CS 188 Spring 2011 | Introduction to Artificial Intelligence | Practice Final Exam Solutions

Q1. [11 pts] Classification and Separating Hyperplanes

For this first part, we will be deciding what makes a good feature-mapping for different datasets, as well as finding feature weights that make the data separable.

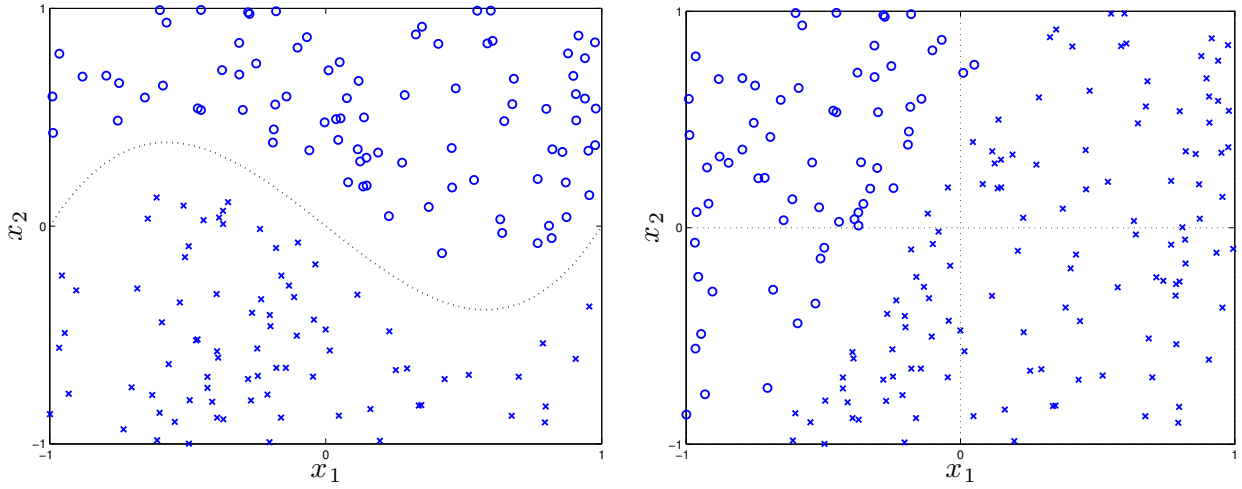


Figure 1: Sets of points separated into positive examples (x's) and negative examples (o's). In plot (a), the dotted line is given by $f(x_1) = x_1^3 - x_1$.

We begin with a series of true/false questions on what kernels can separate the datasets given. We always assume a point x is represented without a bias term, so that $x = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^\top$. We will consider the following four kernels:

A. The linear kernel $K_{\text{lin}}(x, z) = x^\top z = x \cdot z$.

B. The shifted linear kernel $K_{\text{bias}}(x, z) = 1 + x^\top z = 1 + x \cdot z$.

C. The quadratic kernel $K_{\text{quad}}(x, z) = (1 + x^\top z)^2 = (1 + x \cdot z)^2$.

D. The cubic kernel $K_{\text{cub}}(x, z) = (1 + x^\top z)^3 = (1 + x \cdot z)^3$.

(a) **(i)** [*true* or *false*] The kernel K_{lin} can separate the dataset in Fig. 1(b).

Without a bias, any linear separator must pass through the origin; by inspection, some of the x points will not be separated.

(ii) [*true* or *false*] The kernel K_{bias} can separate the dataset in Fig. 1(b).

The kernel K_{bias} is equivalent to the feature mapping $x \mapsto \phi(x) = \begin{bmatrix} 1 & x^\top \end{bmatrix}^\top$, and a linear separator with bias will handle (b) nicely.

(iii) [*true* or *false*] The kernel K_{cub} can separate the dataset in Fig. 1(b).

The quadratic and cubic kernels K_{quad} and K_{cub}, if you recall from class, have a feature mapping that includes all polynomials to degrees 2 and 3, respectively. Certainly linear classifiers are included in that set.

(iv) [*true* or *false*] The kernel K_{lin} can separate the datset in Fig. 1(a).

The separating function requires a 3rd order polynomial; perhaps more directly, by inspection any affine hyperplane must pass through some x and o points.

(v) [*true* or *false*] The kernel K_{quad} can separate the datset in Fig. 1(a).

The reasoning is the same as for part (iv).

(vi) [*true* or *false*] The kernel K_{cub} can separate the datset in Fig. 1(a).

The cubic kernel K_{cub} can represent all polynomials of degree at most 3, as we saw in class, which includes the separator given by $x_1^3 - x_1 = f(x_1) \lesseqgtr x_2$

(b) [2 pts] Now imagine that instead of simply using $x \in \mathbb{R}^2$ as input to our learning algorithm, we use a feature mapping $\phi : x \mapsto \phi(x) \in \mathbb{R}^k$, where $k \gg 2$, so that we can learn more powerful classifiers. Specifically, suppose that we use the feature mapping

$$\phi(x) = \begin{bmatrix} 1 & x_1 & x_2 & x_1^2 & x_2^2 & x_1^3 & x_2^3 \end{bmatrix}^\top \tag{1}$$

so that $\phi(x) \in \mathbb{R}^7$. Give a weight vector w that separates the x points from the o points in Fig. 1(a), that is, $w^\top \phi(x) = w \cdot \phi(x)$ should be > 0 for x points and < 0 for o points.

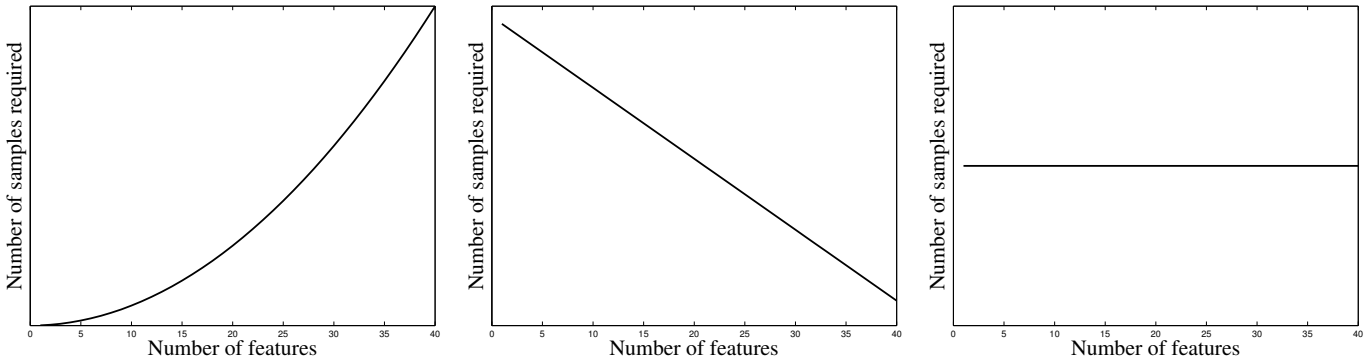
We must choose a weight vector so that $w^\top \phi(x) > 0$ when $x_1^3 - x_1 = f(x_1) > x_2$, that is, the line given by $f(x_1) = x_1^3 - x_1$ lies above the x points. We also want $w^\top \phi(x) < 0$ whenever $x_1^3 - x_1 < x_2$, i.e. for the o points in Fig. 1(a). By inspection, the weight vector $w = (0, -1, -1, 0, 0, 1, 0)$ does exactly that. That is,

$$w^\top \phi(x) = x_1^3 - x_1 - x_2 \quad \begin{cases} > 0 & \text{if } x_1^3 - x_1 > x_2 \\ < 0 & \text{if } x_1^3 - x_1 < x_2. \end{cases}$$

(c) [1 pt] Using the feature mapping (1), give a weight vector w that separates the x points from the o points in Fig. 1(b), assuming that the line given by $f(x_1) = ax_1 + b$ lies completely between the two sets of points.

The reasoning is completely similar to the above. We use the weight vector $w = (b, a, -1, 0, 0, 0, 0)$.

Now it's time to test your understanding of training error, test error, and the number of samples required to learn a classifier. Imagine you are learning a linear classifier of the form $\text{sign}(w^\top \phi(x))$, as in the binary Perceptron or SVM, and you are trying to decide how many features to use in your feature mapping $\phi(x)$.



(a) (b) (c)

Figure 2: Number of samples required to learn a linear classifier $\text{sign}(w^\top \phi(x))$ as a function of the number of features used in the feature mapping $\phi(x)$.

(d) [1 pt] Which of the plots (a), (b), and (c) in Fig. 2 is most likely to reflect the number of samples necessary to learn a classifier with good generalization properties as a function of the number of features used in $\phi(x)$?

(a)—as the number of features grows, we expect to need *more* data to train accurately (and as an aside for your edification, learning theory tells us that the growth is often roughly quadratic in the number of features).

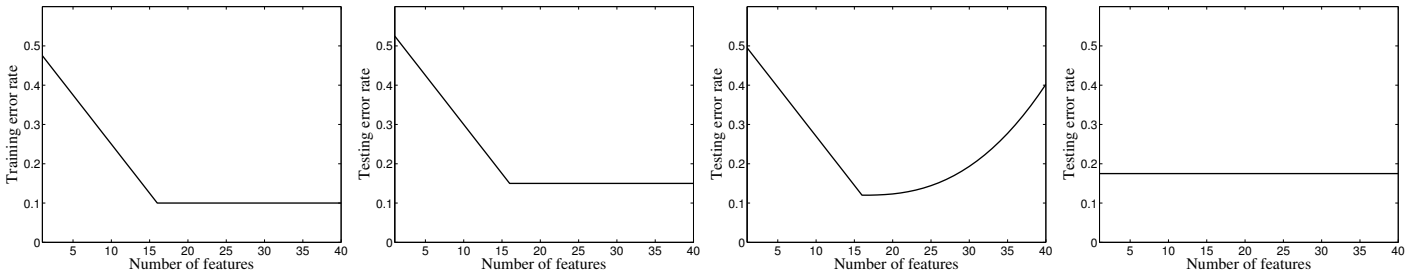


Figure 3: Leftmost plot: training error of your classifier as a function of the number of features.

(e) [1 pt] You notice in training your classifier that the training error rate you achieve, as a function of the number of features, looks like the left-most plot in Fig. 3. Which of the plots (a), (b), or (c) in Fig. 3 is most likely to reflect the error rate of your classifier on a held-out validation set (as a function of the number of features)?

(b)—Initially, as training error improves, the test error also improves (though it is slightly higher than training error). Once training error flattens, it is likely that the classifier overfits.