Answers Final Exam Machine Learning
Early Group, Normal Version

January 15, 2008

Grading works as follows: You start with 1 point, and for each of the 12 subquestions you can get 3/4 points. Partial points may be awarded for partially correct answers.

1. (a)

k	1	3	5
Instance 1	White	Black	Black
Instance 2	White	White	Black

(b) Representing x_1 by assigning integers to Black, White and Brown does not work, because the difference between these integers would be meaningless. One way to represent x_1 would be as

Value	Black	White	Brown
Representation	$\begin{pmatrix}1\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\1\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\\1\end{pmatrix}$

The other feature, x_2 can just be represented by its own value. The feature vector $\mathbf{x}$ can now be composed from the three components of x_1 and one component for x_2.

For example, $x_1 =$ 'Brown' and $x_2 = 32$ would give $\mathbf{x} = \begin{pmatrix}0\\0\\1\\32\end{pmatrix}$.

(c) The influence of x_2 on the Euclidean distance (and therefore on the classifications of the algorithm) would decrease.

(d) This will be hard for the 1-nearest neighbour algorithm: Think of the decision boundary for 1-nearest neighbour from class or in Figure 8.1 from Mitchell. In the chess board case the algorithm will learn patches of Black and White. These patches only become as fine-grained as the chess board when we've seen all possible inputs.

Another way to see this is by noting that the assumption in 1-nearest neighbour that the target function doesn't vary too much locally is violated.

2. (a) These examples would be classified correctly by using the weights for the and-function. Figure 1 shows the decision boundary for $w_0 = -0.8$, $w_1 = 0.5$ and $w_2 = 0.5$.

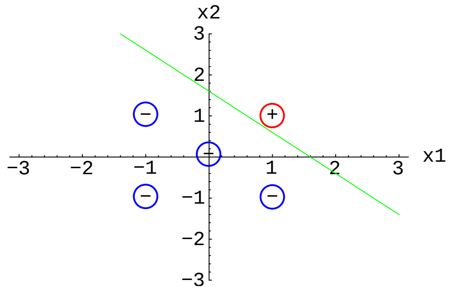

Figure 1: Decision boundary for perceptron that classifies all examples correctly.

(b) Examples that are not linearly separable can never be classified correctly by the perceptron. So for example if the target function is xor, a perceptron will always make at least one mistake if we see all possible inputs:

$D =$

y	1	1	−1	−1
x_1	−1	1	1	−1
x_2	1	−1	1	−1

3. Taking large steps initially is useful to get near the minimum quickly. Smaller steps later are necessary to avoid walking past the minimum.

4. (a) Naive Bayes would classify the new example as True:

$$\begin{aligned} &P(X_1 = \text{True} \mid Y = \text{False})P(X_2 = \text{True} \mid Y = \text{False})P(Y = \text{False}) \\ &\qquad = 0 \cdot 0 \cdot \frac{1}{3} = 0 \\ &< P(X_1 = \text{True} \mid Y = \text{True})P(X_2 = \text{True} \mid Y = \text{True})P(Y = \text{True}) \\ &\qquad = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{2}{3} = \frac{1}{6} \end{aligned}$$

(b) If there are many features, then we will not have seen (many) examples of the feature values that we are interested in together, so we cannot get a reliable estimate of the required probabilities. Therefore some assumption or preference bias is necessary. We will typically have seen the right feature values for each feature separately, however. Therefore the independence assumption made by naive Bayes helps us to estimate the required probabilities.

5. (a) The model $\mathcal{M} = \{P_1, P_2\}$, where

$$P_1(y_n = 1) = 0.9, \qquad \begin{array}{ll} P_2(y_n = 1) = 0.1 & \text{if } n \leq 3, \\ P_2(y_n = 1) = 0.9 & \text{if } n > 3. \end{array}$$

(b) Maximum likelihood would select P_2:

$$P_1(D) = (9/10)^5(1/10)^3 < (9/10)^6(1/10)^2 = P_2(D).$$

(c) MAP with the given prior would select P_1:

$$\begin{aligned}
\pi(\theta = 1 \mid D) &= \frac{P_1(D)\pi(1)}{P_{\text{Bayes}}(D)} = \frac{(9/10)^5(1/10)^3 \cdot 99/100}{P_{\text{Bayes}}(D)} \\
&= \frac{9^5 \cdot 99/10^{10}}{P_{\text{Bayes}}(D)} \\
\pi(\theta = 2 \mid D) &= \frac{P_2(D)\pi(2)}{P_{\text{Bayes}}(D)} = \frac{(9/10)^6(1/10)^2 \cdot 1/100}{P_{\text{Bayes}}(D)} = \frac{9^6/10^{10}}{P_{\text{Bayes}}(D)} \\
&< \frac{9^5 \cdot 99/10^{10}}{P_{\text{Bayes}}(D)} = \pi(\theta = 1 \mid D)
\end{aligned}$$