# Machine Learning 6.867: HW 1

Anonymous Authors

September 27, 2017

## 1 Probabilities [30 pt, Field Cady]

Probability is, in many ways, the most fundamental mathematical technique for machine learning. This problem will review several basic notions from probability and make sure that you remember how to do some elementary proofs.

Recall that for a discrete random variable (r.v.) X whose values are integers, we frequently use the notation $P(X = x)$ for the probability its value is $x$. If the r.v. Y is *continuous*, we typically use a "density function" $p(Y = y)$. The conditions for $P(X = x)$ to be a valid probability distribution are that $\sum_{-\infty}^{\infty} p(X = x) = 1$ and $P(X = x) \geq 0 \forall x$. Similarly for $p(Y = y)$ to be a valid continuous distribution, $\int_{-\infty}^{\infty} p(Y = y)dy = 1$ and $p(Y = y) \geq 0$.

Sometimes the underlying probability space has more than one variable (for example, the height and weight of a person). In this case, we may use notation like $p(X = x, Y = y)$ to denote the probability density function in several dimensions.

### 1.1 Expectations [10 pt]

Expectation is another word for "mean". They are similar to "average"; the difference is that an average usually refers to the average of some data we collected, whereas expectation usually refers to the underlying distribution from which we have sampled. For a discrete r.v. X, the expectation value is defined to be $E[X] = \sum_{-\infty}^{\infty} iP(X = i)$. If Y is a continuous random variable, $E[Y] = \int_{-\infty}^{\infty} yp(y)dy$.

1. Show that, for discrete r.v. W and Z, $E[W + Z] = E[W] + E[Z]$.
2. Show that, for *continuous* r.v. W and Z, $E[W + Z] = E[W] + E[Z]$.

### 1.2 Independence [10 pt]

Intuitively, two r.v. X and Y are "independent" if knowledge of the value of one tells you nothing at all about the value of the other. Precisely, if X and Y are discrete, independence means that $P(X = x, Y = y) = P(X = x)P(Y = y)$, and if they are continuous, $p(X = x, Y = y) = p(X = x)p(Y = y)$. Show the following, for *independent* r.v. X and Y:

1. If X and Y are discrete, $E[XY] = E[X]E[Y]$.
2. If X and Y are *continuous*, $E[XY] = E[X]E[Y]$.

### 1.3 Variance [10 pt]

Variance for a r.v. X indicates how "spread out" the distribution is. Precisely, if $\bar{X} = E[X]$, the variance is defined to be $Var[X] = E[(X - \bar{X})^2]$. Show the following:

1. For a (discrete *or* continuous) random variable X, $Var[X] = E[X^2] - (E[X])^2$.

   *Hint: you don't have to treat the discrete and continuous cases separately; it can be done just using expectation.*

2. Let X be continuous, and let it follow the celebrated normal distribution: $p(X = x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{(x-\mu)^2/(2\sigma^2)}$, where $\sigma$ can be any positive real number, and $\mu$ can be any real number. Show that $Var[X] = \sigma^2$.

## 2 Decision Trees and Information Theory [35+5 pt, Ni Lao]

Suppose a discrete variable $X$ has $n$ categories $1...n$. It's entropy is defined as

$$H(X) = -\sum_i P(X = i) \log P(X = i).$$

Suppose another variable $Y$ has distribution $P(Y = j)$, and joint distribution $P(X = i, Y = j)$. Then their mutual information is

$$I(X;Y) = \sum_{i,j} P(X = i, Y = j) \log \frac{P(X = i, Y = j)}{P(X = i)P(Y = j)} = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

A *K-nary code* $C$ for $X$ is a mapping from $X$ to a string (code word) $C(X)$ of which each character can have $K$ values. A *prefix code* is a code for which no code word is a prefix of other code word. The average code length is defined as $L(C) = \sum_i P(X = i) l_i$ where $l_i$ is the length of $C(i)$. It can be proved that $H(X)$ is the minimum average code length needed to encode $X$, and the minimum is reached if and only if $P(X = i) = K^{-l_i}$.

1. **The 9 ball problem [10 pt]** There are 9 metal balls. One of then is heavier than the others. Please design a strategy given a scale to find out which one is heavier with the least number of expected trials. Then show that it is optimal from information point of view. **Hint:** connect the average code length with the expected number of tests.

2. **Information gain and optimal encoding [10 pt]** show that, when using $K$-outcome tests, if each test $Y_t$ has optimal mutual information $I(Y_t; X|Y_1...Y_{t-1}) = 1$ (in base $K$ numeral system) conditioned on all previous tests, then the expected number of tests is optimal. Assume that the mapping from $X$ to $Y_t$ are deterministic ($H(Y_t|X) = 0$). **Hint:** $H(Y) \geq I(X;Y)$.

3. **The ID3 algorithm [10 pt]** If we run the ID3 algorithm on the 9 metal ball problem, will it generate the optimal decision tree? **Hint:** treat the outcome of comparing any two sets of balls as a feature, and compare their information gains.

4. **The number guessing problem [5 pt]** Alice has a favorite four digit number (e.g. 0123), and she wants Bob to guess it. At each iteration, Bob say a four digit number (e.g. 3210), and then Alice tell him how many digit is correct (four in this example), and how many of them have correct position (zero in this example). From information point of view, what is the minimum expected number of guesses Bob has to make in order to identify the number?

5. [**Extra 5 pt**] Try the number guessing game a few times with your friend. Did you achieve your estimated lower bound? Why?

# 3 KNN and Decision Trees [35 pt, Amr]

1. **(10 points)** For each of the following figures, we are given a few data points in 2-d space,each of which is labeled as either positive (blue) or negative (red). Assuming that we are using L2 distance as a distance metric, draw the decision boundary for 1-NN for each case. For example, the decision boundary for the dataset in 1.b is given in figure 2.a.

Figure 1: Data sets for problem 3.

2. **(3 points)** In class we have mentioned that NN is a *lazy* classifier that needs to store all training instances until test time. However, in this problem we were able to draw a decision boundary for the 1-NN classifier. If we decided to store this decision boundary instead of storing all training data, would that *always* result in an improvement in terms of storage (memory) requirement for this classifier? [Please answer in no longer than 2-5 sentences]

3. **(2 point)** Decision trees are known as batch learners that require the availability of all training data to build the tree. Thus the arrival of additional training data needs to be handled carefully. Does KNN suffer from this problem and why?

4. We would like to build a decision tree over the dataset in Figure 1.d. Each data item is described using two continuous attributes $(x, y)$. One way of handing continuous attributes is discretization, here we will use binary-discretization and test if a given attribute is $\geq$ a given threshold which we will take to be the midpoint. For instance, in Figure 1.d, the range of both $X$ and $Y$ is [-4,4], thus the test at the first level of the tree is either $x \geq 0$ or $y \geq 0$. In the next level, we might test for $x \geq 2$,$x \geq -2$, $y \geq 2$, etc. In other words, we always consider the current range of both attributes in the training data at a given leaf node, and formulate a test that would divide this range into two equal parts. For instance, if we apply this scheme to the data in Figure 1.c, we get the decision boundary in Figure 2.b (other solutions are possible as well).

   (a) **(2 points)** What would you expect the training error to be if we apply the above scheme to the data in Figure 1.d? and why?

   (b) **(7 points)** To limit the size of the resulting decision tree, we will stipulate that any single attribute is tested at most twice on any path from the root of the tree to a a leaf node. With this restriction, draw the decision boundary of a possible decision tree over the data in Figure 1.d. you don't need to do any calculations, just draw the decision boundary that corresponds to a suitable decision tree.

5. **(4 points)** Using the intuition you gained in this problem, state one advantage of Decision Trees over KNN and one advantage of KNN over Decision Trees.