6.867: Exercises (Week 8)

November 3, 2017

Contents

1 Money tree [1]

You have a decision tree algorithm and you are trying to figure out which attribute is the best to test on first. You are using the information gain metric.

- You are given a set of 128 examples, with 64 positively labeled and 64 negatively labeled.
- There are three attributes, Home Owner, In Debt, and Rich.
- For 64 examples, Home Owner is true. The Home Owner=true examples are 1/4 negative and 3/4 positve.
- For 96 examples, In Debt is true. Of the In Debt=true examples, 1/2 are positive and half are negative.
- For 32 examples, Rich is true. 3/4 of the Rich=true examples are positive and 1/4 are negative.

Use $\log_2$ in your computations.

(a) What is the entropy of the initial set of examples?

(b) What is the information gain of splitting on the Home Owner attribute as the root node?

(c) What is the information gain of splitting on the In Debt attribute as the root node?

(d) What is the information gain of splitting on the Rich attribute as the root node?

(e) Which attribute do you split on?

2 Gradient boosting

Boosting is a method for making an additive model, where we explicitly construct new data sets for training each new member of the ensemble, so that new classifiers attempt to "make up for" weaknesses of the current committee.

Assuming we have a binary classification problem with data points (x, y), and we have an ensemble of classifiers $h(x;\theta)$, where θ is the parameter of the classifier. When using m classifiers with weights α, the prediction is given by the sign of the weighted sum of individual classifier outputs

$$h_m(x) = \sum_{j=1}^{m} \alpha_j h(x;\theta_j)$$

where $\alpha_j \geqslant 0$ (not necessarily summing to one) is the weight for classifier j. The base learners $h(x;\theta_j)$ are often simple (weak) classifiers such as decision stumps. There are many variations

[1] CMU 15-381 Fall 2001 Homework 5, Problem 2

on the boosting algorithm due to the choice of base learners or the loss function that the overall algorithm seeks to minimize.

The algorithm starts with no learners $h_0(x) = 0$, then at each stage of the algorithm, we add a weak learner to the current ensemble of learners to minimize the overall loss. Therefore, at stage m of the algorithm, we will try to minimize

$$J(\alpha_m, \theta_m) = \sum_{i=1}^{n} \text{Loss}\Big(y^i h_m(x^i)\Big) = \sum_{i=1}^{n} \text{Loss}\Big(y^i h_{m-1}(x^i) + y^i \alpha_m h(x^i; \theta_m)\Big)$$

where we assume that $h_{m-1}(x)$ is fixed from $m-1$ previous rounds and view J as a function of parameters α_m and θ_m associated with the new base classifier. The margin loss $\text{Loss}(\cdot)$ could be any function that is convex and decreasing in its argument (smaller loss for predictions with larger voting margin).

Let's elaborate on the algorithm a bit. We can initialize $h_0(x) = 0$ for the ensemble with no base learners. Then, at the m-th round, we perform two distinct steps to optimize θ_m and α_m, respectively. First, we find the best new classifier $h(x; \hat{\theta}_m)$ that helps with the current ensemble. Thus, we find the parameters $\hat{\theta}_m$ that minimize

$$\frac{\partial}{\partial \alpha_m} J(\alpha_m, \theta_m)\Big|_{\alpha_m=0} = \sum_{i=1}^{n} \overbrace{\text{dL}\Big(y^i h_{m-1}(x^i)\Big)}^{-W_{m-1}(i)} y^i h(x^i; \theta_m) = -\sum_{i=1}^{n} W_{m-1}(i)\, y^i h(x^i; \theta_m)$$

where $\text{dL}(z) = \text{d}/\text{dz}\,\text{Loss}(z)$ is always negative because the loss is decreasing. (We applied the chain rule of derivatives to get the above form). If we consider $W_{m-1}(i)$ as the weights of sample i after $m-1$ rounds, then we are effectively looking for a classifier that minimizes the weighted classification loss on the dataset. The values of the sample weights will differ based on different loss functions but they will always decrease as a function of how well the current ensemble classifies the training examples. We can also normalize these weights in the algorithm since the normalization does not affect the resulting $\hat{\theta}_m$.

Once we have $\hat{\theta}_m$, we determine the weight $\hat{\alpha}_m$ of the new classifier $h(x; \theta_m)$ that minimizes

$$J(\alpha_m, \hat{\theta}_m) = \sum_{i=1}^{n} \text{Loss}\left(y^i h_{m-1}(x^i) + y^i \alpha_m h(x^i; \hat{\theta}_m)\right)$$

We usually will not get a closed-form expression for $\hat{\alpha}_m$. However, the optimization problem is easy to solve since $\text{Loss}(\cdot)$ is convex and we only have one parameter to optimize.

At the end of each round, we reweight the weights of each data sample so that the classifiers added in the future focuses more on the misclassified samples.

We can now write the general boosting algorithm more succinctly. After initializing $W_0(i) = 1/n$, each boosting iteration consists of the following three steps:

(Step 1) Find $\hat{\theta}_m$ that (approximately) minimizes the weighted error ϵ_m or $2\epsilon_m - 1$ given by

$$-\sum_{i=1}^{n} W_{m-1}(i)\, y^i h(x^i; \theta_m)$$

(Step 2) Find $\hat{\alpha}_m$ that minimizes

$$J(\alpha_m, \hat{\theta}_m) = \sum_{i=1}^{n} \text{Loss}\left(y^i h_{m-1}(x^i) + y^i \alpha_m h(x^i; \hat{\theta}_m)\right)$$

(Step 3) Reweight the examples

$$W_m(i) = -c_m \, dL\left(y^i h_{m-1}(x^i) + y^i \hat{\alpha}_m h(x^i; \hat{\theta}_m\right)$$

where c_m normalizes the new weights so that they sum to one.

Now that we have a boosting algorithm for any loss function, we can select a particular one. Specifically, we will consider the logistic loss:

$$\text{Loss}(z) = \log\left(1 + \exp(-z)\right)$$

(a) Show that the unnormalized weights $W_m(i)$ from the logistic loss are bounded by 1. What can you say about the resulting normalized weights for examples that are clearly misclassified in comparison to those that are just slightly misclassified by the current ensemble? If the training data contains mislabeled examples, why do we prefer the logistic loss over the exponential loss, $\text{Loss}(z) = \exp(-z)$?

(b) Suppose the training set is linearly separable and we would use a hard-margin linear support vector machine (no slack) as a base classifier. In the first boosting iteration, what would the resulting $\hat{\alpha}_1$ be?

(c) Show that we need at most $2n$ stumps to correctly classify n training examples with distinct coordinate values. Consider the case in which the training examples are distinct points in one dimension.

3 Holmes and Watson in LA

Holmes and Watson have moved to LA. Holmes wakes up to find that his lawn is wet. He wonders if it has rained or if he left his sprinkler on. He looks at his neighbor Watson's lawn and sees that it is wet, too. So, he concludes it must have rained.

Use the binary random variables R for rain, S for sprinkler, H for Holmes' lawn being wet and W for Watson's lawn being wet. Assume you are given the following probability distributions:

$$\begin{aligned}
P(R = 1) &= 0.2 \\
P(S = 1) &= 0.1 \\
P(W = 1 \mid R = 0) &= 0.2 \\
P(W = 1 \mid R = 1) &= 1.0 \\
P(H = 1 \mid R = 0, S = 0) &= 0.1 \\
P(H = 1 \mid R = 0, S = 1) &= 0.9 \\
P(H = 1 \mid R = 1, S = 0) &= 1.0 \\
P(H = 1 \mid R = 1, S = 1) &= 1.0
\end{aligned}$$

(a) Draw the corresponding directed graphical model?

(b) What is $P(H)$?

(c) What probability expression corresponds to Holmes' belief that it rained before he goes out? What is its value?

(d) What probability expression corresponds to Holmes' belief that it rained after he sees that his lawn is wet? What is its value?

(e) What probability expression corresponds to Holmes' belief that it rained after he sees that his lawn is wet and that the sprinkler is off? What is its value?

(f) What probability expression corresponds to Holmes' belief that it rained after he sees that his lawn is wet and that Watson's is wet as well? What is its value?

(g) What probability expression corresponds to Holmes' belief that the sprinkler was on after he sees that his lawn is wet and that Watson's is wet as well? What is its value?

(h) Discuss the results of questions (c)-(e). What does these results suggest about conditional dependency of S and R (conditioned on H)?

4 Bayesian networks must be acyclic

Suppose we have a graph $G = (V, E)$ and discrete random variables $X_1, \ldots, X_n$, and define

$$f(x_1, \ldots, x_n) = \prod_{v \in V} f_v(x_v | x_{pa(v)})$$

where $pa(v)$ refers to the parents of variable X_v in G and $f_v(x_v|x_{pa(v)})$ specifies a distribution over X_v for every assignment to X_v's parents, i.e. $0 \leqslant f_v(x_v|x_{pa(v)}) \leqslant 1$ for all $x_v \in Vals(X_v)$ and $\sum_{x_v \in Vals(X_v)} f_v(x_v|x_{pa(v)}) = 1$. Recall that this is precisely the definition of the joint probability distribution associated with the Bayesian network G, where the f_v are the conditional probability distributions.

Show that if G has a directed cycle, f may no longer define a valid probability distribution. In particular, give an example of a cycle graph G and distributions f_v such that $\sum_{x_1,\ldots,x_n} f(x_1, \ldots, x_n) \neq 1$ (A valid probability distribution must be non-negative and sum to one). This is why Bayesian networks must be defined on *acyclic* graphs.

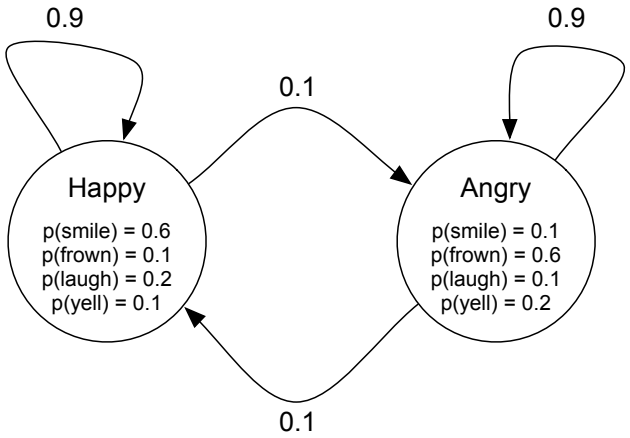
5 Hidden Markov Model

Harry lives a simple life. Some days he is Angry and some days he is Happy. But he hides his emotional state, and so all we can observe is whether he smiles, frowns, laughs, or yells. Harry's best friend is utterly confused about whether Harry is actually happy or angry and decides to model his emotional state using a hidden Markov model.

Let $X_d \in \{\text{Happy, Angry}\}$ denote Harry's emotional state on day d, and let $Y_d \in \{\text{smile, frown, laugh, yell}\}$ denote the observation made about Harry on day d. **Assume that on day 1 Harry**

is in the Happy state, i.e. $X_1 = \text{Happy}$. Furthermore, assume that Harry transitions between states exactly once per day (staying in the same state is an option) according to the following distribution: $p(X_{d+1} = \text{Happy} \mid X_d = \text{Angry}) = 0.1$, $p(X_{d+1} = \text{Angry} \mid X_d = \text{Happy}) = 0.1$, $p(X_{d+1} = \text{Angry} \mid X_d = \text{Angry}) = 0.9$, and $p(X_{d+1} = \text{Happy} \mid X_d = \text{Happy}) = 0.9$.
The observation distribution for Harry's Happy state is given by $p(Y_d = \text{smile} \mid X_d = \text{Happy}) = 0.6, p(Y_d = \text{frown} \mid X_d = \text{Happy}) = 0.1, p(Y_d = \text{laugh} \mid X_d = \text{Happy}) = 0.2$, and $p(Y_d = \text{yell} \mid X_d = \text{Happy}) = 0.1$. The observation distribution for Harry's Angry state is $p(Y_d = \text{smile} \mid X_d = \text{Angry}) = 0.1, p(Y_d = \text{frown} \mid X_d = \text{Angry}) = 0.6, p(Y_d = \text{laugh} \mid X_d = \text{Angry}) = 0.1$, and $p(Y_d = \text{yell} \mid X_d = \text{Angry}) = 0.2$.

All of this is summarized in the following figure:



Be sure to show all of your work for the below questions. Note, the goal of this question is to get you to start thinking deeply about probabilistic inference. Thus, although you could look at Chapter 17 for an overview of HMMs, try to solve this question based on first principles (also: no programming needed!).

(a) What is $p(X_2 = \text{Happy})$?

(b) What is $p(Y_2 = \text{frown})$?

(c) What is $p(X_2 = \text{Happy} \mid Y_2 = \text{frown})$?

(d) What is $p(Y_{80} = \text{yell})$?

(e) Assume that $Y_1 = Y_2 = Y_3 = Y_4 = Y_5 = \text{frown}$. What is the most likely sequence of the states? That is, compute the MAP assignment $\arg\max_{x_1,\ldots,x_5} p(X_1 = x_1, \ldots, X_5 = x_5 \mid Y_1 = Y_2 = Y_3 = Y_4 = Y_5 = \text{frown})$.

6 Naive Bayes

In this problem you will show that naive Bayes corresponds to a linear classifier. Consider using a naive Bayes algorithm for binary prediction (two classes), where the features $x_1, \ldots, x_k$ are also binary valued. Let $\theta_c = \Pr(Y = c)$ and $\theta_{ci} = \Pr(X_i = 1 | Y = c)$ for $c \in \{0, 1\}$. It will be helpful to

use the following form for the joint distribution:

$$\Pr(Y = 1, x_1, \ldots, x_k; \vec{\theta}) = \theta_1 \prod_{i=1}^{k} \theta_{1i}^{x_i} (1 - \theta_{1i})^{1-x_i}$$
$$\Pr(Y = 0, x_1, \ldots, x_k; \vec{\theta}) = \theta_0 \prod_{i=1}^{k} \theta_{0i}^{x_i} (1 - \theta_{0i})^{1-x_i}$$

For a naive Bayes model given by parameters $\vec{\theta}$, demonstrate a weight vector $\mathbf{w}$ and offset b such that for any new example $\mathbf{x}$,

$$\arg\max_y \Pr(y|\mathbf{x}; \vec{\theta}) = \arg\max_y y(\mathbf{w} \cdot \mathbf{x} + b)$$

where $\vec{\theta}$ refers to all parameters, including θ_c and θ_{ci}.
Hint: Use Bayes' rule to obtain the posterior, and then take its logarithm (noticing that this is a monotonic transformation which does not change the argmax).
Thus, if one had a sufficient amount of data, one would prefer to directly learn a linear model using logistic regression or a SVM rather than using naive Bayes, since the former consider a strictly larger hypothesis class than the latter. With limited numbers of training points (or settings where some features may be missing) naive Bayes may be preferable.