Statistical Learning for Text Data Analytics
Text Categorization 2: Clustering

Yangqiu Song

Hong Kong University of Science and Technology

yqsong@cse.ust.hk

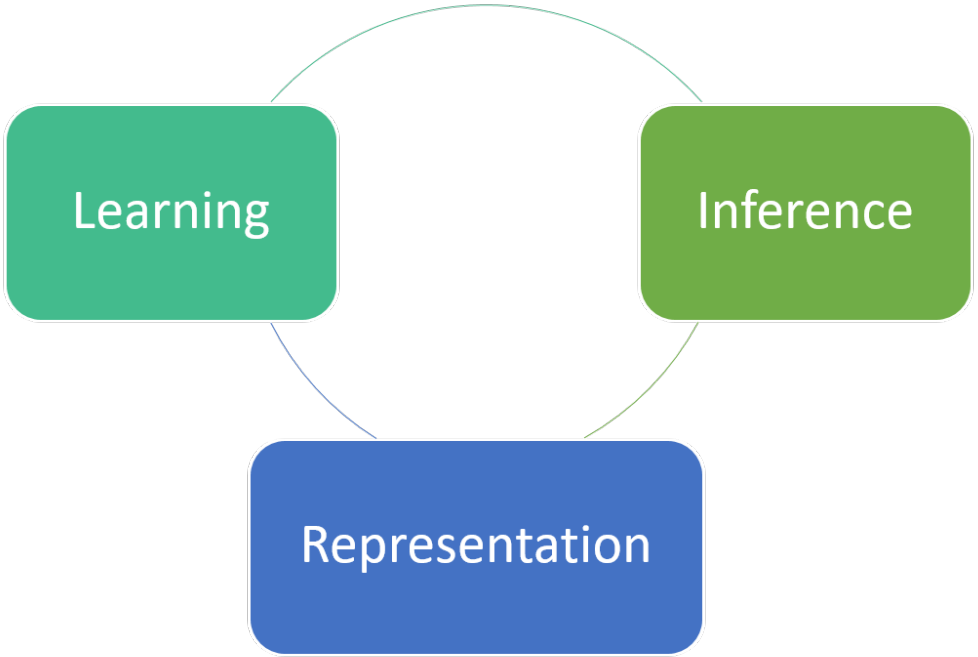
Spring 2018

∗Contents are based on materials created by Noah Smith, Xiaojin (Jerry) Zhu, Eric Xing, Vivek Srikumar, Dan Roth

Reference Content

- Noah Smith. CSE 517: Natural Language Processing https://courses.cs.washington.edu/courses/cse517/16wi/
- Xiaojin (Jerry) Zhu. CS 769: Advanced Natural Language Processing. http://pages.cs.wisc.edu/~jerryzhu/cs769.html
- Eric Xing. 10715 Advanced Introduction to Machine Learning. https://www.cs.cmu.edu/~epxing/Class/10715/lectures/lecture1.pdf
- Vivek Srikumar. CS 6355 Structured Prediction. https://svivek.com/teaching/structured-prediction/spring2018/
- Dan Roth. CS546: Machine Learning and Natural Language . http://l2r.cs.uiuc.edu/~danr/Teaching/CS546-16/

Course Topics



- Representation: language models, word embeddings, topic models
- Learning: supervised learning, unsupervised learning, semi-supervised learning, sequence models, deep learning, optimization techniques
- Inference: constraint modeling, joint inference, search algorithms

NLP applications: tasks introduced in Lecture 1

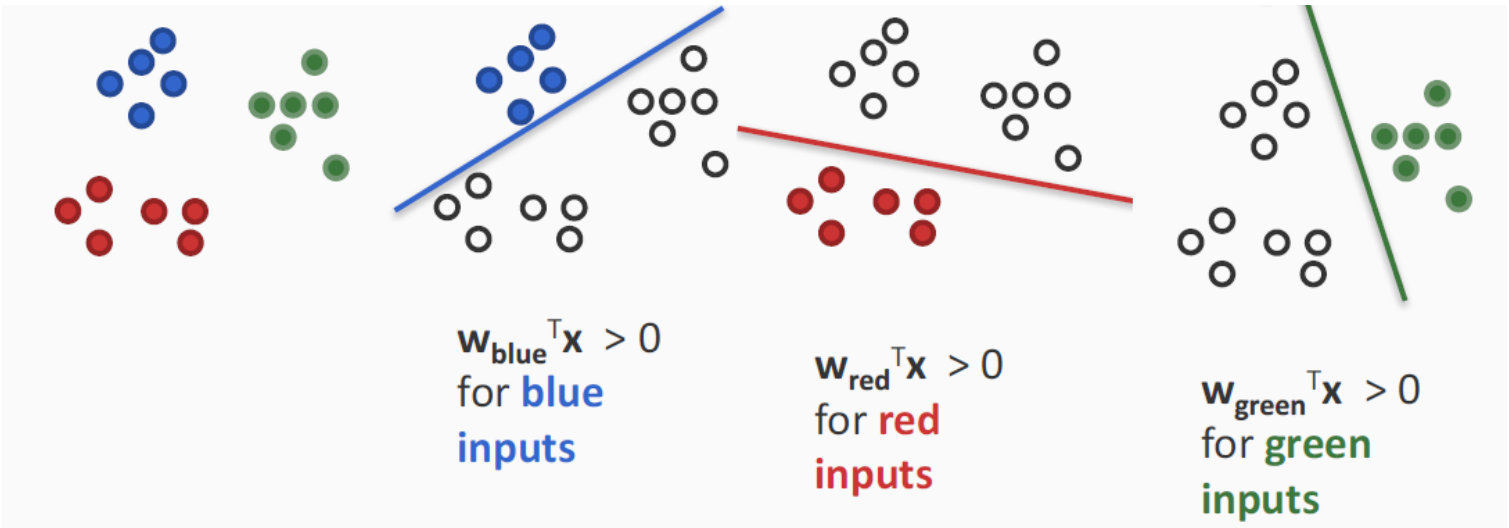
Overview

Binary to Multi-class

- Decompose the prediction into multiple binary decisions
  - One-vs-all
    - No theoretical justification
    - Calibration issues: We are comparing scores produced by K classifiers trained independently. No reason for the scores to be in the same numerical range!
    - Might not always work: Yet, works fairly well in many cases, especially if the underlying binary classifiers are tuned, regularized
  - All-vs-all
    - $O(K^2)$ weight vectors to train and store
    - Size of training set for a pair of labels could be very small, leading to overfitting of the binary classifiers
    - Prediction is often ad-hoc and might be unstable. E.g., What if two classes get the same number of votes? For a tournament, what is the sequence in which the labels compete?

Recall: One-vs-all Classification

- Assumption: Each class individually separable from all the others
- Train K binary classifiers $\mathbf{w}_1, \mathbf{w}_2, \ldots \mathbf{w}_K$ using any binary classification algorithm we have seen
- Prediction: "Winner Takes All": $label = \arg\max_i \mathbf{w}_i^\top \mathbf{x}$



Training a Single Classifier

- Rewrite input features and weight vector
  - Define a feature vector for label i being associated to input $\mathbf{x}$
  - Stack all weight vectors into an nK-dimensional vector

$$\phi(\mathbf{x}, i) = \begin{bmatrix} \mathbf{0}_n \\ \vdots \\ \mathbf{x} \\ \vdots \\ \mathbf{0}_n \end{bmatrix}_{nK \times 1} \qquad \mathbf{w} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_K \end{bmatrix}_{nK \times 1}$$

This is called the Kesler construction

Let Us Examine One-vs-all Again

- For an example with label i, we want $\mathbf{w}_i^\top \mathbf{x} > \mathbf{w}_j^\top \mathbf{x}$ for all j
- This is equivalent to

$$\mathbf{w}^\top \phi(\mathbf{x}, i) > \mathbf{w}^\top \phi(\mathbf{x}, j)$$

or

$$\mathbf{w}^\top [\phi(\mathbf{x}, i) - \phi(\mathbf{x}, j)] > 0$$

Notes

- The number of weights is still same as one-vs-all, much less than all-vs-all K(K-1)/2
- Still account for all pairwise label preferences
- Come with theoretical guarantees for generalization
- Important idea that is applicable when we move to arbitrary structures

Linear Models for Classification

- "Linear" decision rule

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} \mathbf{w}^\top \phi(\mathbf{x}, y)$$

  where $\phi : \mathcal{V} \times \mathcal{Y} \to \mathbb{R}^d$
- Parameters: $\mathbf{w} \in \mathbb{R}^d$
- What does this remind you of?

MLE for Multinomial Logistic Regression

- When we discussed log-linear language models, we transformed the score into a probability distribution. Here, that would be

$$P(y|\mathbf{x}) = \frac{\exp(\mathbf{w}^\top \phi(\mathbf{x}, y))}{\sum_{y'} \exp(\mathbf{w}^\top \phi(\mathbf{x}, y'))}$$

- MLE can be rewritten as a maximization problem:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \sum_{\mathbf{x},y} \underbrace{\mathbf{w}^\top \phi(\mathbf{x}, y)}_{\text{hope}} - \underbrace{\log \sum_{y'} \exp(\mathbf{w}^\top \phi(\mathbf{x}, y'))}_{\text{fear}}$$

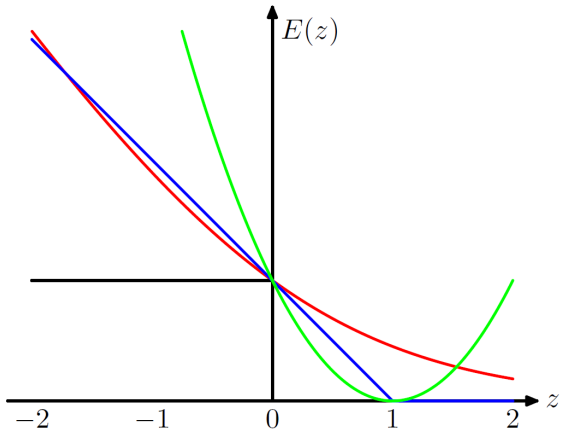
- Recall from language models:
  - Be wise and regularize!
  - Solve with batch or stochastic gradient methods
  - w_i has an interpretation

Log Loss for $(\mathbf{x}, y)$

- Another view is to minimize the negated log-likelihood, which is known as "log loss":

$$\min_{\mathbf{w}} \sum_{\mathbf{x},y} \underbrace{\log \sum_{y'} \exp(\mathbf{w}^\top \phi(\mathbf{x}, y'))}_{\text{fear}} - \underbrace{\mathbf{w}^\top \phi(\mathbf{x}, y)}_{\text{hope}}$$

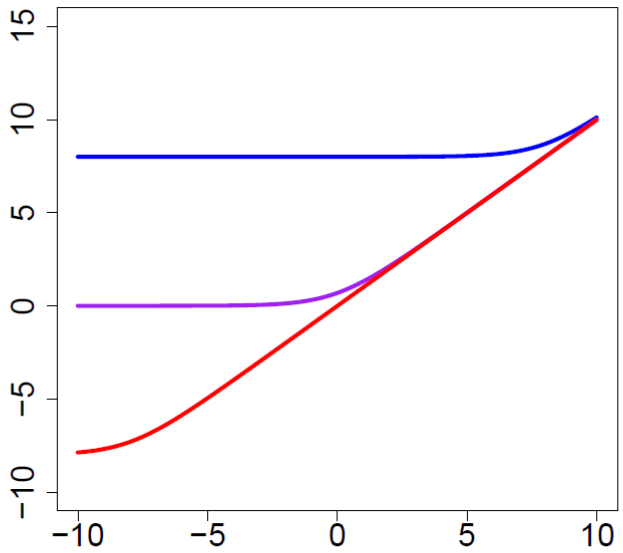
- In the binary case, where the x-axis is the difference in scores between correct and incorrect labels:



All loss functions are considered as upper-bound of "zero-one" loss (error)

Log Sum Exp

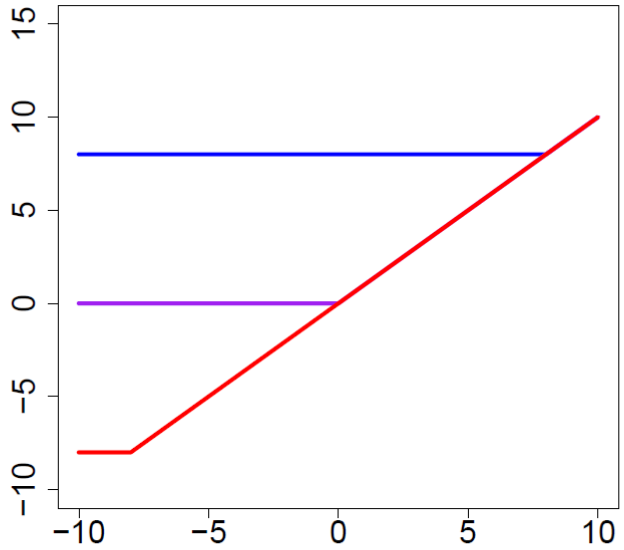
- Below, y-axis plots the log $\sum$ exp" part of the objective function (with two labels), against x, assuming the other score is one of $\{8, 0, -8\}$



$$\log(e^x + e^8),\ \log(e^x + e^0),\ \log(e^x + e^{-8})$$

Hard Maximum

- Why not use a hard max instead?



$\max(x, 8)$, $\max(x, 0)$, $\max(x, -8)$

Hinge Loss for $(\mathbf{x}, y)$

- Average log-loss

$$\min_{\mathbf{w}} \sum_{\mathbf{x},y} \underbrace{\log \sum_{y'} \exp(\mathbf{w}^\top \phi(\mathbf{x}, y'))}_{\text{fear}} - \underbrace{\mathbf{w}^\top \phi(\mathbf{x}, y)}_{\text{hope}}$$

- Hinge loss

$$\min_{\mathbf{w}} \sum_{\mathbf{x},y} \underbrace{\max_{y'} (\mathbf{w}^\top \phi(\mathbf{x}, y'))}_{\text{fear}} - \underbrace{\mathbf{w}^\top \phi(\mathbf{x}, y)}_{\text{hope}}$$

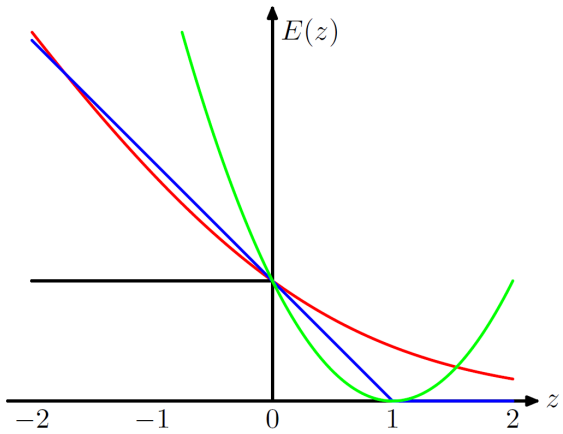
  - When two labels are tied, the function is not differentiable
  - But it's still sub-differentiable. Solution: (stochastic) sub-gradient descent!

Compare Loss

$$\min_{\mathbf{w}} \sum_{\mathbf{x},y} \underbrace{\max_{y'}(\mathbf{w}^\top \phi(\mathbf{x}, y'))}_{\text{fear}} - \underbrace{\mathbf{w}^\top \phi(\mathbf{x}, y)}_{\text{hope}}$$

In binary case:

$$\Rightarrow \min_{\mathbf{w}} \sum_{\mathbf{x},y} \max\{0, -y\mathbf{w}^\top \mathbf{x}\}$$



Any thoughts about negative sampling?

Minimizing Hinge Loss: Perceptron

$$\min_{\mathbf{w}} \sum_{m=1}^{M} \max_{y'} (\mathbf{w}^\top \phi(\mathbf{x}_m, y')) - \mathbf{w}^\top \phi(\mathbf{x}_m, y_m)$$

- Stochastic subgradient descent on the above is called the perceptron algorithm
  - For $t = 1, \ldots, T$
    - Pick i_t randomly from $\{1, \ldots, n\}$
    - $\hat{y}_{i_t} = \arg\max_{y'} \mathbf{w}^\top \phi(\mathbf{x}, y')$
    - $\mathbf{w} \leftarrow \mathbf{w} - \eta \left(\mathbf{w}^\top \phi(\mathbf{x}_{i_t}, \hat{y}_{i_t}) - \mathbf{w}^\top \phi(\mathbf{x}_{i_t}, y_{i_t}) \right)$

Error Costs

- Suppose that not all mistakes are equally bad
- E.g., false positives vs. false negatives in spam detection
- Let $\text{cost}(y', y)$ quantify the "badness" of substituting y' for correct label y
- Intuition: estimate the scoring function so that $\text{score}(y) - \text{score}(y') \propto \text{cost}(y', y)$

$$\left(\max_{y'} (\mathbf{w}^\top \phi(\mathbf{x}, y')) + \text{cost}(y, y') \right) - \mathbf{w}^\top \phi(\mathbf{x}, y)$$

General Remarks

- Text classification: many problems, all solved with supervised learners
  - Lexicon features can provide problem-specific guidance
- Naive Bayes, log-linear, and linear SVM are all linear methods that tend to work reasonably well, with good features and smoothing/regularization
- Rumor: random forests are widely used in industry when performance matters more than interpretability
- Lots of papers about neural networks, though with hyper-parameter tuning applied fairly to linear models, the advantage is not clear (Yogatama et al. (2015))
- Lots of work on feature design

Overview

∗Contents are based on materials created by Noah Smith, Xiaojin Zhu, Eric Xing, Vivek Srikumar, Dan Roth

Clustering

- Clustering is an unsupervised learning method
- Given items $\mathbf{x}_1, \ldots, \mathbf{x}_M \in \mathbb{R}^d$, the goal is to group them into reasonable clusters
- We also need a pairwise distance/similarity function between items, and sometimes the desired number of clusters
- When documents are represented by feature vectors, a commonly used similarity measure is the cosine similarity

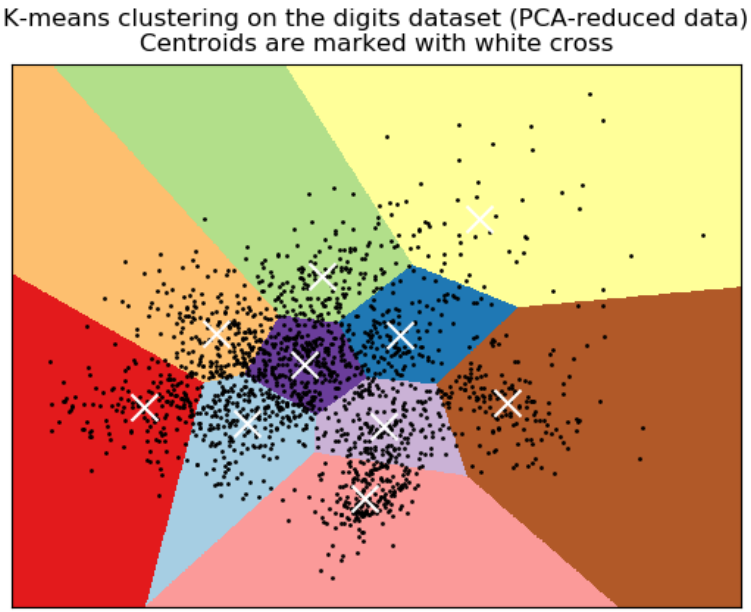
$$sim(\mathbf{x}, \mathbf{x}') = \cos(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^\top \mathbf{x}'}{||\mathbf{x}|| \cdot ||\mathbf{x}'||}$$

- This similarity has the nice property that document length is implicitly normalized (so that a long document can be similar to a short document)

K-Means Clustering

1. Randomly choose K centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$
2. Repeat
   3. Assign $\mathbf{x}_1, \ldots, \mathbf{x}_M$ to their nearest centers to obtain $\hat{y}_m$, respectively
   4. Update $\boldsymbol{\mu}_k = \frac{1}{\sum_m I(\hat{y}_m = k)} \sum_m \mathbf{x}_m I(\hat{y}_m = k)$
5. Until the clusters no longer change

Step 3 is equivalent to creating a Voronoi diagram under the current centers

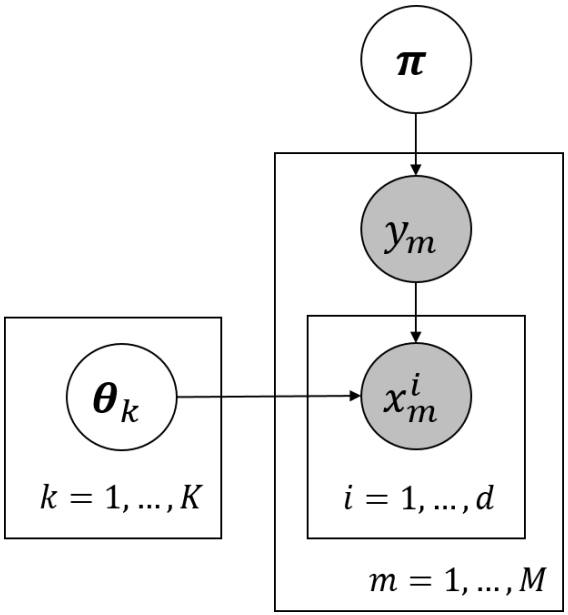


http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html

K-Means Clustering Remarks

- K-means clustering is sensitive to the initial cluster centers
- It is in fact an optimization problem with a lot of local optima
  - To be exact, k-means clustering is a special case of Gaussian Mixture Model (GMM) when the covariance of the Gaussian components tends to zero
- It is of course sensitive to k too
- Both should be chosen with care

Recall Naive Bayes Classifier: A Generative View



Both y_m and $\mathbf{x}_m = (x_m^1, \ldots, x_m^d)^T$ are observed variables; π and θ_k are parameters

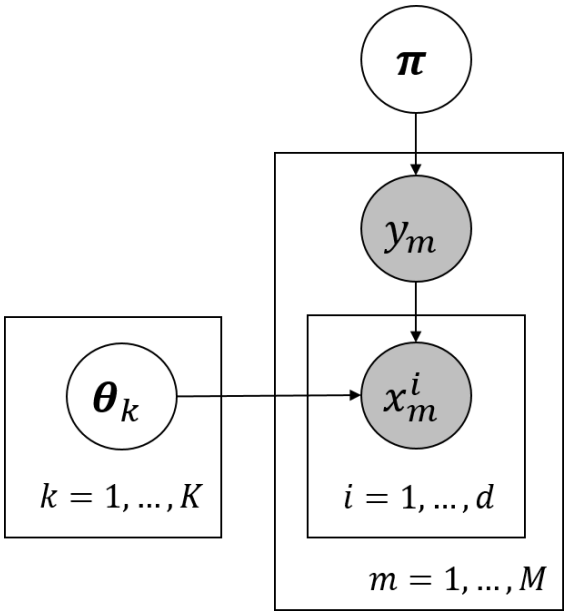
Naive Bayes from Class Conditional Unigram Model

- For $m = 1, \ldots, M$
  - Choose $y_m \sim Multinomial(y_m|1, \pi)$
  - Choose $N_m = \sum_j^d x_m^j \sim Poisson(\xi)$
  - For $n = 1, \ldots, N_m$
    - Choose $v \sim Multinomial(v|1, \theta_{*|y_m}) = \prod_{j=1}^d (\theta_{*|y_m}^j)^{v=j}$

Alternative views

- Choose $\mathbf{x}_m \sim Multinomial(\mathbf{X}|N_m, \theta_{*|y_m}) = \begin{pmatrix} N_m \\ \mathbf{x}_m \end{pmatrix} \prod_{j=1}^d (\theta_{*|y_m}^j)^{x_m^j}$
- Choose $x_m^d \sim Binomial(X|N_m, \theta_{*|y_m}^j) = \begin{pmatrix} N_m \\ x_m^j \end{pmatrix} (\theta_{*|y_m}^j)^{x_m^j} (1 - \theta_{*|y_m}^j)^{N_m - x_m^j}$

Parameter Estimation (based on Multinomial)



Both y_m and $\mathbf{x}_m = x_m^1, \ldots, \mathbf{x}_m^d$ are observed variables; $\boldsymbol{\pi}$ and $\boldsymbol{\theta}_k$ are parameters

Maximum likelihood of the training set:

$$\begin{aligned}\mathcal{J} &= \log \prod_{m=1}^{M} P_{\boldsymbol{\pi},\{\boldsymbol{\theta}_k\}}(\mathbf{x}_m, y_m) \\ &= \sum_{m=1}^{M} \log P_{\boldsymbol{\pi},\{\boldsymbol{\theta}_k\}}(\mathbf{x}_m, y_m) \\ &= \sum_{m=1}^{M} \log P(y_m|\boldsymbol{\pi}) P(\mathbf{x}_m|y_m, \boldsymbol{\theta}_{*|y_m})\end{aligned}$$

We can formulate a constrained optimization problem

$$\begin{aligned}\max \quad & \mathcal{J} \\ s.t. \quad & \sum_{k=1}^{K} \pi_k = 1 \\ & \sum_{j=1}^{d} \theta_k^j = 1 (k = 1, \ldots, K)\end{aligned}$$

It's easy to solve with Lagrange multiplier and arrive at:

$$\begin{aligned}\pi_k &= \frac{|\{y_m = k\}|}{M} \\ \theta_k^j &= \frac{\sum_{m, y_m = k} x_m^j}{\sum_{m, y_m = k} \sum_{j=1}^{d} x_m^j}\end{aligned}$$

What if the documents are not labeled?

In naive Bayes, both y_m and $\mathbf{x}_m = (x_m^1, \ldots, x_m^d)^T$ are observed variables; $\boldsymbol{\pi}$ and $\boldsymbol{\theta}_k$ are parameters

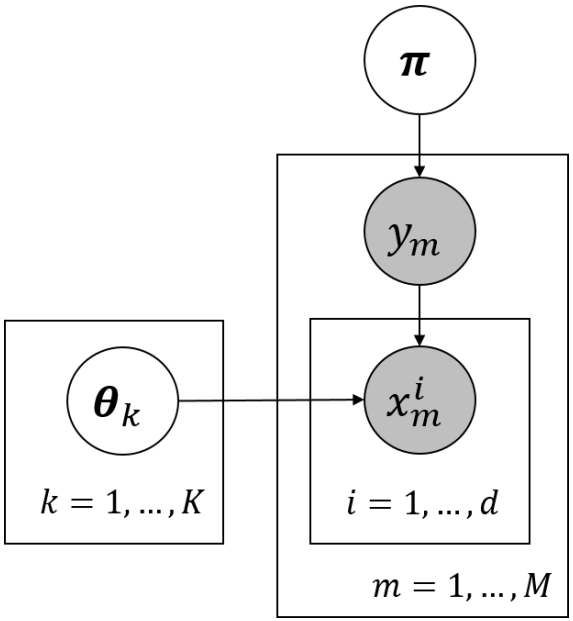


Figure: Native Bayes

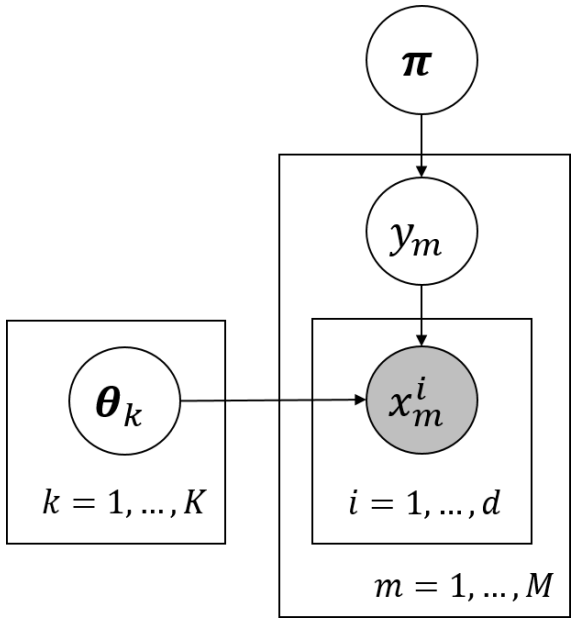


Figure: Mixture Model

However, in clustering problems, y_m is not observed (labeled before feeding into machine learning algorithm)

Expectation Maximization (EM) Algorithm

- EM might look like a heuristic method. However, it is not.
- EM is guaranteed to find a local optimum of data log likelihood
- Recall if we have complete data set $\{\mathbf{x}_m, y_m\}_{m=1}^M$ and denote parameter set as $\Theta = \{\boldsymbol{\pi}, \{\boldsymbol{\theta}_k\}\}$, the likelihood estimation of native Bayes is

$$\mathcal{J}_{NB}(\Theta) = \log \prod_{m=1}^{M} P_{\boldsymbol{\pi},\{\boldsymbol{\theta}_k\}}(\mathbf{x}_m, y_m) = \log P(\{\mathbf{x}_m, y_m\}_{m=1}^M | \Theta)$$

- However, now $\{y_m\}_{m=1}^M$ are not observed (labeled), so we treat them as hidden variables
- We instead maximize the marginal log likelihood:

$$\mathcal{J}(\Theta) = \log P(\{\mathbf{x}_m\}_{m=1}^M | \Theta)$$

Maximizing the Marginal Log Likelihood

We optimize following objective function:

$$\begin{aligned}\mathcal{J}(\Theta) &= \log P(\{\mathbf{x}_m\}_{m=1}^M|\Theta) \\ &= \sum_{m=1}^M \log P(\mathbf{x}_m|\Theta) \\ &= \sum_{m=1}^M \log \sum_{y=1}^K P(\mathbf{x}_m, y|\Theta) \\ &= \sum_{m=1}^M \log \sum_{y=1}^K P(y|\Theta)P(\mathbf{x}_m|y,\Theta) \\ &= \sum_{m=1}^M \log \sum_{y=1}^K P(y|\boldsymbol{\pi})P(\mathbf{x}_m|y,\boldsymbol{\theta}_{*|y})\end{aligned}$$

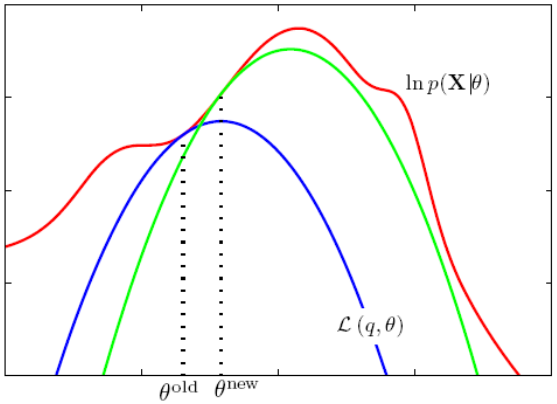
Compared to supervised learning:

$$\begin{aligned}\mathcal{J}_{NB}(\Theta) &= \log \prod_{m=1}^M P_{\boldsymbol{\pi},\{\boldsymbol{\theta}_k\}}(\mathbf{x}_m, y_m) \\ &= \sum_{m=1}^M \log P_{\boldsymbol{\pi},\{\boldsymbol{\theta}_k\}}(\mathbf{x}_m, y_m) \\ &= \sum_{m=1}^M \log P(y_m|\boldsymbol{\pi})P(\mathbf{x}_m|y_m,\boldsymbol{\theta}_{*|y_m})\end{aligned}$$

- It's more complicated with a summation inside the log!
- If we try to maximize the marginal log likelihood by setting the gradient to zero, we will find that there is no longer a nice closed form solution, unlike the joint log likelihood with complete data

EM Algorithm: General Idea

- EM is an iterative procedure to maximize the marginal log likelihood $\mathcal{J}(\Theta)$
- It constructs a concave, easy-to-optimize lower bound $\mathcal{J}(\Theta) \geq Q(\Theta, \Theta^t)$, where Θ is the variable and Θ^t is the previous, fixed, parameter
- The lower bound has an interesting property $Q(\Theta^t, \Theta^t) = \mathcal{J}(\Theta^t)$
- Therefore the new parameter Θ^{t+1} that maximizes $Q(\Theta^t, \Theta)$ is guaranteed to have $Q \geq \mathcal{J}(\Theta^t)$. Since Q lower bounds $\mathcal{J}$, we have $\mathcal{J}(\Theta^{t+1}) \geq \mathcal{J}(\Theta^t)$

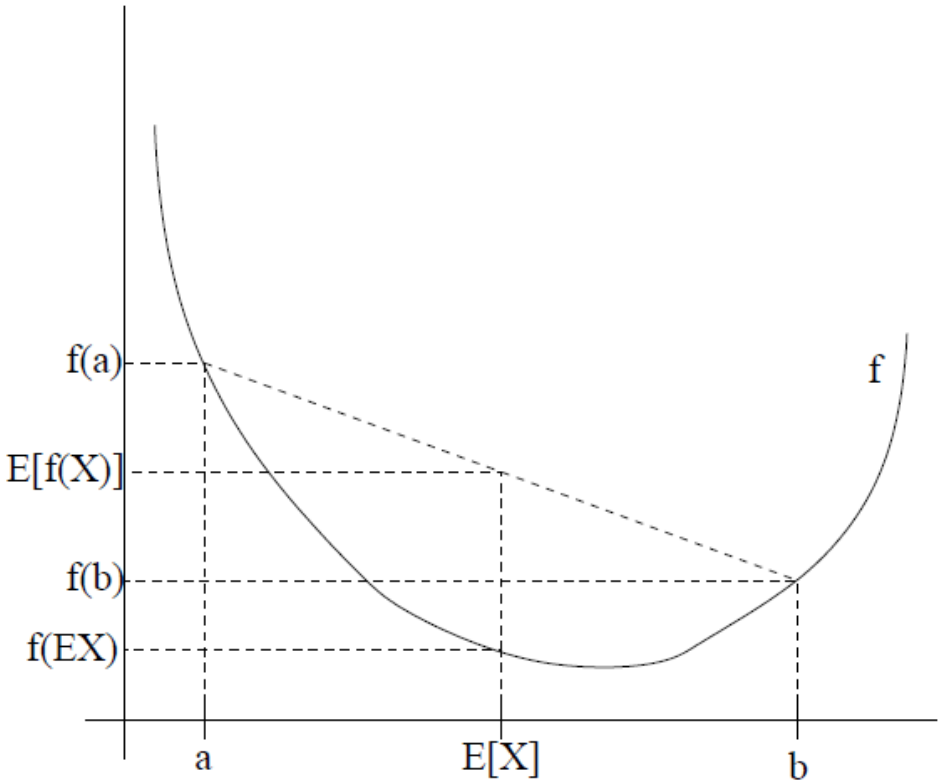


Lower Bound $Q(\Theta, \Theta^t)$

- The lower bound is obtained via Jensens inequality (concavity of log function)

$$\log \sum_i P_i f_i(x) \geq \sum_i P_i \log f_i(x)$$

which holds if the p_i's form a probability distribution



Lower Bound $Q(\Theta, \Theta^t)$ (Cont'd)

- The lower bound is obtained via Jensens inequality (concavity of log function)

$$\log \sum_i P_i f_i(x) \geq \sum_i P_i \log f_i(x)$$

  which holds if the p_i's form a probability distribution

- Then the lower bound can be derived:

$$\begin{aligned}
\mathcal{J}(\Theta^t) &= \sum_{m=1}^{M} \log \sum_{y=1}^{K} P(\mathbf{x}_m, y|\Theta^t) \\
&= \sum_{m=1}^{M} \log \sum_{y=1}^{K} q_{\mathbf{x}_m,y}(\Theta) \frac{P(\mathbf{x}_m,y|\Theta^t)}{q_{\mathbf{x}_m,y}(\Theta)} \\
&\geq \sum_{m=1}^{M} \sum_{y=1}^{K} q_{\mathbf{x}_m,y}(\Theta) \log \frac{P(\mathbf{x}_m,y|\Theta^t)}{q_{\mathbf{x}_m,y}(\Theta)} \\
&\doteq Q(\Theta, \Theta^t)
\end{aligned}$$

  where $\sum_{y=1}^{K} q_{\mathbf{x}_m,y}(\Theta) = 1$ is some distribution

E-step

$$\sum_{m=1}^{M} \log \sum_{y=1}^{K} q_{\mathbf{x}_m,y}(\Theta) \frac{P(\mathbf{x}_m, y|\Theta^t)}{q_{\mathbf{x}_m,y}(\Theta)} \geq \sum_{m=1}^{M} \sum_{y=1}^{K} q_{\mathbf{x}_m,y}(\Theta) \log \frac{P(\mathbf{x}_m, y|\Theta^t)}{q_{\mathbf{x}_m,y}(\Theta)}$$

- To make the bound tight for a particular value of Θ, we need for the step involving Jensens inequality in our derivation above to hold with equality
- For this to be true, we know it is sufficient that that the expectation be taken over a constant-valued random variable $\frac{P(\mathbf{x}_m,y|\Theta^t)}{q_{\mathbf{x}_m,y}(\Theta)} = c$
- This is easily done by choosing $q_{\mathbf{x}_m,y}(\Theta) \propto P(\mathbf{x}_m, y|\Theta^t)$
- Since $\sum_{y=1}^{K} q_{\mathbf{x}_m,y}(\Theta) = 1$, we have (considered as E-step)

$$q_{\mathbf{x}_m,y}(\Theta) = \frac{P(\mathbf{x}_m, y|\Theta^t)}{\sum_{y=1}^{K} P(\mathbf{x}_m, y|\Theta^t)} = P(y|\mathbf{x}_m, \Theta^t)$$

- The equation holds in the inequality iff $q_{\mathbf{x}_m,y} = P(y|\mathbf{x}_m, \Theta^t)$

M-step

- In M-step, we maximize the lower bound

$$\begin{aligned} Q(\Theta^t, \Theta) &= \sum_{m=1}^{M} \sum_{y=1}^{K} q_{\mathbf{x}_m, y} \log \frac{P(\mathbf{x}_m, y|\Theta)}{q_{\mathbf{x}_m, y}} \\ &= \sum_{m=1}^{M} \sum_{y=1}^{K} q_{\mathbf{x}_m, y} \log \frac{P(y_m|\boldsymbol{\pi}) P(\mathbf{x}_m|y_m, \boldsymbol{\theta}_{*|y_m})}{q_{\mathbf{x}_m, y}} \end{aligned}$$

- Now we can set the gradient of Q w.r.t. $\boldsymbol{\pi}$ and $\boldsymbol{\theta}_k$'s to zero and obtain a closed form solution

$$\begin{aligned} \pi_k &= \frac{\sum_m q_{\mathbf{x}_m, y}}{M} \\ \theta_k^j &= \frac{\sum_m q_{\mathbf{x}_m, y} x_m^j}{\sum_m \sum_{j=1}^{d} q_{\mathbf{x}_m, y} x_m^j} \end{aligned}$$

- Compared to naive Bayes:

$$\begin{aligned} \pi_k &= \frac{|\{y_m = k\}|}{M} \\ \theta_k^j &= \frac{\sum_{m, y_m = k} x_m^j}{\sum_{m, y_m = k} \sum_{j=1}^{d} x_m^j} \end{aligned}$$

EM Algorithm

- Repeat
  - E-step: compute posterior of hidden variables

$$q_{\mathbf{x}_m,y} = P(y|\mathbf{x}_m, \Theta)$$

  - M-step: parameter estimation by maximizing the lower bound

$$\pi_k = \frac{\sum_m q_{\mathbf{x}_m,y}}{M}$$
$$\theta_k^j = \frac{\sum_m q_{\mathbf{x}_m,y} x_m^j}{\sum_m \sum_{j=1}^d q_{\mathbf{x}_m,y} x_m^j}$$

- Until the convergence of the objective function

- Randomly choose K centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$
- Repeat
  - Assign $\mathbf{x}_1, \ldots, \mathbf{x}_M$ to their nearest centers to obtain $\hat{y}_m$, respectively
  - Update $\boldsymbol{\mu}_k = \frac{1}{\sum_m I(\hat{y}_m = k)} \sum_m \mathbf{x}_m I(\hat{y}_m = k)$
- Until the clusters no longer change

In practice, K-means is cheaper. We can run multiple times to find good initialization to mixture models.

Convergence of EM Algorithm

- E-step: With $q_{\mathbf{x}_m,y}(\Theta) = P(y|\mathbf{x}_m, \Theta^t)$, the equation holds, which leads

$$Q(\Theta^t, \Theta^t) = \mathcal{J}(\Theta^t)$$

- M-step: Since Θ^{t+1} maximizes $Q(\Theta^t, \Theta)$, we have

$$Q(\Theta^t, \Theta^{t+1}) \geq Q(\Theta^t, \Theta^t) = \mathcal{J}(\Theta^t)$$

- On the other hand, Q is lower bound of $\mathcal{J}$, we have:

$$\mathcal{J}(\Theta^{t+1}) \geq Q(\Theta^t, \Theta^{t+1}) \geq Q(\Theta^t, \Theta^t) = \mathcal{J}(\Theta^t)$$

- This shows EM algorithm always increase the objective function (log likelihood)
- By iterating, we arrive at a local maximum of it

A More General View of EM

- EM is general and applies to joint probability models whenever some random variables are missing
- EM is advantageous when the marginal is difficult to optimize, but the joint is easy
- To be general, consider a joint distribution $P(X, Z|\Theta)$, where X is the collection of observed variables, and Z unobserved variables
- The quantity we want to maximize is the marginal log likelihood

$$\mathcal{J}(\Theta) = \log P(X|\Theta) = \log \sum_{Z} P(X, Z|\Theta)$$

which we assume difficult to optimize

A More General View of EM (Cont'd)

- One can introduce an arbitrary distribution over hidden variables $Q(Z)$

$$\begin{aligned}
\mathcal{J}(\Theta) &= \log P(X|\Theta) = \log \textstyle\sum_Z P(X,Z|\Theta) \\
&= \textstyle\sum_Z Q(Z) \log P(X|\Theta) \\
&= \textstyle\sum_Z Q(Z) \log \frac{P(X|\Theta)Q(Z)P(X,Z|\Theta)}{P(X,Z|\Theta)Q(Z)} \\
&= \textstyle\sum_Z Q(Z) \log \frac{P(X,Z|\Theta)}{Q(Z)} + \sum_Z Q(Z) \log \frac{P(X|\Theta)Q(Z)}{P(X,Z|\Theta)} \\
&= \textstyle\sum_Z Q(Z) \log \frac{P(X,Z|\Theta)}{Q(Z)} + \sum_Z Q(Z) \log \frac{Q(Z)}{P(Z|X,\Theta)} \\
&= F(Q,\Theta) + KL[Q(Z)||P(Z|X,\Theta)]
\end{aligned}$$

- Note $F(Q,\Theta)$ is the right hand side of Jensen's inequality
  - If $KL > 0$, $F(Q,\Theta)$ is a lower bound of $\mathcal{J}(\Theta)$
- First consider the maximization of F on Q with Θ^t fixed
  - $F(Q,\Theta)$ is maximized by $Q(Z) = P(Z|X,\Theta^t)$ since $\mathcal{J}(\Theta)$ is fixed and KL attends its minimum zero (E-Step)
- Next consider the maximization of F on Θ with Q fixed as above
  - Note in this case $F(Q,\Theta) = Q(\Theta^t,\Theta)$ (M-Step)

Variations of EM

- Generalized EM (GEM) finds Θ that improves, but not necessarily maximizes, $F(Q, \Theta) = Q(\Theta, \Theta^t)$ in the M-step. This is useful when the exact M-step is difficult to carry out. Since this is still coordinate ascent, GEM can find a local optimum.
- Stochastic EM: The E-step is computed with Monte Carlo sampling. This introduces randomness into the optimization, but asymptotically it will converge to a local optimum.
- Variational EM: $Q(Z)$ is restricted to some easy-to-compute subset of distributions, for example the fully factorized distributions $Q(Z) = \prod_i Q(z_i)$. In general $P(Z|X, \Theta)$, which might be intractable to compute, will not be in this subset. There is no longer guarantee that variational EM will find a local optimum.

References I

Yogatama, D., Kong, L., and Smith, N. A. (2015). Bayesian optimization of text representations. In *EMNLP*, pages 2100–2105.