Notes on Machine Learning for 16.410 and 16.413

(Notes adapted from Tom Mitchell and Andrew Moore.)

Choosing Hypotheses

Generally want the most probable hypothesis given the training data

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Maximum a posteriori hypothesis h_{MAP}:

$$\begin{aligned} h_{MAP} &= \arg\max_{h \in H} P(h|D) \\ &= \arg\max_{h \in H} \frac{P(D|h)P(h)}{P(D)} \\ &= \arg\max_{h \in H} P(D|h)P(h) \end{aligned}$$

If assume $P(h_i) = P(h_j)$, then can choose the *Maximum likelihood* (ML) hypothesis

$$h_{ML} = \arg\max_{h_i \in H} P(D|h_i)$$

Brute Force MAP Hypothesis Learner

1. For each hypothesis h in H, calculate the posterior probability

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

2. Output the hypothesis h_{MAP} with the highest posterior probability

$$h_{MAP} = \operatorname*{argmax}_{h \in H} P(h|D)$$

Relation to Concept Learning

Consider our usual concept learning task

- instance space X, hypothesis space H, training examples D
- List-then-Eliminate learning algorithm (outputs set of hypotheses from the version space $VS_{H,D}$)

Choose $P(D|h)$:

- $P(D|h) = 1$ if h consistent with D
- $P(D|h) = 0$ otherwise

Choose $P(h)$ to be *uniform* distribution

- $P(h) = \frac{1}{|H|}$ for all h in H

Then,

$$P(h|D) = \begin{cases} \frac{1}{|VS_{H,D}|} & \text{if } h \text{ is consistent with } D \\ 0 & \text{otherwise} \end{cases}$$

Maximum-likelihood parameter learning:

If prior over hypotheses is uniform, then there is a standard method for maximum-likelihood parameter learning:

1. Write down an expression for the likelihood of the data as a function of the parameter(s).
2. Write down the derivative of the log likelihood with respect to each parameter.
3. Find the parameter values such that the derivatives are zero.

By taking logarithms, we reduce the product to a sum over the data, which is usually easier to maximize.) To find the maximum-likelihood value of θ, we differentiate L with respect to θ and set the resulting expression to zero.

Discrete models

Assume data set $\mathbf{d}$ is N independent draws from binomial population with unknown parameter θ, that is,

$$P(\mathbf{d}|\theta) = \prod_{j=1}^{N} P(d_j|\theta) = \theta^c(1-\theta)^l$$

c instances have $+ve$ label and $(N-c)$ have $-ve$ label and we want to learn θ.

$$\begin{aligned} \mathbf{L}(\mathbf{d}|\theta) &= \log P(\mathbf{d}|\theta) \\ &= \sum_{j=1}^{N} \log P(d_j|\theta) \\ &= c\log\theta + l\log(1-\theta) \\ \Rightarrow \frac{d\mathbf{L}(\mathbf{d}|\theta)}{d\theta} &= \frac{c}{\theta} - \frac{l}{1-\theta} = 0 \\ \Rightarrow \theta &= \frac{c}{c+l} = \frac{c}{N} \end{aligned}$$

Minimum Description Length Principle

MDL: prefer the hypothesis h that minimizes

$$\begin{aligned} h_{MDL} &= \arg\max_{h\in H} P(D|h)P(h) \\ &= \arg\max_{h\in H} \log_2 P(D|h) + \log_2 P(h) \\ &= \arg\min_{h\in H} -\log_2 P(D|h) - \log_2 P(h) \end{aligned}$$

- $-\log_2 P(h)$ is length of h under optimal code
- $-\log_2 P(D|h)$ is length of D given h under optimal code

Most Probable Classification of New Instances

So far we've sought the most probable *hypothesis* given the data D (i.e., h_{MAP})

Given new instance x, what is its most probable *classification*?

- $h_{MAP}(x)$ is not the most probable classification

Bayes Optimal Classifier

$$\arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D)$$

Gibbs Classifier

Bayes optimal classifier provides best result, but can be expensive if many hypotheses.

Gibbs algorithm:

1. Choose one hypothesis at random, according to $P(h|D)$
2. Use this to classify new instance

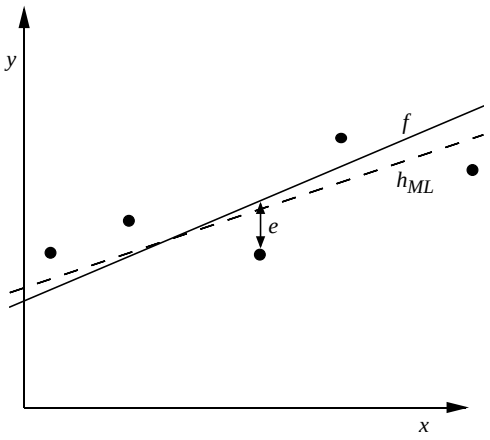
Surprising fact: Assume target concepts are drawn at random from H according to priors on H. Then:

$$E[error_{Gibbs}] \leq 2E[error_{BayesOptimal}]$$

Suppose correct, uniform prior distribution over H, then

- Pick any hypothesis from version space with uniform probability
- Its expected error is no worse than twice Bayes optimal

Learning A Real Valued Function



Consider any real-valued target function f

Training examples $\langle x_i, d_i \rangle$, where d_i is noisy training value

- $d_i = f(x_i) + e_i$
- e_i is random variable (noise) drawn independently for each x_i according to some Gaussian distribution with mean=0

$$\begin{aligned} h_{ML} &= \underset{h \in H}{\operatorname{argmax}} p(D|h) \\ &= \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} p(d_i|h) \\ &= \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(\frac{d_i - h(x_i)}{\sigma})^2} \end{aligned}$$

Maximize natural log of this instead...

$$\begin{aligned} h_{ML} &= \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} \ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2} \left(\frac{d_i - h(x_i)}{\sigma} \right)^2 \\ &= \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} -\frac{1}{2} \left(\frac{d_i - h(x_i)}{\sigma} \right)^2 \\ &= \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} -\left(d_i - h(x_i) \right)^2 \\ &= \underset{h \in H}{\operatorname{argmin}} \sum_{i=1}^{m} \left(d_i - h(x_i) \right)^2 \end{aligned}$$

Then the maximum likelihood hypothesis h_{ML} is the one that minimizes the sum of squared errors.

Naive Bayes Classifier

Assume data has attributes $a_1, a_2, \ldots, a_n$ and has possible labels v_j. The Naive Bayes assumption:

$$P(a_1, a_2 \ldots a_n | v_j) = \prod_i P(a_i | v_j)$$

Assume target function $f : X \rightarrow V$, where each instance x described by attributes $\langle a_1, a_2 \ldots a_n \rangle$.

Most probable value of $f(x)$ is:

$$\begin{aligned} v_{MAP} &= \operatorname*{argmax}_{v_j \in V} P(v_j | a_1, a_2 \ldots a_n) \\ v_{MAP} &= \operatorname*{argmax}_{v_j \in V} \frac{P(a_1, a_2 \ldots a_n | v_j) P(v_j)}{P(a_1, a_2 \ldots a_n)} \\ &= \operatorname*{argmax}_{v_j \in V} P(a_1, a_2 \ldots a_n | v_j) P(v_j) \end{aligned}$$

which gives

$$\textbf{Naive Bayes classifier: } v_{NB} = \operatorname*{argmax}_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$$

Bayes Algorithm

Naive_Bayes_Learn($examples$)

For each target value v_j

$\hat{P}(v_j) \leftarrow$ estimate $P(v_j)$

For each attribute value a_i of each attribute a

$\hat{P}(a_i | v_j) \leftarrow$ estimate $P(a_i | v_j)$

Bayes: Subtleties

1. Conditional independence assumption is often violated: $P(a_1, a_2 \ldots a_n | v_j) = \prod_i P(a_i | v_j)$

   ...but it works surprisingly well anyway. Note don't need estimated posteriors $\hat{P}(v_j | x)$ to be correct; need only that

   $$\operatorname*{argmax}_{v_j \in V} \hat{P}(v_j) \prod_i \hat{P}(a_i | v_j) = \operatorname*{argmax}_{v_j \in V} P(v_j) P(a_1 \ldots, a_n | v_j)$$

2. Naive Bayes posteriors often unrealistically close to 1 or 0
3. What if none of the training instances with target value v_j have attribute value a_i? Then

   $$\hat{P}(a_i | v_j) = 0, \text{ and...} \hat{P}(v_j) \prod_i \hat{P}(a_i | v_j) = 0$$

   Typical solution is Bayesian estimate for $\hat{P}(a_i | v_j)$

   $$\hat{P}(a_i | v_j) \leftarrow \frac{n_c + mp}{n + m}$$

   where

   - n is number of training examples for which $v = v_j$,
   - n_c number of examples for which $v = v_j$ and $a = a_i$
   - p is prior estimate for $\hat{P}(a_i | v_j)$
   - m is weight given to prior (i.e. number of "virtual" examples)

Bayesian Belief Networks

Recall that X is *conditionally independent* of Y given Z if the probability distribution governing X is independent of the value of Y given the value of Z; that is, if

$$(\forall x_i, y_j, z_k)\ P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z = z_k)$$

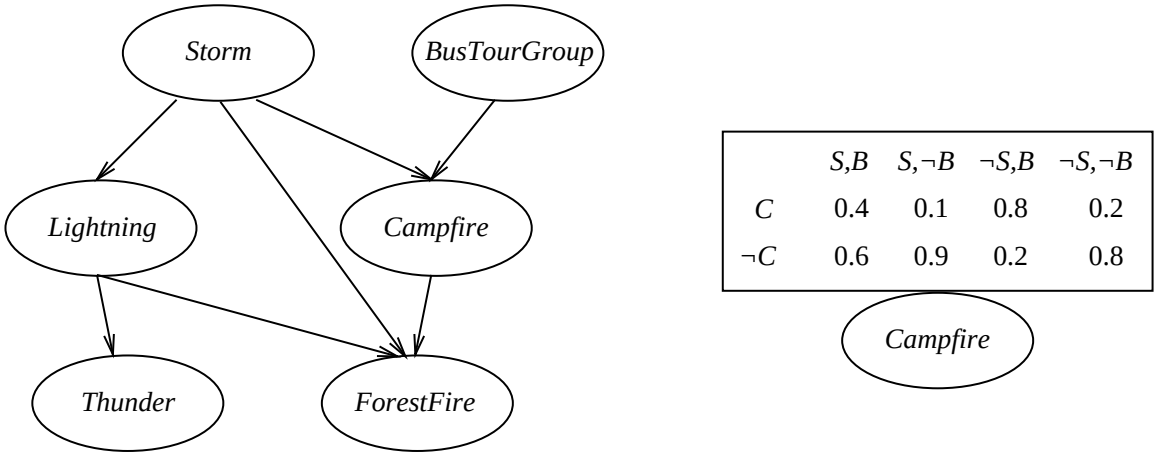
more compactly, we write

$$P(X|Y,Z) = P(X|Z)$$

Naive Bayes uses conditional independence to justify

$$\begin{aligned} P(X,Y|Z) &= P(X|Y,Z)P(Y|Z) \\ &= P(X|Z)P(Y|Z) \end{aligned}$$

Bayesian Belief Network



Network represents a set of conditional independence assertions:

- Each node is asserted to be conditionally independent of its nondescendants, given its immediate predecessors.

Network uses independence assertions to represent the joint probability distribution,
e.g., $P(Storm, BusTourGroup, \ldots, ForestFire)$, over all variables compactly

$$P(y_1, \ldots, y_n) = \prod_{i=1}^{n} P(y_i | Parents(Y_i))$$

where $Parents(Y_i)$ denotes immediate predecessors of Y_i in graph. So, the joint distribution is fully defined by graph, plus the $P(y_i|Parents(Y_i))$

Inference in Bayesian Networks

How can one infer the (probabilities of) values of one or more network variables, given observed values of others?

- Bayes net contains all information needed for this inference
- If only one variable with unknown value, easy to infer it

- In general case, problem is NP hard

In practice, can succeed in many cases

- Exact inference methods work well for some network structures
- Monte Carlo methods "simulate" the network randomly to calculate approximate solutions

Learning of Bayesian Networks

Several variants of this learning task

- Network structure might be *known* or *unknown*
- Training examples might provide values of *all* network variables, or just *some*

If structure known and observe all variables

- Then it's easy as training a Naive Bayes classifier

Gradient Ascent for Bayes Nets

Suppose structure known, variables partially observable e.g., observe *ForestFire, Storm, BusTourGroup, Thunder*, but not *Lightning, Campfire...*

Let w_{ijk} denote one entry in the conditional probability table for variable Y_i in the network

$$w_{ijk} = P(Y_i = y_{ij} | Parents(Y_i) = \text{the list } u_{ik} \text{ of values})$$

e.g., if $Y_i = Campfire$, then u_{ik} might be $\langle Storm = T, BusTourGroup = F \rangle$

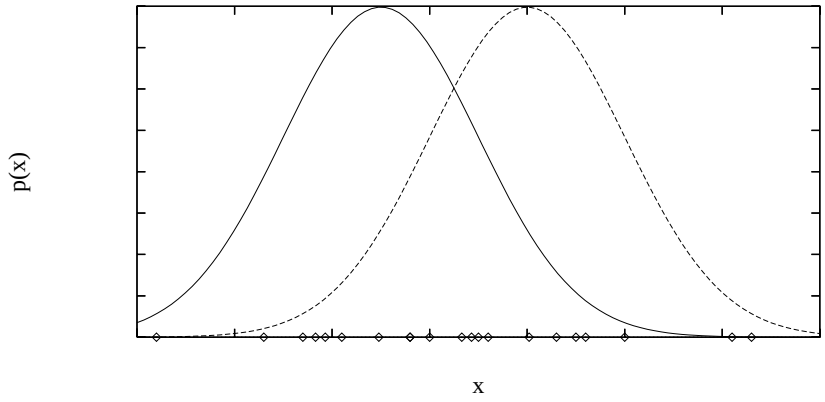
Perform gradient ascent by repeatedly

1. update all w_{ijk} using training data D

$$w_{ijk} \leftarrow w_{ijk} + \eta \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik} | d)}{w_{ijk}}$$

2. then, renormalize the w_{ijk} to assure
   - $\sum_j w_{ijk} = 1$
   - $0 \leq w_{ijk} \leq 1$

Unsupervised Learning –Expectation Maximization (EM)



Each instance x generated by

1. Choosing one of the k Gaussians with uniform probability
2. Generating an instance at random according to that Gaussian

EM for Estimating k Means

Given:

- Instances from X generated by mixture of k Gaussian distributions
- Unknown means $\langle \mu_1, \ldots, \mu_k \rangle$ of the k Gaussians
- Don't know which instance x_i was generated by which Gaussian

Determine:

- Maximum likelihood estimates of $\langle \mu_1, \ldots, \mu_k \rangle$

Think of full description of each instance as $y_i = \langle x_i, z_{i1}, z_{i2} \rangle$, where

- z_{ij} is 1 if x_i generated by jth Gaussian
- x_i observable
- z_{ij} unobservable

EM for Estimating k Means

EM Algorithm: Pick random initial $h = \langle \mu_1, \mu_2 \rangle$, then iterate

E step: Calculate the expected value $E[z_{ij}]$ of each hidden variable z_{ij}, assuming the current hypothesis $h = \langle \mu_1, \mu_2 \rangle$ holds.

$$\begin{aligned} E[z_{ij}] &= \frac{p(x = x_i | \mu = \mu_j)}{\sum_{n=1}^{2} p(x = x_i | \mu = \mu_n)} \\ &= \frac{e^{-\frac{1}{2\sigma^2}(x_i - \mu_j)^2}}{\sum_{n=1}^{2} e^{-\frac{1}{2\sigma^2}(x_i - \mu_n)^2}} \end{aligned}$$

M step: Calculate a new maximum likelihood hypothesis $h' = \langle \mu'_1, \mu'_2 \rangle$, assuming the value taken on by each hidden variable z_{ij} is its expected value $E[z_{ij}]$ calculated above. Replace $h = \langle \mu_1, \mu_2 \rangle$ by $h' = \langle \mu'_1, \mu'_2 \rangle$.

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

EM Algorithm

Converges to local maximum likelihood h and provides estimates of hidden variables z_{ij} In fact, local maximum is $E[\ln P(Y|h)]$

- Y is complete (observable plus unobservable variables) data
- Expected value is taken over possible values of unobserved variables in Y

General EM Problem

Given:

- Observed data $X = \{x_1, \ldots, x_m\}$
- Unobserved data $Z = \{z_1, \ldots, z_m\}$
- Parameterized probability distribution $P(Y|h)$, where
  - $Y = \{y_1, \ldots, y_m\}$ is the full data $y_i = x_i \cup z_i$
  - h are the parameters

Determine:

- h that (locally) maximizes $E[\ln P(Y|h)]$

General EM Method

Define likelihood function $Q(h'|h)$ which calculates $Y = X \cup Z$ using observed X and current parameters h to estimate Z

$$Q(h'|h) \leftarrow E[\ln P(Y|h')|h, X]$$

EM Algorithm:

Estimation (E) step: Calculate $Q(h'|h)$ using the current hypothesis h and the observed data X to estimate the probability distribution over Y.

$$Q(h'|h) \leftarrow E[\ln P(Y|h')|h, X]$$

Maximization (M) step: Replace hypothesis h by the hypothesis h' that maximizes this Q function.

$$h \leftarrow \underset{h'}{\operatorname{argmax}} \, Q(h'|h)$$