Homework 1 Solutions

1. *The different types of machine learning problems.* Determine whether the tasks described below involve supervised learning or unsupervised learning. For supervised learning problems, identify them as regression, classification, or probabilistic classification.

   (a) Predict the risk of an accident at an intersection, given features such as the time of day and weather.
   (b) Identify cars, bicyclists, and pedestrians in video taken by an autonomous vehicle’s cameras.
   (c) Determine the probability that there is a stop sign in an image.
   (d) Generate new road scenarios (generate streets, place stop signs and intersections) for testing autonomous vehicles in a simulation.

   **Solution.**
   
   (a) Regression
   (b) Classification
   (c) Probabilistic classification
   (d) Unsupervised learning

2. *Train vs test datasets.* Suppose you are building a classifier that identifies cats and dogs. You have a dataset of 3,000 images containing cats, dogs, or other objects (neither cat nor dog). You randomly split the data into a 2,500 image training set and a 500 image test set.

   (a) Why is it important to “reserve” some images for the test dataset? (Why shouldn’t we use all 3,000 images to train the classifier?)
   (b) After training your classifier for a while, you observe it performs well on the training images, but poorly on the test images. What is one possible explanation?

   **Solution.**
   
   (a) The test dataset should be separate from the training dataset because we want to make sure the classifier performs well on images it has never seen before.
   (b) The classifier could be overtraining. Conceptually, it has “memorized” the training dataset instead of learning features that generalize to the test dataset.

3. *Fitting a known function using samples.* In this problem you will use various nearest neighbor methods to predict \( y \in \mathbb{R} \) given \( x \in \mathbb{R} \), for a simple case in which we know the exact relation between \( x \) and \( y \). (This is never the case in practical prediction problems.)

   Consider the function \( f(x) = \sin(10x) \) over \( x \in [0, 1] \).
(a) Randomly sample 30 points $x^i$ from $[0,1]$ using a uniform distribution, and let $y^i = f(x^i)$. Plot these data points as dots, along with $f$ as a curve. (To plot $f$, evaluate it for 500 points uniformly spaced in $[0,1]$, i.e., $x = (k-1)/499$, $k = 1,\ldots,500$.)

(b) On eight separate plots, plot the $k$-nearest neighbor predictors for $k = 1,2,3$ and the soft nearest neighbor predictors for $\rho = \sqrt{0.0001}$, $\sqrt{0.0003}$, $\sqrt{0.001}$, $\sqrt{0.003}$, $\sqrt{0.01}$. Include the 30 data points, shown as dots, in these plots.

(c) RMS error. For each of the eight predictor functions in part (b), evaluate the RMS error on the 500 uniformly spaced points used to plot the functions, given by

$$\left(\frac{1}{500} \sum_{k=1}^{500} (\hat{y}_k - y_k)^2\right)^{1/2},$$

with $y_k = f((k-1)/499)$ and $\hat{y}_k = g((k-1)/499)$, where $g$ is your predictor.

**Julia hints.** `rand(N)` generates $N$ points from a uniform distribution on $[0,1]$. To generate a uniformly spaced set of $N$ values between $a$ and $b$ (with $a < b$), use `range(a, stop=b, length=N)`. To apply a function $f : \mathbb{R} \to \mathbb{R}$ elementwise to a vector $x$, use $f.(x)$.

**Solution.** The precise plots and values for this problem will vary (slightly) for each student, as each student generated their own samples of $f(x)$.

An example of a possible solution in Julia is shown below.

```julia
# plotting
import PyCall
import PyPlot; const plt = PyPlot
plt.plt.style.use("seaborn")

using Random

Random.seed!(0)

#normal knn predictor
function knn(X, Y, x, k)
    n = size(X)[1]
    #find distances of examples to x
    dists = [sum((X[i,:] .- x).^2) for i=1:n]
    #find k-nearest neighbors
    nearest_neighbor_idxs = sortperm(dists)[1:k]
    #average k-nearest neighbors
    y_hat = sum(Y[nearest_neighbor_idxs, :], dims=1)/k
    return y_hat
end
```
#soft nn predictor

```matlab
def function softnn(X, Y, x, rho)
    n = size(X)[1]
    exp_dists = [exp(-sum((X[i, :] .- x).^2)/rho) for i=1:n]
    w = exp_dists / sum(exp_dists)  #find weights
    y_hat = sum(w .* Y, dims=1)  #weighted combination of Y
    return y_hat
end
```

\[f(x) = \sin(10*x)\]

#part a

```matlab
samples = sort(rand(30))
x = [(k-1)/499 for k=1:500]
plt.close("all")
plt.scatter(samples, f.(samples), color="blue", label="samples")
plt.plot(x, f.(x), color="black", label="f(x)")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.legend()
plt.tight_layout()
plt.savefig("q2a.pdf")
```

#part b, c

```matlab
function RMSE(y, yhat)
    mse = sum((y .- yhat).^2)/length(y)
    rmse = sqrt(mse)
    return rmse
end
```

```matlab
plt.close("all")
y_samples = f.(samples)
```

#knn

```matlab
fig, ax = plt.subplots(3,1, figsize=(10,10), sharex=true)
for (j,k) in enumerate([1, 2, 3])
    Yhat = [float(knn(samples,y_samples,x[i],k))[1] for i=1:length(x)]
    ax[j].plot(x, f.(x), color="black", label="f(x)")
    ax[j].plot(x, Yhat, color="green", label=string("k-NN"))
    ax[j].scatter(samples, f.(samples), color="blue", label="samples")```
(a) The plot is shown in figure [1]

(b) The plots are shown in figures [2] and [3]
Figure 2 Plots of $k$-nearest neighbor predictors, along with $f(x)$.

Figure 3 Plots of soft nearest neighbor predictors, along with $f(x)$.  

(c) The RMS errors are reported in the title of each subplot in figures 2 and 3. The model that performed the best was the soft nearest neighbor predictor with \( \rho = 10^{-3} \), as this model yielded the smallest RMS error. (The best model could be different for each student; we gave full credit if you justified your answer properly, i.e., you chose the model with the smallest RMS error.)

4. **Polynomial embedding.** You are given raw data \((u, v)\) with \( u \in \mathbb{R}^3 \) and \( v \in \mathbb{R} \). We embed \( v \) as \( y = v \) and \( u \) as \( x = \phi(u) \). We will use a linear regression model, \( \hat{y} = x^T \theta = \phi(u)^T \theta \), with \( \theta \in \mathbb{R}^d \). Your job is to find an appropriate embedding function \( \phi : \mathbb{R}^3 \to \mathbb{R}^d \).

An expert on the data and associated application believes that a polynomial of \( u \) will give a good model of \( v \). Specifically, she believes that a good prediction model can be found as a polynomial of degree no more than 3, with degree in each component \( u_i \) no more than 2. We describe these terms below.

A polynomial of a vector \( u \in \mathbb{R}^3 \) is a linear combination of terms \( u_1^p u_2^q u_3^r \), which are called monomials, where \( p, q, \) and \( r \) are nonnegative integers, called the degree of the monomial in \( u_1, u_2, \) and \( u_3 \), respectively. The degree of the monomial \( u_1^p u_2^q u_3^r \) is \( p + q + r \). The degree of a polynomial of \( u \) is the maximum of the degrees of its monomials, and its degree in each \( u_i \) is the maximum of the degrees of its monomials in \( u_i \). For example, the polynomial \( 5.7 + u_1^2 u_2 - 3.2 u_1^3 u_2^2 u_3 + 1.3 u_3 \) has degree 6 and degree 3 in \( u_1 \), degree 2 in \( u_2 \), and degree 1 in \( u_3 \).

Suggest an appropriate embedding \( \phi \), based on the expert’s advice. **Hint:** \( d = 17 \).

**Solution.** We just enumerate all monomials that the expert has suggested are needed. There is one monomial of degree 0, namely 1. There are three monomials of degree 1: \( u_1, u_2, u_3 \). There are six monomials of degree 2:

\[
\begin{align*}
&u_1^2, \\
&u_1 u_2, \\
&u_1 u_3, \\
&u_2^2, \\
&u_2 u_3, \\
&u_3^2.
\end{align*}
\]

(All of these monomials have degree no more than 2 in each \( u_i \).) Finally, we list the monomials of degree 3 that have maximum degree 2 in each variable. There are 7 of these,

\[
\begin{align*}
&u_1 u_2 u_3, \\
&u_1 u_2^2, \\
&u_1^2 u_2, \\
&u_1 u_3^2, \\
&u_2^2 u_3, \\
&u_2 u_3^2.
\end{align*}
\]

All together there are 17 monomials of \( u \in \mathbb{R}^3 \) that have degree no more than 3 and degree in each component no more than 2. These monomials will be the components of \( \phi \).