions are on the web page and will be sent also on Piazza.

- HW4: Out now.

Administration (2)

- Exam:
  - The exam will take place on the originally assigned date, 12/17.
    - CHEM 102, 12-2pm
    - Structured similarly to the midterm.
    - 120 minutes; closed books.
  - What is covered:
    - Cumulative!
    - Slightly more focus on the material covered after the previous mid-term.
    - However, notice that the ideas in this class are cumulative!!
    - Everything that we present in class and in the homework assignments
    - Material that is in the slides but is not discussed in class is not part of the material required for the exam.
      - Example 1: We talked about Boosting. But not about boosting the confidence.
      - Example 2: We talked about multiclass classification: OvA, AvA, but not Error Correcting codes, and not about constraint classification (in the slides).
    - We will give practice exams. HW5 will also serve as preparation.

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 $A_1, A_2, \ldots A_n$ are mutually exclusive: $A_i \wedge A_j = \Phi$, $\sum_i P(A_i)= 1$
  - $P(B) = \sum P(B, A_i) = \sum_i P(B|A_i)\,P(A_i)$
- **Total Conditional Probability:**
  - If events $A_1, A_2, \ldots A_n$ are mutually exclusive: $A_i \wedge A_j = \Phi$, $\sum_i P(A_i)= 1$
  - $P(B|C) = \sum P(B, A_i|C) = \sum_i P(B|A_i,C)\,P(A_i|C)$

Expectation of a Random Variable

- *Let X be* a *random variable with arity k* that takes the values $\{x_1, x_2, \ldots, x_k\}$ *with probabilities* $\{p_1, p_2, \ldots, p_k\}$, *respectively,*

  with $\sum_{i=1}^{k} p_i = 1$

- Then, the expectation of the random variable X is:
- E[X] = $\sum_{i=1}^{k} p_i x_i$

- 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
- 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).

- Assume we will use naïve Bayes for learning to decide 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:

$P(n)$	$P(v)$
$P(x_1 \mid n)$	$P(x_1 \mid v)$
$P(x_2 \mid n)$	$P(x_2 \mid v)$
……	
$P(x_n \mid n)$	$P(x_n \mid v)$

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

$$P(n \mid x) \sim = P(n)\, P(x_1 \mid n)\, P(x_2 \mid n) \ldots P(x_n \mid n)$$

and

$$P(v \mid x) \sim = P(v)\, P(x_1 \mid v)\, P(x_2 \mid v) \ldots P(x_n \mid v)$$

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:

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

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

- Now, assume that in addition to the 10 labeled examples, we also have 100 unlabeled examples.
- 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 $P_n(x)/(P_n(x) + P_v(x))$
  - A v-labeled example with probability $P_v(x)/(P_n(x) + P_v(x))$

- Estimation of probabilities does not require working with integers!

Using Unlabeled Data

The discussion suggests several algorithms:

1. Use a threshold. Chose examples labeled with high confidence. Label 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.
- There are other approaches to Semi-Supervised learning:
  - Most are conceptually similar: bootstrapping algorithms
  - 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 HHHHT, THTHT, 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)

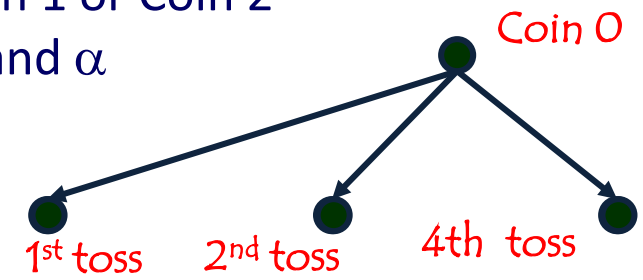
- Scenario III: Toss coin 0. If Head – toss coin 1, o/w – toss coin 2

  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

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 Coin1 and which from Coin2, 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:

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

- Solving this for p, gives: $p=k/m$

Key Intuition (2)

- 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 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.

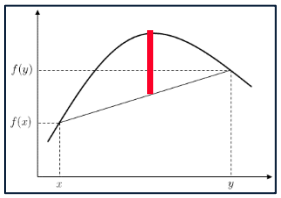
    (Labels ←→ Model Parameters)←→ Likelihood of the data
- This process can be iterated and can be shown to converge to a local maximum of the likelihood function

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 ith data point, D^i, came from Coin1 ?
- STEP 1 (Expectation Step): (Here $h=h_i$)

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

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



EM Algorithm

- Now, we would like to compute the … the parameters that maximize it.
- We will maximize the log likelihood of the d… …a points)
  - $LL = \sum_{1,n} \log P(D^i \mid p,q,\alpha)$
- But, one of the variables – the coin's name - … …n. We can marginalize:
  - $LL = \sum_{i=1,n} \log \sum_{y=0,1} P(D^i, y \mid p,q,\alpha)$
- However, the sum is inside the log, making ML … …on difficult.
- Instead of maximizing the LL we will maximize th… …pectation of the LL of the data (over the coin's name, y).
- Explanation:
  - Since the variable y is not observed, we cannot use the co… …ete-data log likelihood. Instead, we use the expectation of the complete… …ata log likelihood under the posterior distribution of y to approximate log p(D… p,q,®) [see above]
  - We think of the likelihood $\log P(D^i \mid p,q,\alpha)$ as a random variab… that depends on the value y of the coin in the i^{th} toss. Therefore, instead of m…ximizing the LL we will maximize the expectation of this random variable (over the coin's name).
    [Justified using Jensen's Inequality; later & above]

$LL = \sum_{i=1,n} \log \sum_{y=0,1} P(D^i, y \mid p,q,\alpha) =$
$= \sum_{i=1,n} \log \sum_{y=0,1} P(D^i \mid p,q,\alpha) P(y \mid D^i,p,q,\alpha) =$
$= \sum_{i=1,n} \log E\_y\, P(D^i \mid p,q,\alpha) \geq$
$\geq \sum_{i=1,n} E\_y \log P(D^i \mid p,q,\alpha)$
Where the inequality is due to Jensen's Inequality.
We maximize a lower bound on the Likelihood.

EM Algorithm

$$E[LL] = E[\sum_{i=1}^{n} logP(D^i|p,q,\alpha)] \quad (1)$$
$$= \sum_{i=1}^{n} E[logP(D^i|p,q,\alpha)] \quad (2)$$
$$= \sum_{i=1}^{n} p_1^i logP(D^i|p,q,\alpha) + (1-p_1^i)logP(D^i|p,q,\alpha) \quad (3)$$
$$= \sum_{i=1}^{n} p_1^i log\frac{P(D^i,1|p,q,\alpha)}{P(1|D^i,p,q,\alpha)} + (1-p_1^i)log\frac{P(D^i,0|p,q,\alpha)}{P(0|D^i,p,q,\alpha)} \quad (4)$$
$$= \sum_{i=1}^{n} p_1^i log\frac{P(D^i,1|p,q,\alpha)}{p_1^i} + (1-p_1^i)log\frac{P(D^i,0|p,q,\alpha)}{1-p_1^i} \quad (5)$$
$$= \sum_{i=1}^{n} p_1^i logP(D^i,1|p,q,\alpha) - p_1^i logp_1^i + (1-p_1^i)logP(D^i,0|p,q,\alpha) - (1-p_1^i)log(1-p_1^i)$$

- We maximize the expectati… the coin name).

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

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

~~$- P_1^i \log P_1^i \; (1-P_1^i) \log (1-P_1^i)$~~

(Does not matter when we maximize)

- This is due to the linearity of the expectation and the random variable definition

EM Algorithm (Coins) - IV

- Explicitly, we get:

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

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

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

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

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

EM Algorithm (Coins) - V

- Finally, to find the most likely para… 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.

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

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

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

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

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 $p(Z|X,\theta^{\text{old}})$

$$\mathcal{Q}(\theta,\theta^{\text{old}}) = \sum_{Z} p(Z|X,\theta^{\text{old}}) \ln p(X,Z|\theta)$$

E

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

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

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.

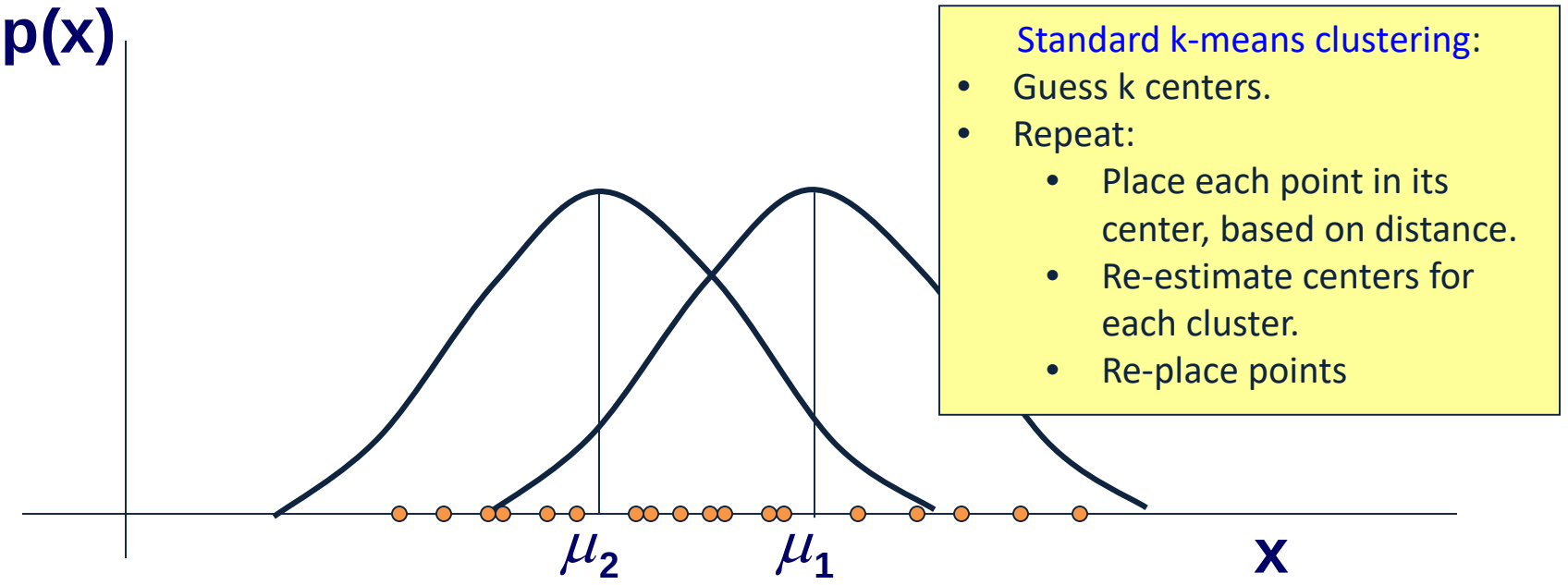
- It depends on assuming a family of probability distributions.
- In this sense, it is a family of algorithms. The update rules you will derive depend on the model assumed.
- It has been shown to be quite useful in practice, when the assumptions made on the probability distribution are correct, but can 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 μ_i, i=1,…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.

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

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

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

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 x_i, where is it sampled from.

Assume that we observe data point x_i ;what is the probability that it was sampled from the distribution μ_j ?

$$P_{ij} = P(\mu_j \mid x_i) = \frac{P(x_i \mid \mu_j)P(\mu_j)}{P(x_i)} = \frac{\frac{1}{k}P(x = x_i \mid \mu = \mu_j)}{\sum_{n=1}^{k}\frac{1}{k}P(x = x_i \mid \mu = \mu_n)} =$$

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

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 xi, where is it sampled from.

For a data point x_i, define k binary hidden variables, $z_{i1}, z_{i2}, \ldots, z_{ik}$, s.t $z_{ij} = 1$ iff x_i is sampled from the j-th distribution.

$$E[z_{ij}] = 1 \bullet P(x_i \text{ was sampled from } \mu_j) +$$

$$0 \bullet P(x_i \text{ was not sampled from } \mu_j) = P_{ij}$$

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

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

Example: K-Means Algorithms

Expectation: (here: h = $\sigma, \mu_1, \mu_2, \ldots, \mu_k$)

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

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

$$\ln(P(Y \mid h)) = \sum_{i=1}^{m} -\frac{1}{2\sigma^2} \sum_j z_{ij}(x_i - \mu_j)^2$$

$$E[\ln(P(Y \mid h))] = E\left[\sum_{i=1}^{m} -\frac{1}{2\sigma^2} \sum_j z_{ij}(x_i - \mu_j)^2\right] =$$

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

Example: K-Means Algorithms

Maximization: Maximizing

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

with respect to μ_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 = $\{x_1,\ldots,x_m\}$ of data points,

guess initial parameters $\sigma, \mu_1, \mu_2, \ldots, \mu_k$

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

Difference: now we place "fractional" points into clusters.

Recall: Stand 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

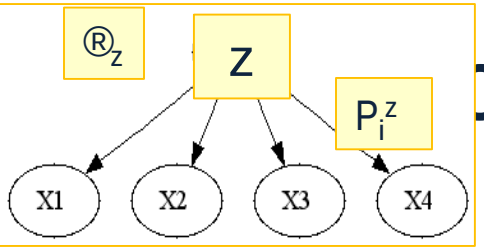
Notice that this algorithm will find the bes
sense of minimizing the sum of square dis

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, might requires 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, $(x_0, x_1, \ldots, x_n) \in \{0,1\}^{n+1}$
- Task: predict the value of x_0, given assignments to all n variables.

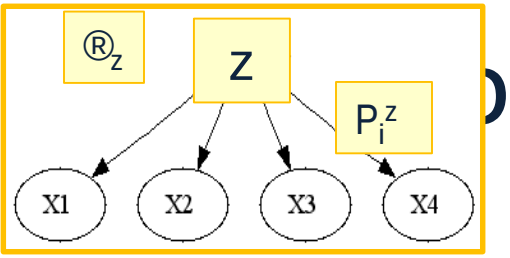


More Thoughts about EM

- Assume that a set $x^i \in \{0,1\}^{n+1}$ of data points is generated as follows:
- Postulate a hidden variable Z, with k values, $1 \le z \le k$

  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, ….n]

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



ore Thoughts about EM

- 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 variables, when k=2 (what does it mean)?
- When k is known, the EM approach pe if an incorrec methods

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., $p(x_0, x_7, \ldots, x_8 | x_1, x_2, \ldots, x_n)$ while with Perceptron (say) you need to learn separate models for each prediction problem.