Homework 3 Solutions

1. Feature engineering for nearest neighbor predictors. Some common feature engineering transforms have no effect for some types of predictors. Here we examine some of these.

(a) Adding a constant feature. Is it very common to add a first feature that always has the value 1. This is done with the feature engineering transform $T(x) = (1, x)$. How does this feature engineering mapping affect a $k$-nearest neighbor or soft nearest neighbor predictor? Justify your answer.

(b) Splitting into positive and negative parts. Consider the feature engineering transform $T(x) = ((x)_+, (x)_-)$, where $(x)_+ = \max\{x, 0\}$ and $(x)_- = \min\{x, 0\}$, both of these acting elementwise, i.e., separately on each entry. How does this feature engineering mapping affect a $k$-nearest neighbor or soft nearest neighbor predictor? Justify your answer.

Hint. How are $\|x - x^i\|_2$ and $\|T(x) - T(x^i)\|_2$ related?

Solution.

(a) This mapping has no effect on the predictor. Consider a point $a \in \mathbb{R}^d$. Then for all $x \in \mathbb{R}^d$, we have that

$$\|T(x^i) - T(x)\|_2 = \|(1, x^i) - (1, x)\|_2 = \|(1 - 1, x^i - x)\|_2 = \|(0, x^i - x)\|_2 = \|x^i - x\|_2.$$ 

Again, because this feature engineering transform does not affect the distances, it does not affect the $k$-nearest neighbor or soft nearest neighbor predictors.

(b) This mapping expands the size of the vector. First, we note that we have for all $x \in \mathbb{R}^d$, $x = (x)_+ + (x)_-$. But to embed the feature, we split $(x)_+$ and $(x)_-$. So $\|x\|_2 = \left\| \begin{bmatrix} (x)_+ \\ (x)_- \end{bmatrix} \right\|_2$. Now, consider a point $x^i \in \mathbb{R}^d$. For all $x \in \mathbb{R}^d$, we have

$$\|T(x^i) - T(x)\|_2 = \left\| \begin{bmatrix} (x^i)_+ \\ (x^i)_- \end{bmatrix} - \begin{bmatrix} (x)_+ \\ (x)_- \end{bmatrix} \right\|_2$$

If the components of $x^i$ and $x$ have the same sign (for example, $x_i = [1, -1]^\top$ and $x = [2, -2]^\top$ then this mapping has no effect. We would embed this example as $T(x^i) = [1, 0, 0, -1]^\top$ and $T(x) = [2, 0, 0, -2]^\top$. Then the distance $\|T(x^i) - T(x)\|_2$ is:

$$\|T(x^i) - T(x)\|_2 = \sqrt{(1 - 2)^2 + (-1 + 2)^2} = \sqrt{2}$$

Observe that:

$$\|x^i - x\|_2 = \sqrt{(1 - 2)^2 + (-1 + 2)^2} = \sqrt{2}$$
If the signs were different, for example $x = [-2, 2]^T$, this would not be true. The distances would be different. However, one can observe that if we use the $\|\cdot\|_1$ norm instead of $\|\cdot\|_2$, we get the same distance and nothing will change.

In general, depending on the definition of the distance used in the algorithm, the distance may change or remain constant, but the general performance will not be affected in either case.

Note: In the EE104 slides, $(x)_-$ is negative. But some people define $(x)_-$ as a positive number: $\max\{-x, 0\}$. It is OK if you did this.

2. *All-pairs interactions*. The following problem will use $U$, and $v$ found in `all_pairs_data.json`. The data has $U \in \mathbb{R}^{n \times 3}$. Throughout this problem, use a 50-50 train/test split.

(a) Fit a linear least-squares model directly to the data matrix, with the first feature being a constant feature $x_1 = 1$. Since we’ve given you enough data and the data is approximately standardized, you do not have to worry about regularization or standardization.

The linear least-squares model is:

$$\sum_{i=1}^{N} (\theta^T u^{(i)} - v^{(i)})^2$$

If we define $v$ as the vector of $v^{(i)}$'s and $U$ as a matrix of $u^{(i)}$'s (every row of $U$ is one observation $u^{(i)}$), we can rewrite this in matrix form:

$$\|U\theta - v\|_2^2$$

Report the train and test RMSE of this predictor.

*Hint:* In Julia, you can compute the least-squares solution as `theta = U \ v`.

*Hint:* In this case, it is fine to split the data in half naively, using the first half as the train dataset and the second half as the test dataset (or vice-versa).

(b) Create an embedding which includes all of the interactions (products) between every pair of distinct variables, along with a constant feature and the variables themselves. For example, if $u \in \mathbb{R}^2$, then the embedding should be

$$\phi(u) = (1, u_1, u_2, u_1u_2).$$

You’ll have to construct a new data matrix. This data matrix should include a column of ones and $d$ columns for $u_1, \ldots, u_d$. The remaining columns will contain the products $u_1^2$, $u_1u_2, \ldots, u_3u_2$, $u_3^2$.

Fit a new $\theta$ to this data matrix. Report the train and test RMSE of this predictor. Compare it with the RMSEs you got in (a).

(c) Is it necessary to use all of the features? Print the $\theta$ you found in part (b) and inspect it. Remember $\theta$ represents the weights assigned to each feature. Are there any features that aren’t being used in the predictor? Remove them. (You’ll have to construct a smaller data matrix - see the hint.)
Report which indices $\theta_i$ you removed. Remember that in Julia, vectors are 1-indexed, so $\theta = [\theta_1, \theta_2, \ldots, \theta_{13}]^\top$.

Fit a new $\theta$ with the unnecessary features removed and report the train and test RMSE of this predictor. Compare it with the RMSEs you got in (a) and (b).

**Hint:** To remove column $i$ from a matrix $U$ in Julia, you can use array slicing:

$$U = [U[:,1:i-1] \ U[:,i+1:end]]$$

Be careful to slice the columns (second index) and not the rows. In the above line, we use : to index all the rows.

**Hint:** It is perfectly fine to hard-code the array slicing: if you decide to remove index 9, you may hard-code the value 9.

**Hint:** You should remove at least 3 features. You may remove more if you like.

**Solution.**

(a) We received a training RMSE of around 1.048 and a test RMSE of around 1.013 for this predictor.

(b) For the all-pairs predictor we received a training RMSE of around 0.097 and a test RMSE of around 0.097. The code for this problem is found below.

(c) There are 3 theta-values that are exactly 0: indices 8, 11, and 12. If the student removes exactly these 3 values, the training RMSE and test RMSE should remain around 0.097. Indices 13 and 3 are also very small and the student may choose to remove these (or others). The RMSE changes very slightly, but is still around 0.097 for train and test.

```julia
include("readclassjson.jl")

all_data = readclassjson("all_pairs_data.json")
U = all_data["U"]
v = all_data["v"]

n = size(U, 1)
d = size(U, 2)

n_train = div(n, 2)
n_test = div(n, 2)

v_train = v[1:n_train]
v_test = v[n_train+1:end]

X_simple = [ones(n) U]

X_simple_train = X_simple[1:n_train,:]
X_simple_test = X_simple[n_train+1:end,:]```
theta_simple = X_simple_train \ v_train

simple_RMSE_train = norm(X_simple_train * theta_simple - v_train)/sqrt(n_train)
simple_RMSE_test = norm(X_simple_test * theta_simple - v_test)/sqrt(n_test)

println("Simple train RMSE $(simple_RMSE_train)")
println("Simple test RMSE $(simple_RMSE_test)")

X_all_pairs = ones(n,1)

for i = 1:d
    for j = i:d
        if i==j
            X_all_pairs = [X_all_pairs U[:,i]]
            continue
        end
        X_all_pairs = [X_all_pairs U[:,i].*U[:,j]]
    end
end

X_all_pairs_train = X_all_pairs[1:n_train, :]
X_all_pairs_test = X_all_pairs[n_train+1:end, :]

theta_pairs = X_all_pairs_train \ v_train

pairs_RMSE_train = norm(X_all_pairs_train * theta_pairs - v_train)/sqrt(n)
pairs_RMSE_test = norm(X_all_pairs_test * theta_pairs - v_test)/sqrt(n)

println()
println("Pairs train RMSE $(pairs_RMSE_train)")
println("Pairs test RMSE $(pairs_RMSE_test)")

3. Feature engineering. In each of the parts (a)-(i), you are given a set of $N = 1000$ raw data points $(u, v)$, where $u \in \mathbb{R}^2$ and $v \in \{\text{Red}, \text{Blue}\}$. We will be using the raw output feature map

$$\psi(v) = \begin{cases} +1 & v = \text{Red} \\ -1 & v = \text{Blue} \end{cases}$$
and we will use a linear classification predictor with the form

\[ \hat{y} = \begin{cases} +1 & \theta^\top x \geq 0 \\ -1 & \theta^\top x < 0 \end{cases} \]

where we will find \( \theta \) by training a machine learning algorithm over the data. For each part, introduce a feature mapping \( \phi \) for the input data, i.e., \( x = \phi(u) \), such that if we find the best possible \( \theta \), then the RMS error

\[ \left( \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - \bar{y}_i)^2 \right)^{1/2} \]

is almost zero. For each part, give a brief justification of why you chose a specific \( \phi \).

**Solution.**

(a) \( \phi(u) = (1, u_1, u_2) \)

(b) \( \phi(u) = (1, u_1, \sin(ku_2)) \) for some \( k > 1 \).

(c) \( \phi(u) = (1, u_1^3, u_2) \)

(d) \( \phi(u) = (1, u_1u_2, u_1^2, u_2^2) \)

(e) \( \phi(u) = (1, |u_1|, |u_2|) \)

(f) \( \phi(u) = (1, |u_1|) \) or \( (1, u_1^2) \)

(g) \( \phi(u) = (u_1u_2) \)

(h) \( \phi(u) = (|u_1|, |u_2|) \) or \( (u_1^2, u_2^2) \)

Note that there might be other correct answers for each part. Also, note that the above solutions are minimal, and any other answer \( \phi'(u) \) that includes all elements of \( \phi(u) \) and possibly some more entries is also correct.