Model Averaging (Bayesian Learning)

We want to predict the output Y of a new case that has input $X = x$ given the training examples Es:

$$
\begin{aligned}
p(Y \mid x \wedge Es) &= \sum_{m \in M} P(Y \wedge m \mid x \wedge Es) \\
&= \sum_{m \in M} P(Y \mid m \wedge x \wedge Es) P(m \mid x \wedge Es) \\
&= \sum_{m \in M} P(Y \mid m \wedge x) P(m \mid Es)
\end{aligned}
$$

M is a set of mutually exclusive and covering models (hypotheses).

- What assumptions are made here?

Probabilistic Learning

- The posterior probability of a model m given training examples Es:

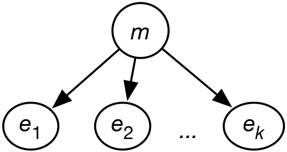
$$P(m \mid Es) = \frac{P(Es \mid m) \times P(m)}{P(Es)}$$

- The likelihood, $P(Es \mid m)$, is the probability that model m would have produced examples Es.
- The prior, $P(m)$, encodes a learning bias
- $P(Es)$ is a normalizing constant so the probabilities of the models sum to 1.
- You could try to fit the training data as well as possible by picking the maximum likelihood model, but that overfits.

Independent and Identically Distributed

- Examples $Es = [e_1, \ldots, e_k]$ are independent and identically distributed (i.i.d.) given model m if

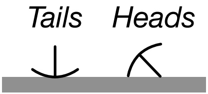
$$P(Es \mid m) = \prod_{i=1}^{k} P(e_i \mid m)$$



- Conditioning on the observed e_i and querying an unobserved e_j provides a probabilistic prediction for unseen examples.
- Conditioning on the observed e_i and querying m provides a distribution over models.

Learning probabilities — the simplest case

Observe tosses of thumbtack:
n_0 instances of *Heads* = *false*
n_1 instances of *Heads* = *true*
what should we use as $P(heads)$?



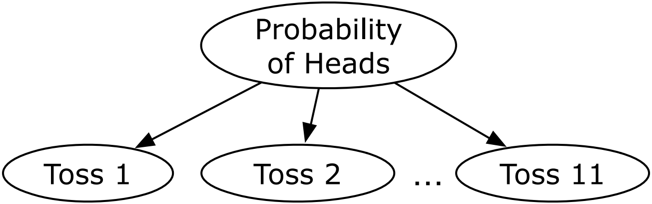
- Empirical frequency: $P(heads) = \dfrac{n_1}{n_0 + n_1}$
- Laplace smoothing [1812]: $P(heads) = \dfrac{n_1 + 1}{n_0 + n_1 + 2}$
- Informed priors: $P(heads) = \dfrac{n_1 + c_1}{n_0 + n_1 + c_0 + c_1}$
  for some informed pseudo counts $c_0, c_1 > 0$.
  $c_0 = 1$, $c_1 = 1$, expressed ignorance (uniform prior)

Pseudo-counts convey prior knowledge. Consider: "how much more would I believe α if I had seen one example with α true than if I has seen no examples with α true?"
— empirical frequency overfits to the data.

Example of Overfitting

- Consider a web site where people rate restaurants with 1 to 5 stars.
- We want to report the most liked restaurant(s) — the one predicted to have the best future ratings.
- How can we determine the most liked restaurant?
- Are the restaurants with the highest average rating the most liked restaurants?
- Which restaurants have the highest average rating?
- Which restaurants have a rating of 5?
  - Only restaurants with few ratings have an average rating of 5.
- Solution: add some "average" ratings for each restaurant!

Bayesian Learning of Probabilities



aipython.org: coinTossBN in learnBayesian.py

- $Probablity\_of\_Heads$ is a random variable representing the probability of heads.
- Domain is $\{0.0, 0.1, 0.2, \ldots, 0.9, 1.0\}$ or interval $[0, 1]$.
- $P(Toss\#n{=}Heads \mid Probablity\_of\_Heads{=}v) = v$
- $Toss\#i$ is independent of $Toss\#j$ (for $i \neq j$) given $Probablity\_of\_Heads$
- i.i.d. or independent and identically distributed.

Bayesian Learning of Probabilities

- Y has two outcomes y and $\neg y$.
  We want the probability of y given training examples Es.
- Treat the probability of y as a real-valued random variable on the interval $[0, 1]$, called ϕ. Bayes' rule gives:

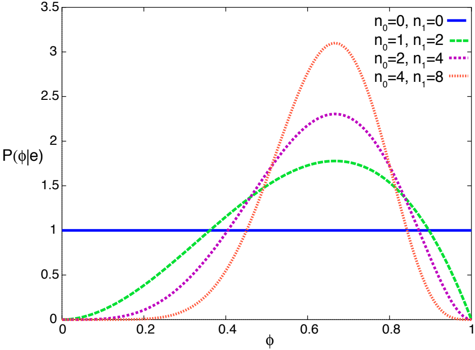
  $$P(\phi{=}p \mid Es) = \frac{P(Es \mid \phi{=}p) \times P(\phi{=}p)}{P(Es)}$$

- Suppose Es is a sequence of n_1 instances of y and n_0 instances of $\neg y$:

  $$P(Es \mid \phi{=}p) = p^{n_1} \times (1 - p)^{n_0}$$

- Uniform prior: $P(\phi{=}p) = 1$ for all $p \in [0, 1]$.

Posterior Probabilities for Different Training Examples (beta distribution)



AIPython.org see probBeta.py

- The mode is the empirical average.
- The expected value is Laplace smoothing (pseudo-count of 1).

Beta Distribution

$$Beta^{\alpha_0,\alpha_1}(p) = \frac{1}{K} p^{\alpha_1 - 1} \times (1-p)^{\alpha_0 - 1}$$

where K is a normalizing constant. $\alpha_i > 0$.

- The uniform distribution on $[0, 1]$ is $Beta^{1,1}$.
- The expected value is $\alpha_1/(\alpha_0 + \alpha_1)$.
- If the prior probability of a Boolean variable is $Beta^{\alpha_0,\alpha_1}$, the posterior distribution after observing n_1 true cases and n_0 false cases is:

$$Beta^{\alpha_0+n_0,\alpha_1+n_1}$$

- If the prior is of the form of a beta distribution, so is the posterior — called a conjugate distribution.

Categorical Variables

- Suppose Y is a categorical variable with k possible values.
- A distribution over a categorical variable is called a multinomial distribution.
- The Dirichlet distribution is the generalization of the beta distribution to cover categorical variables.
- A Dirichlet distribution has form:

$$Dirichlet^{\alpha_1,\ldots,\alpha_k}(p_1,\ldots,p_k) = \frac{\prod_{j=1}^{k} p_j^{\alpha_j - 1}}{Z}$$

where
  - p_i is the probability of the ith outcome (and so $0 \leq p_i \leq 1$)
  - α_i is a positive real number (a "count")
  - Z is a normalizing constant that ensures the integral over all the probability values is 1.

Probabilities from Experts

Problems with using probabilities from experts for cases with little data or poor data – e.g., medical diagnosis from health records:

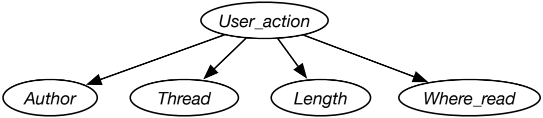
- experts are reluctant to give a precise number
- representing the uncertainty of a probability estimate
- combining the estimates from multiple experts
- combining expert opinion with actual data.

Instead of giving a real number for the probability of proposition α, an expert gives a pair $\langle n, m \rangle$ of numbers, interpreted as though the expert had observed n occurrences of α out of m trials.

- the numbers from different experts can be added
- the number can be combined with real data
- the effective sample size (m) can be tuned to reflect expertise.

Probabilistic Classifiers

- A Bayes classifier is a probabilistic model that is used for supervised learning.
- idea: the role of a class is to predict the values of features for members of that class.
- In a naive Bayes classifier the input features are conditionally independent of each other given the classification.
  Example: Suppose an agent wants to predict the user action based on properties of an online discussion:



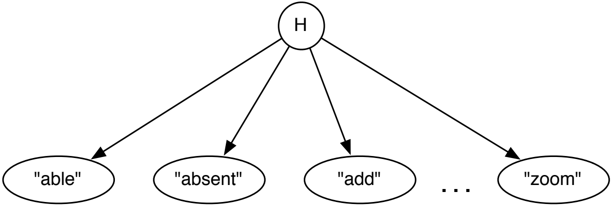
Naive Bayes Classifiers

With inputs $X_1{=}v_1, \ldots, X_k{=}v_k$, and classification, Y:

$$
\begin{aligned}
& P(y \mid X_1{=}v_1, \ldots, X_k{=}v_k) \\
& \quad = \frac{P(y) * \prod_{i=1}^{k} P(X_i{=}v_i \mid y)}{P(y) * \prod_{i=1}^{k} P(X_i{=}v_i \mid y) + P(\neg y) * \prod_{i=1}^{k} P(X_i{=}v_i \mid \neg y)} \\
& \quad = \frac{1}{1 + \frac{P(\neg y) * \prod_{i=1}^{k} P(X_i{=}v_i | \neg y)}{P(y) * \prod_{i=1}^{k} P(X_i{=}v_i | y)}} \\
& \quad = \frac{1}{1 - exp(log(\frac{P(\neg y) * \prod_{i=1}^{k} P(X_i{=}v_i | \neg y)}{P(y) * \prod_{i=1}^{k} P(X_i{=}v_i | y)}))} \\
& \quad = sigmoid\left(log(\frac{P(y)}{P(\neg y)}) + \sum_i \log \frac{(X_i{=}v_i \mid y)}{(X_i{=}v_i \mid \neg y)} \right)
\end{aligned}
$$

is a Logistic regression model when all X_i are observed

Naive Bayes Classifier: User's request for help



H is the help page the user is interested in.
We observe the words in the query.
What probabilities are required?
What counts are required?

- number of times each help page h_i is the best one
- number of times word w_j is used when h_i is the help page.

When can the counts be updated?

- When the correct page is found.

What prior counts should be used? Can they be zero?

Help System

- Suppose the help pages are $\{h_1, \ldots, h_k\}$.
- Words are $\{w_1, \ldots, w_m\}$.
- Bayes net requires:
  - $P(h_i)$, these sum to 1 ($\sum_i P(h_i) = 1$)
  - $P(w_j \mid h_i)$, do not sum to one in a set-of-words model
- Maintain "counts" (pseudo counts + observed cases):
  - c_i the number of times h_i was the correct help page
  - $s = \sum_i c_i$
  - u_{ij} the number of times h_i was the correct help page and word w_j was used in the query.
- $P(h_i) = c_i/s$
- $P(w_j \mid h_i) = u_{ij}/c_i$

Learning + Inference

- Q is the set of words in the query.
- Learning: if h_i is the correct page: Increment s, c_i and u_{ij} for each $w_j \in Q$.
- Inference:

$$
\begin{aligned}
P(h_i \mid Q) &\propto P(h_i) \prod_{w_j \in Q} P(w_j \mid h_i) \prod_{w_j \notin Q} (1 - P(w_j \mid h_i)) \\
\left.\begin{array}{c}\textit{expensive}\\ \textit{inference}\end{array}\right\rangle &= \frac{c_i}{s} \prod_{w_j \in Q} \frac{u_{ij}}{c_i} \prod_{w_j \notin Q} \frac{c_i - u_{ij}}{c_i} \\
&= \frac{c_i}{s} \prod_{w_j} \frac{c_i - u_{ij}}{c_i} \prod_{w_j \in Q} \frac{u_{ij}}{c_i - u_{ij}} \\
\left.\begin{array}{c}\textit{expensive}\\ \textit{learning}\end{array}\right\rangle &= \Psi_i \prod_{w_j \in Q} \frac{u_{ij}}{c_i - u_{ij}}
\end{aligned}
$$

Issues

If you were designing such a system, many issues arise such as:

- What if the most likely page isn't the correct page?
- What if the user can't find the correct page?
- What if the user mistakenly thinks they have the correct page?
- Can some pages never be found?
- What about common words?
- What about words that affect other words, e.g. "not"?
- What about new words?
- What do we do with new help pages?
- How can we transfer the language model to a new help system?