CS 189 Introduction to Machine Learning
Fall 2017

HW11

This homework is due **Monday, November 13 at 10pm.**

1 Getting Started

You may typeset your homework in latex or submit neatly handwritten and scanned solutions. Please make sure to start each question on a new page, as grading (with Gradescope) is much easier that way! Deliverables:

1. Submit a PDF of your writeup to assignment on Gradescope, “HW[n] Write-Up”
2. Submit all code needed to reproduce your results, “HW[n] Code”.
3. Submit your test set evaluation results, “HW[n] Test Set”.

After you’ve submitted your homework, be sure to watch out for the self-grade form.

(a) Before you start your homework, write down your team. Who else did you work with on this homework? List names and email addresses. In case of course events, just describe the group. How did you work on this homework? Any comments about the homework?

(b) Please copy the following statement and sign next to it:

I certify that all solutions are entirely in my words and that I have not looked at another student’s solutions. I have credited all external sources in this write up.

2 Kernel SVM and Kernel Ridge Regression

In lecture, kernels were discussed in a way that avoided formality.

In this problem, we will give a derivation of kernel SVM and kernel ridge regression from the view of function approximation with pernalization. The objective function of a linear SVM can be interpreted as Hinge Loss combined with L_2 regularization over the space of linear functions. The objective function of a kernel SVM can be interpreted in the same way: Hinge Loss plus L_2 regularization over the Hilbert space of functions defined by a kernel function.

Assume we are doing classification or regression over $\mathbb{R}^d$. We first introduce the following abstract vector space:

$$H = \{f : \mathbb{R}^d \to \mathbb{R} : f(x) = \sum_{m=1}^{M} \alpha_m k(x, y_m) : \alpha_i \in \mathbb{R}, M \in \mathbb{N}, y_m \in \mathbb{R}^d\}, \tag{1}$$

where $k(x,y) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a kernel function that satisfies the property $k(x,y) = k(y,x)$ for any $x, y \in \mathbb{R}^d$ and for any distinct $y_1, y_2, \ldots, y_M \in \mathbb{R}^d$, the matrix $K \in \mathbb{R}^{M \times M}$ defined by $K_{ij} = k(y_i, y_j)$ is positive definite.

This is a vector space since it has a zero and linear combinations make sense. It can be infinite-dimensional however since there are lots of possible y_m.

(a) Now we introduce an inner product on the above vector space H. We define the inner product between any two functions

$$f(x) = \sum_{m=1}^{M} \alpha_m k(x, y_m), g(x) = \sum_{s=1}^{S} \beta_s k(x, x_s) \in H,$$

in H as

$$\langle f, g \rangle_H = \sum_{m=1}^{M} \sum_{s=1}^{S} \alpha_m \beta_s k(y_m, x_s). \tag{2}$$

Show that the defined inner product is valid. That is, it satisfies the symmetry, linearity and positive-definiteness properties stated below: For any for any $f, g, h \in H$ and any $a \in \mathbb{R}$, we have

- $\langle f, g \rangle_H = \langle g, f \rangle_H$.
- $\langle af, g \rangle_H = a\langle f, g \rangle_H$ and $\langle f + h, g \rangle_H = \langle f, g \rangle_H + \langle h, g \rangle_H$.
- $\langle f, f \rangle_H \geq 0$; $\langle f, f \rangle = 0$ if and only if f is an constant zero function.

What is the norm of the function f? (The natural norm in an inner product space is defined as $\|f\|_H = \sqrt{\langle f, f \rangle_H}$.)

Solution:

$$\begin{aligned}\langle f,g\rangle_H &= \sum_{m=1}^{M}\sum_{s=1}^{S}\alpha_m\beta_s k(y_m,x_s)\\ &= \sum_{s=1}^{S}\sum_{m=1}^{M}\beta_s\alpha_m k(x_s,y_m)\\ &= \langle g,f\rangle_H,\end{aligned}$$

where the second equation follows from the symmetry of kernel function.

$$\begin{aligned}\langle af,g\rangle_H &= \sum_{m=1}^{M}\sum_{s=1}^{S}(a\alpha_m)\beta_s k(y_m,x_s)\\ &= a\sum_{m=1}^{M}\sum_{s=1}^{S}\alpha_m\beta_s k(y_m,x_s)\\ &= a\langle f,g\rangle_H,\end{aligned}$$

We write h as $h(x)=\sum_{t=1}^{N}\gamma_t k(x,z_t)$. Then

$$(f+h)(x)=\sum_{m=1}^{M}\alpha_m k(x,y_m)+\sum_{t=1}^{N}k(x,z_t).$$

$$\begin{aligned}\langle f+h,g\rangle_H &= \sum_{m=1}^{M}\sum_{s=1}^{S}\alpha_m\beta_s k(y_m,x_s)+\sum_{t=1}^{T}\sum_{s=1}^{S}\gamma_t\beta_s k(z_t,x_s)\\ &= \langle f,g\rangle_H+\langle h,g\rangle_H,\end{aligned}$$

where the first equality follows directly from the definition of the inner product.

Without loss of generality, we assume x_i are distinct. Then

$$\begin{aligned}\langle f,f\rangle_H &= \sum_{m=1}^{M}\sum_{s=1}^{M}\alpha_m\alpha_s k(y_m,y_s)\\ &= \alpha^T K\alpha \geq 0,\end{aligned}$$

where $\alpha\in\mathbb{R}^n$ with ith element being α_i and $K\in\mathbb{R}^{n\times n}$ with i,jth element being $k(y_i,y_j)$. Because K is positive definite, the last inequality follows. And the equality holds if and only if $\alpha=0$, which is equivalent to f is a constant zero function.

The norm of a function f with $f(x)=\sum_{m=1}^{M}\alpha_m k(x,y_m)$ is

$$\|f\|_H=\sqrt{\langle f,f\rangle_H}=\sqrt{\sum_{m=1}^{M}\sum_{s=1}^{M}\alpha_m\alpha_s k(y_m,y_s)}=\sqrt{\alpha^T K\alpha}. \tag{3}$$

(b) **Show that the defined inner product has the reproducing property** $\langle k(x,\cdot), k(y,\cdot)\rangle_H = k(x,y)$, where we take $\cdot \to k(x,\cdot)$ and $\cdot \to k(y,\cdot)$ as two functions in H. In other words, the inner-product is natural for the vector space as defined. **Conclude that**

$$\langle k(\cdot,x_i), f\rangle_H = f(x_i). \tag{4}$$

(For those who have taken signal processing courses, you can see here that what this family of definitions is trying to do is to parallel the example of the sifting property that we know from signal processing.) **Solution:** We show the reproducing property: $\langle k(x,\cdot), k(y,\cdot)\rangle_H = k(x,y)$. In this case, we let $f : z \to k(x,z)$ and $g : z \to k(y,z)$. The reproducing property follows from the definition of inner product directly.

For any f in H, write f as $f(x) = \sum_{m=1}^{M} \alpha_m k(x, y_m)$. Then

$$\begin{aligned}
\langle k(\cdot,x_i), f\rangle_H &= \langle k(\cdot,x_i), \sum_{m=1}^{M} \alpha_m k(\cdot, y_m)\rangle_H \\
&= \sum_{m=1}^{M} \alpha_m \langle k(\cdot,x_i), k(\cdot,y_m)\rangle_H \\
&= \sum_{m=1}^{M} \alpha_m k(y_m, x_i) \\
&= f(x_i),
\end{aligned}$$

where the third equality follows from the reproducing property.

(c) The completion of this inner product space is a Hilbert space called a Reproducing Kernel Hilbert Space (RKHS), but in this problem, this knowledge is not required. From now on, we will work in the completion of H. We will also call it H for notational simplicity. We assume we have the following knowledge of a Hilbert space: Suppose M is a finite-dimensional subspace of a Hilbert space H. Then any element f in the full space H has a unique representation as the sum of an element of M and an element that is orthogonal to any element in M. That is,

$$f = m + g, \tag{5}$$

for some $m \in M$ and some g such that $\langle m', g\rangle_H = 0$ for all $m' \in M$. (In other words, it behaves exactly like the vector spaces that you are used to.)

Below we introduce a general optimization problem over the Hilbert space H. We will see many kernalized machine learning algorithms, including kernel SVM, can be written in the following form: Given a data set with N points $x_i, y_i, i = 1, \dots, N$, the optimization problem is

$$\min_{f\in H} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda \|f\|_H^2, \tag{6}$$

where L is any loss function on pairs of real numbers. (Remember, the y_i here in this part are real numbers. They are not training points in d-dimensional space. Those are the x_i.)

Show that the minimizing solution to the problem has the form

$$f(x) = \sum_{i=1}^{N} \alpha_i k(x, x_i).$$

That is, the soluton can be expressed as a weighted sum of kernel functions based on the training points. This phenomenon that reduces an infinite-dimensional optimization problem to be finite-dimensional is called the *kernel property*. Hint: Define $M = \{\sum_{n=1}^{N} \alpha_n k(x, x_n) : \alpha_i \in \mathbb{R}\}$ to be the subspace of interest.

Solution: For any $f \in H$, by the theorem of orthogonal compliment mentioned above, we can decompose f as

$$f(x) = \sum_{i=1}^{N} \alpha_i k(x, x_i) + g(x), \tag{7}$$

for some function g that is orthogonal to any element in $M = \{\sum_{n=1}^{N} \alpha_n k(x, x_n) : \alpha_i \in \mathbb{R}\}$. From Part (b), we have

$$\begin{aligned} f(x_j) &= \sum_{i=1}^{N} \alpha_j k(x_j, x_i) + g(x_j) \\ &= \sum_{i=1}^{N} \alpha_j k(x_j, x_i) + \langle g, k(\cdot, x_j) \rangle_H \\ &= \sum_{i=1}^{N} \alpha_j k(x_j, x_i), \end{aligned}$$

where the second equality follows from the reproducing property and the third equality follows from the orthogonality. Define $\tilde{f}(x) = \sum_{i=1}^{N} \alpha_i k(x, x_i)$. Then we have $f = \tilde{f} + g$ and $L(y_i, f(x_i)) = L(y_i, \tilde{f}(x_i))$. On the other hand, we have

$$\begin{aligned} \|f\|_H^2 &= \|\tilde{f}\|_H^2 + \|g\|_H^2 + 2\langle \tilde{f}, g \rangle \\ &= \|\tilde{f}\|_H^2 + \|g\|_H^2 \\ &\geq \|\tilde{f}\|_H^2, \end{aligned}$$

where the second equality follows from the orthogonality and the third inequality follows from the nonnegativeness of the norm. Therefore,

$$\frac{1}{N} \sum_{i=1}^{N} L(y_i, \tilde{f}(x_i)) + \lambda \|\tilde{f}\|_H^2 \leq \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda \|f\|_H^2. \tag{8}$$

The optimization problem

$$\min_{f \in H} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda \|f\|_H^2, \tag{9}$$

is therefore equivalent to

$$\min_{f \in M} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda \|f\|_H^2, \tag{10}$$

which means the solution f lies in M and is of the form

$$f(x) = \sum_{i=1}^{N} \alpha_i k(x, x_i). \tag{11}$$

(d) (Kernel SVM) The kernel SVM, is nothing but defining the loss function L concretely as a Hinge loss:

$$L(y, f(x)) = \max(0, 1 - y f(x)). \tag{12}$$

In other words, kernel SVM is

$$\min_{f \in H} \frac{1}{N} \sum_{i=1}^{N} \max(0, 1 - y_i f(x_i)) + \lambda \|f\|_H^2. \tag{13}$$

Show kernel SVM is of the form

$$\min_{\alpha \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} \max(0, 1 - y_i \sum_{j=1}^{N} \alpha_j k(x_i, x_j)) + \lambda \alpha^T K \alpha. \tag{14}$$

Solution: Any $f \in M$ can be expressed as $f(x) = \sum_{i=1}^{N} \alpha_i k(x, x_i)$ for some $\alpha \in \mathbb{R}^d$. Then $\|f\|_H^2 = \alpha^T K \alpha$ from Part (a).

The optimization problem

$$\min_{f \in M} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda \|f\|_H^2, \tag{15}$$

can be rewriten as

$$\min_{\alpha \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} L(y_i, \sum_{j=1}^{N} \alpha_j k(x_i, x_j)) + \lambda \alpha^T K \alpha. \tag{16}$$

Write out L specifically as a Hinge loss, we get the kernel SVM:

$$\min_{\alpha \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} \max(0, 1 - y_i \sum_{j=1}^{N} \alpha_j k(x_i, x_j)) + \lambda \alpha^T K \alpha. \tag{17}$$

(e) (Kernel ridge regression) Take L as l_2 loss, that is, $L(a, b) := \|a - b\|_2^2$. **Show that optimization problem of kernel ridge regression has the following form**

$$\min_{\alpha \in \mathbb{R}^d} \frac{1}{N} \|Y - K\alpha\|_2^2 + \lambda \alpha^T K \alpha, \tag{18}$$

where $Y \in \mathbb{R}^n$ with the ith element of Y being y_i. **Derive a closed form solution for the kernel ridge regression:**

$$\alpha = (K + \lambda N I_N)^{-1} Y, \tag{19}$$

where I_N is an n-dimensional identity matrix.

Solution: Plugging L with the l_2 loss, we get

$$\begin{aligned}\frac{1}{N}\sum_{i=1}^{N} L(y_i, \sum_{j=1}^{N} \alpha_j k(x_i,x_j)) &= \frac{1}{N}\sum_{i=1}^{N}(y_i - \sum_{j=1}^{N}\alpha_j k(x_i,x_j)))^2 \\ &= \frac{1}{N}\|K\alpha - Y\|^2\end{aligned}$$

Then the optimization problem can be written as

$$\min_{\alpha\in\mathbb{R}^d} \frac{1}{N}\|K\alpha - Y\|^2 + \lambda\alpha^T K\alpha. \tag{20}$$

Write $F(\alpha) = \frac{1}{N}\|K\alpha - Y\|^2 + \lambda\alpha^T K\alpha$. Then

$$\begin{aligned}\nabla_\alpha F(\alpha) &= \nabla_\alpha \frac{1}{N}(\alpha^T K^2\alpha - 2Y^T K\alpha + Y^T Y) + \nabla_\alpha(\alpha^T K\alpha) \\ &= \frac{2}{N}(K^2\alpha - KY) + 2\lambda K\alpha \\ &= 2(\frac{1}{N}K^2 + \lambda K)\alpha - \frac{2}{N}KY.\end{aligned}$$

The Hessian is $2(\frac{1}{N}K^2 + \lambda K)$, which is positive definite. Therefore, the optimum is obtained when $\nabla_\alpha F(\alpha) = 0$. That is

$$\alpha = (K + \lambda N I_N)^{-1} Y. \tag{21}$$

(f) (Polynomial Regression from a kernelized view) In this part, we will show that polynomial regression with a particular Tiknov regularization is the same as kernel ridge regression with a polynomial kernel for second-order polynomials. Recall that a polynomial kernel function on $\mathbb{R}^d$ is defined as

$$k(x,y) = (1 + x^T y)^2, \tag{22}$$

for any $x, y \in \mathbb{R}^d$. Given a dataset (x_i, y_i) for $i = 1, 2, \ldots, N$. **Show the solution to kernel ridge regression is the same as the least square solution to polynomial regression for** $d = 2$ **given the right choice of Tikhonov regularization for the polynomial regression.** That is, show for any new point x given in the prediction stage, both methods give the same y. What is the Tikhonov regularization matrix here?

Hint: You may or may not use the following matrix identity:

$$A(aI_d + A^T A)^{-1} = (aI_N + AA^T)^{-1} A, \tag{23}$$

for any matrix $A \in \mathbb{R}^{n\times d}$ and any positive real number a.

Solution: Define a vector-valued function from $\mathbb{R}^2 \to \mathbb{R}^6$ such that

$$\phi(a) = (1, a_1^2, a_2^2, \sqrt{2}a_1, \sqrt{2}a_2, \sqrt{2}a_1 a_2)^T$$

for $a = (a_1, a_2)^T$.

Define a matrix in $\mathbb{R}^{N\times 6}$ such that

$$\Phi = \begin{bmatrix} \phi(x_1)^T \\ \phi(x_2)^T \\ \vdots \\ \phi(x_N)^T \end{bmatrix} \tag{24}$$

We observe that $k(x,y) = \phi(x)^T\phi(y)$. For a kernel matrix $K \in \mathbb{R}^{n\times n}$ with $K_{ij} = k(x_i,x_j)$, we have

$$K_{ij} = \phi(x_i)^T\phi(x_j) = (\Phi\Phi^T)_{ij}. \tag{25}$$

That is

$$K = \Phi\Phi^T.$$

Recall the solution to Kernel ridge regression is a function f with

$$\begin{aligned} f(x) &= \sum_{i=1}^{N} \alpha_i k(x_i,x) \\ &= \sum_{i=1}^{N} \alpha_i \phi(x_i)^T\phi(x) \\ &= \alpha^T\Phi\phi(x), \end{aligned}$$

where

$$\alpha = (K+\lambda N I_N)^{-1}Y. \tag{26}$$

Therefore, we can write $f(x)$ as

$$f(x) = Y^T(\Phi\Phi^T + \lambda N I_N)^{-1}\Phi\phi(x). \tag{27}$$

For polynomial regression, define a vector-valued function from $\mathbb{R}^2 \to \mathbb{R}^6$ such that

$$\tilde{\phi}(a) = (1, a_1^2, a_2^2, a_1, a_2, a_1a_2)^T$$

for $a = (a_1,a_2)^T$.

Define a matrix in $\mathbb{R}^{N\times 6}$ such that

$$\tilde{\Phi} = \begin{bmatrix} \tilde{\phi}(x_1)^T \\ \tilde{\phi}(x_2)^T \\ \vdots \\ \tilde{\phi}(x_N)^T \end{bmatrix} \tag{28}$$

Observe the relationship between ϕ and $\tilde{\phi}$: We have

$$\tilde{\phi}(x) = D\phi(x), \tilde{\Phi} = \Phi D, \tag{29}$$

for a diagonal matrix $D \in \mathbb{R}^{6\times 6}$, with

$$D = \operatorname{diag}(1,1,1,1/\sqrt{2},1/\sqrt{2},1/\sqrt{2}).$$

A polynomial regression is nothing but replacing linear feature X by $\tilde{\phi}(X) \in \mathbb{R}^6$ and add a Tikhonov regularization over the parameters $w \in \mathbb{R}^6$. Recall in a previous homework, we've shown it has a closed form solution

$$w = (\tilde{\Phi}^T \tilde{\Phi} + \Lambda)^{-1} \tilde{\Phi}^T Y, \tag{30}$$

for a polynomial regression with Tikhonov regularization matrix $\Lambda \in \mathbb{R}^{d \times d}$. Let Λ be a diagonal matrix defined by

$$\Lambda = \text{diag}(\lambda N, \lambda N, \lambda N, \lambda N/2, \lambda N/2, \lambda N/2). \tag{31}$$

Then

$$\Lambda = D(\lambda N I_6) D. \tag{32}$$

Then the predictor produced by Tikhonov regression is

$$\begin{aligned}
g(x) &= w^T \tilde{\phi}(x) \\
&= [(\tilde{\Phi}^T \tilde{\Phi} + \Lambda)^{-1} \tilde{\Phi}^T Y]^T \tilde{\phi}(x) \\
&= Y^T \tilde{\Phi} (\tilde{\Phi}^T \tilde{\Phi} + \Lambda)^{-1} \tilde{\phi}(x) \\
&= Y^T \Phi D (D \Phi^T \Phi D + D(D^{-1} \Lambda D^{-1}) D)^{-1} D \phi(x) \\
&= Y^T \Phi D D^{-1} (\Phi^T \Phi + (D^{-1} \Lambda D^{-1}))^{-1} D^{-1} D \phi(x) \\
&= Y^T \Phi (\Phi^T \Phi + (D^{-1} \Lambda D^{-1}))^{-1} \phi(x) \\
&= Y^T \Phi (\Phi^T \Phi + \lambda N I_6)^{-1} \phi(x) \\
&= Y^T (\Phi \Phi^T + \lambda N I_N)^{-1} \Phi \phi(x) = f(x),
\end{aligned}$$

where the last equation follows from the hint. Hence, we have shown the equivalence between the two predictors.

(g) (Bonus) In general, for any polynomial regression with pth order polynomial on $\mathbb{R}^d$, with a selected Tikhonov regression, we can show the equivalence between it and kernel ridge regression with a polynomial kernel of order p. (You are not required to show this.) **Comment on the computational complexity of doing least squares for polynomial regression with a Tikhonov regression directly and that of doing kernel ridge regression in the training stage.** (That is, the complexity of finding α and finding w.) **Compare with the computational complexity of actually doing prediction as well.**

Solution: In the polynomial regression with Tikhonov regularization, for any data point (x_i, y_i), computing its polynomial features of order p takes $O(d^p)$. The complexity of solving least square is $O(d^{3p} + d^p n)$. The total complexity is $O(d^{3p} + d^p n)$.

In the kernel ridge regression, the complexity of computing the kernel matrix is $O(n^2 d \log p)$. The complexity of getting α after that is $O(n^3)$. The total complexity is $O(n^3 + n^2 d \log p)$. It only has a log dependence on p but cubic dependence on n. Kernel ridge regression is preferred when p is large.

3 Nearest Neighbors, from A to Z

For this problem, we will use data from the UN to have some fun with the nearest neighbors algorithm. A lot of the code you will need has been provided for you.

The data we are using is called the "World Values Survey." It consists of survey data collection over several years from almost all countries. The survey asked "Which of these are most important for you and your family?" There were 16 possible responses, including needs like "Freedom from Discrimination and Persecution" and "Better Transport and Roads." The data reported is the fraction of responses in each country that chose each option.

We would like to use these 16 features of each country (the citizen's responses to the survey) to predict that country's HDI (Human Development Index). In reality, the HDI is a complex measure which takes into account lots of data about a country, including factors like life expectancy, education, per capita income, etc. Intuitively, though, you would expect citizens of countries with different HDI to have different priorities. For that reason, predicting the HDI from survey data might be a reasonable endeavor.

Note that throughout the problem we will be using RMSE, which stands for Root Mean Squared Error.

(a) (Bonus): **Fill out the "Berkeley F2017 Values Survey."** The purpose of this is so that you have a sense of how the data was generated, a useful first step in any ML problem. Just for fun, at the end of this problem we will attempt to predict what the HDI of Berkeley would be if it were its own country.

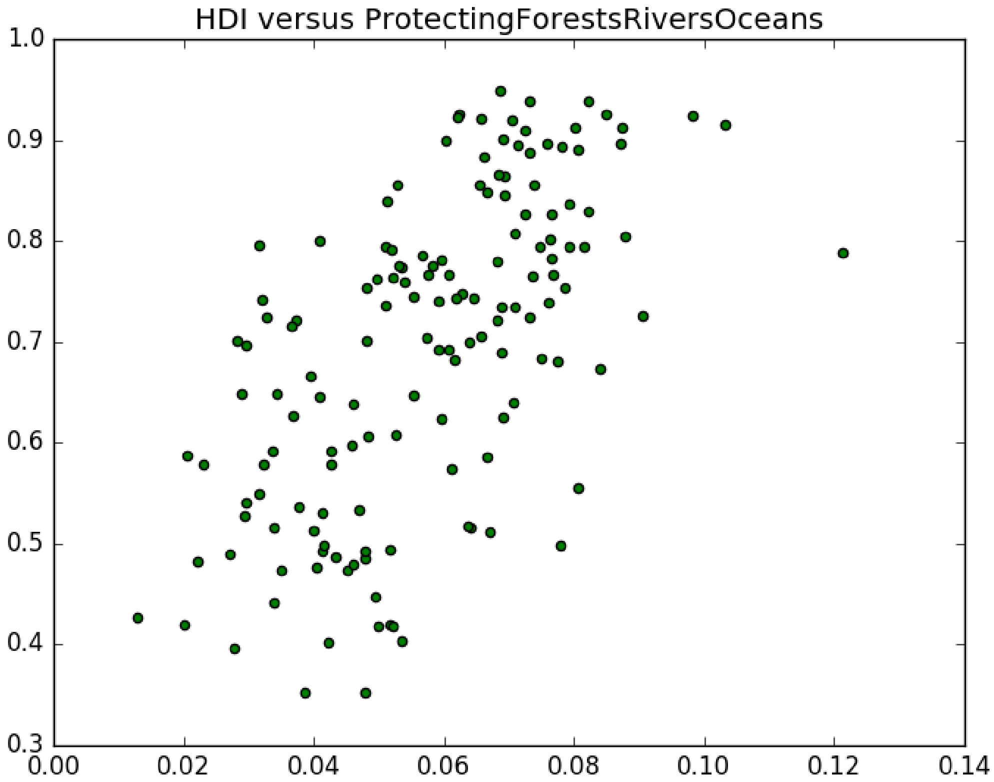
Solution: N/A

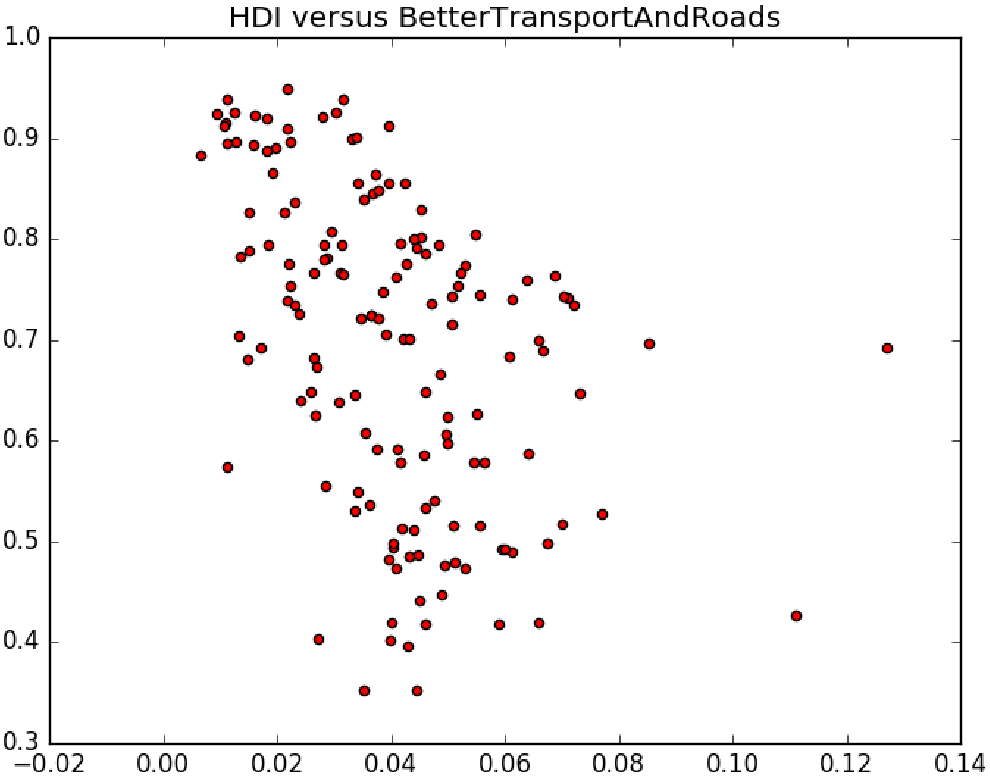
(b) First, we should do some basic data exploration. **Compute the correlation of each feature with HDI. Which feature is the most positively correlated with HDI? Which feature is the most negatively correlated with HDI? Which feature is the least correlated with HDI (closest to** 0**)?**

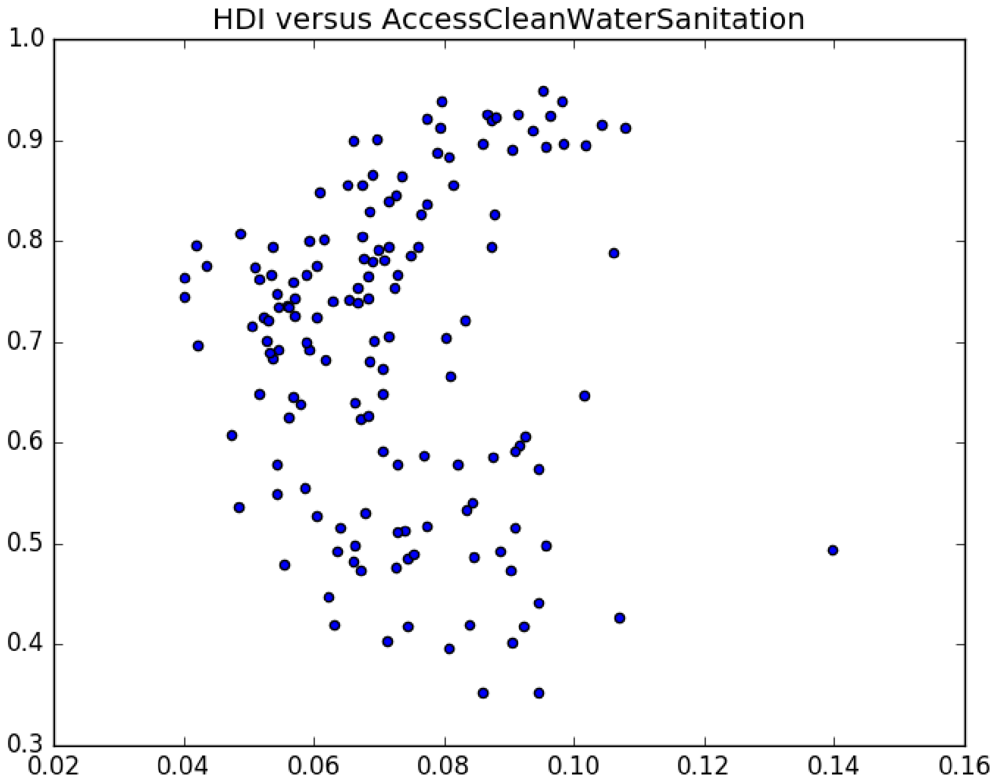
Solution: Run the code provided. The most positively correlated is "Protecting forests rivers and oceans" ($\rho = 0.61$). The most negatively correlated is "Better transport and roads" ($\rho = -0.44$). This is arguably logical, since countries with higher HDI are less likely to be concerned with basic needs. The least correlated is "Access to clean water and sanitation" ($\rho = -0.02$).

(c) **For each of these three features identified in (b) (most positively correlated, most negatively correlated, least correlated), plot "HDI versus [Feature]."** You will create three plots in total. **What do you observe?**

Solution: Run the code provided to see the graphs below. Interestingly, looking at the graph, it appears that the response "Access to clean water and sanitation." was chosen more by countries with very low HDI and very high HDI, but not "middle" HDI. This suggest that we will ultimately want to use features which are more sophisticated than linear.

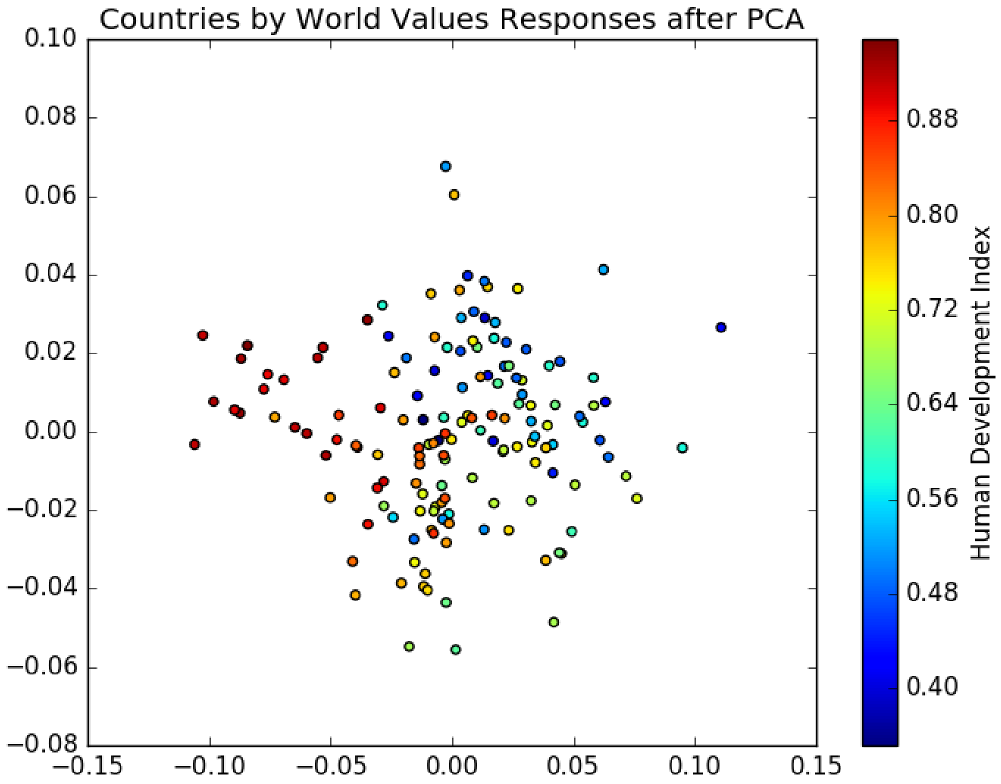

HDI versus ProtectingForestsRiversOceans
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14


HDI versus BetterTransportAndRoads
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
−0.02
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14



(d) Let's visualize the data a bit more. **Plot the data in its first two PCA dimensions, colored by HDI.** The code to do this has been provided for you.

Solution: Run the code provided to see the graph below:



(e) Now, let's use our first ML technique. **Use the code provided to train and cross-validate ridge regression to predict a country's HDI from its citizens' world values survey responses.** You may need to modify the hyper-parameters. **What is the best RMSE?**

Solution: Run the code provided to get an RMSE 0.123. Your values may differ slightly based on your hyper-parameters and randomness. If your value is above 0.13, you probably made a mistake.

(f) Let's try another ML technique. **Use the code provided to train and cross-validate lasso regression to predict a country's HDI from its citizens' world values survey responses.** You may need to modify the hyper-parameters. **What is the best RMSE?**

Solution: Run the code provided to get an RMSE of 0.124. Your values may differ slightly based on your hyper-parameters and randomness. If your value is above 0.13, you probably made a mistake.

(g) **Examine the model returned by lasso regression (that is, the** 16 **feature weights). Does lasso regression indeed give more** 0 **weights?**

Solution: There are a few interesting observations you could make:

- There are lots of 0 weights (about half the weights are 0).
- The signs of the weights agree with the signs of the correlations found earlier.

- A lot of the weight is going towards the most correlated features (the weight is more than 3, while the others are less than 1 in absolute value).

(h) In lecture, we covered k Nearest Neighbors for classification problems. We decided that the class of a test point would be the plurality of the classes of the k nearest training points. That algorithm makes sense when the outputs are discrete, so we can vote. Here, the outputs are continuous. **How would you adapt the k Nearest Neighbors algorithm for a regression problem?**

Solution: There are many possible answers. We could take the *average* of the labels of the neighbors or the *weighted average* (weighted by distance) or the *median*. Basically, any measure of central tendency.

(i) **Which countries are the 7 nearest neighbors of the USA (in order)?**

Solution: In order, they are:

(a) Ireland

(b) United Kingdom

(c) Belgium

(d) Finland

(e) Malta

(f) Austria

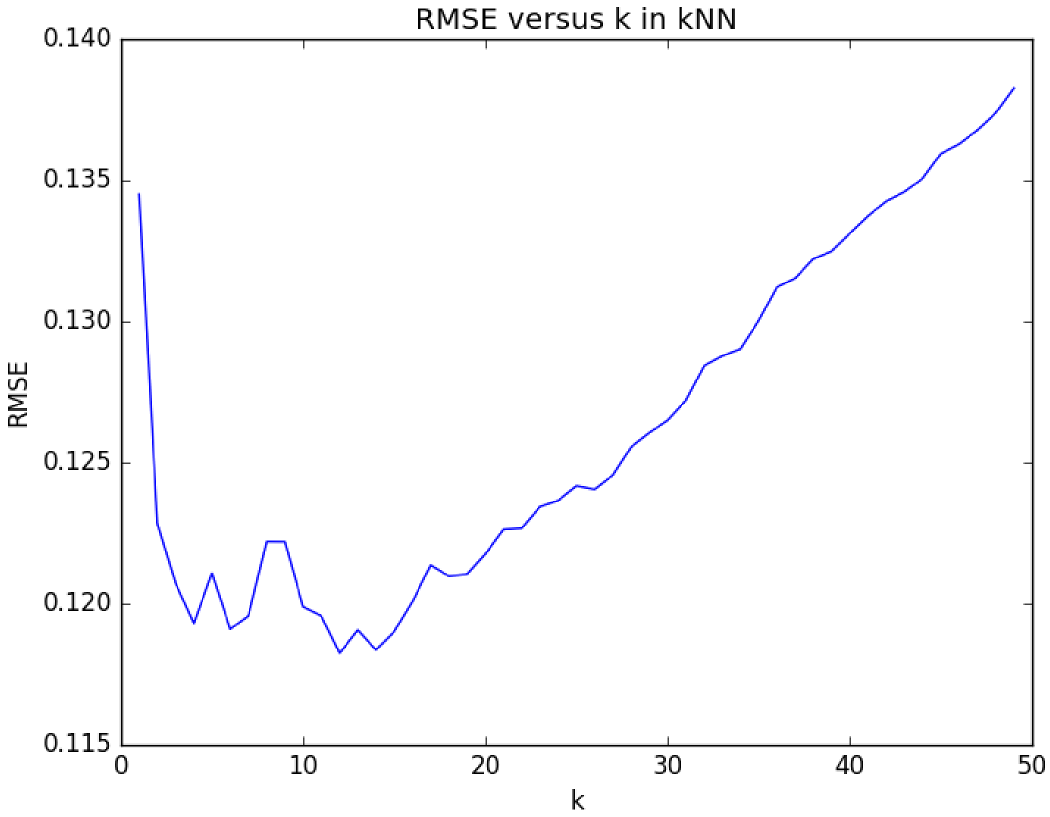
(g) France

The easiest way to do this calculation was using the `kneighbors` method of the `KNeighborsClass:` class.

(j) The most important meta-parameter of k nearest neighbors is k itself. **Plot the RMSE of kNN regression versus k, where k is the number of neighbors. What is the best value of k? What is the RMSE?**

Solution: As k increases, it is initially helpful: averaging more neighbors gives better results. But as k gets too large, we are essentially averaging everything. There is a tradeoff between being too local and being too global.

The best value of k in this case is 12 and gives an RMSE of 0.118 (your value may differ slightly due to randomness). This slightly outperforms regression!



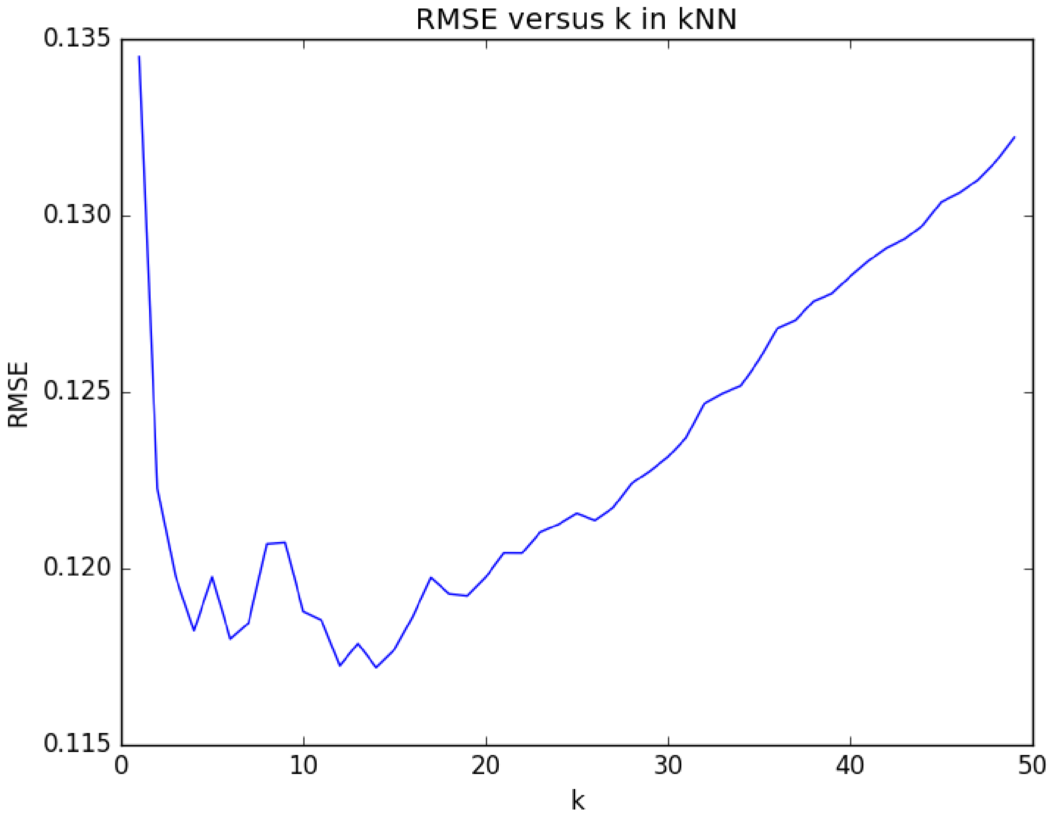
(k) **Explain your plot in (j) in terms of bias and variance.** This is tricky, so take some time to think about it. Think about the spirit of bias and variance more than their precise definitions.

Solution: As k increases, bias *increases*. As k increases, variance *decreases*. There is a still a trade-off, of course, but the model the complexity of the model shrinks as we add more k. This requires a bit of thought, but here are a few ways to understand it:

- When k equals 1, the training error is 0.
- When k equals n, we average all of them so we get a constant estimator, which has variance 0.
- Between 1 and n, k is averaging more points at test time, generating a smoother set of predictions.

(l) We do not need to give every neighbor an equal weight. Maybe closer neighbors are more relevant. For the sake of this problem, let's weight each neighbor by the inverse of its distance to the test point. **Plot the RMSE of kNN regression with distance weighting versus k, where k is the number of features. What is the best value of k? What is the RMSE?**

Solution: The graph looks similar to the graph from earlier, though now the asymptotic behavior is not as bad. This is because at k gets large we are not returning a global average but instead returning an average that favors nearer neighbors. The optimal value of k is 14 and gives an RMSE of 0.117 (your value may differ slightly due to randomness).



(m) One of the challenges of k Nearest Neighbors is that it is very sensitive to the scale of the features. For example, if one feature takes on values 0 or 0.1 and another takes on values 0 or 10, then neighbors will almost certainly agree in the second feature. **Which countries are the 7 nearest neighbors of the USA after scaling (in order)? Compare your result to (i).**

Solution: You will notice that the nearest neighbors do not change much after applying the standard scaling (just two pairs of countries swap orders). In a data set like this, where the features are already in roughly the same scale, that is to be expected. In another data set, the neighbors could change completely!

(a) Ireland

(b) United Kingdom

(c) Finland

(d) Belgium

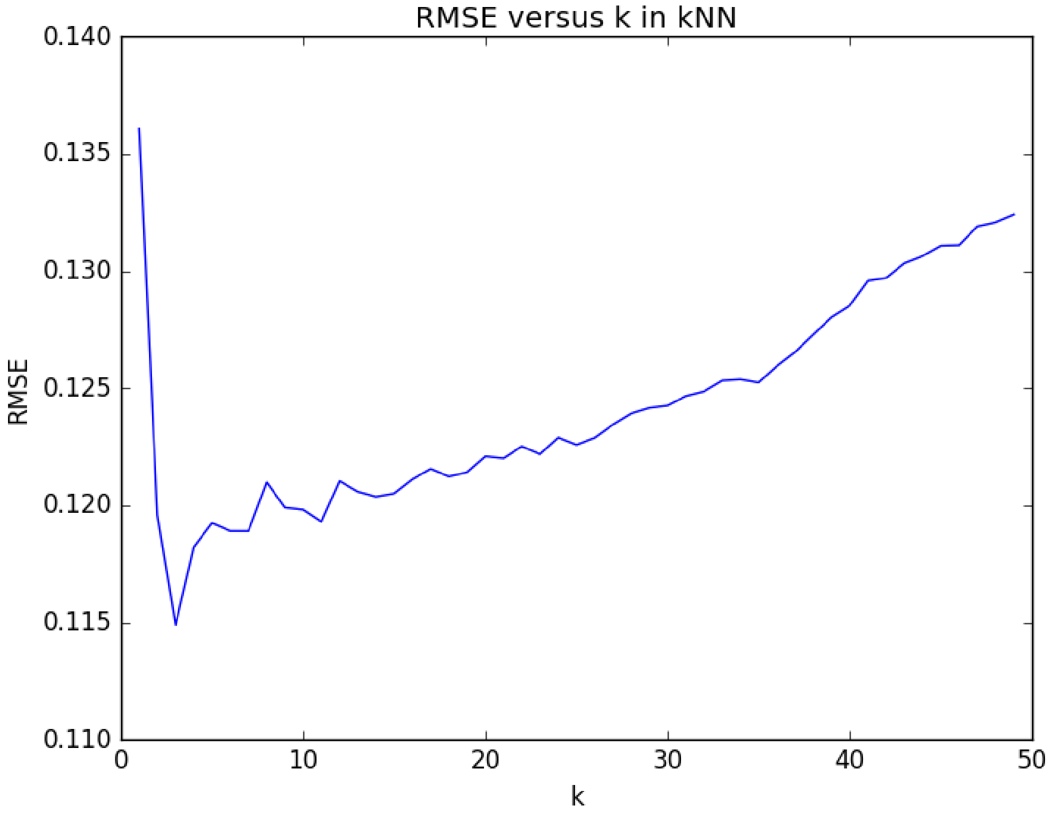
(e) Malta

(f) France

(g) Austria

(n) **Add scaling to your k nearest neighbors pipeline (continue to use distance weighting). Plot RMSE versus k. What is the best value for k? What is the RMSE?**

Solution: The new RMSE is ~ 0.114. This is a slight improvement over the RMSE without scaling. Interestingly, k gets a lot smaller. Now the best k is $k = 3$!



(o) (Bonus): **Rather than scaling each feature to have unit variance, explore ways of scaling the features non-uniformly. How much does this help, if at all?**

Solution: Hopefully, if you tried this you got some intuition for the relation between kNN and scaling. Namely, multiplying important features by larger constants should improve RMSE, as well as multiplying noisy features by small constants. The intuition here is that the distance between two points would be determined primarily by their most scaled features. As an extreme example, scaling some features by 0 is essentially the same as removing them from distance calculations (this is worth thinking about if you do not see why!) On the other hand, scaling one of the world values features by 100 would cause the distance between two countries to be largely determined by that one feature.

As it turns out, doing significantly better than the RMSE attained with just standard scaling appears to be extremely difficult. If you got a negative result (no improvement) with justification, that's fine!

(p) You have been given a set of test features: countries where the responses to the world values survey are given but the HDI is not known. **Using the best model developed so far, predict the HDI values of the countries in the test set. Submit your predictions on Gradescope.**

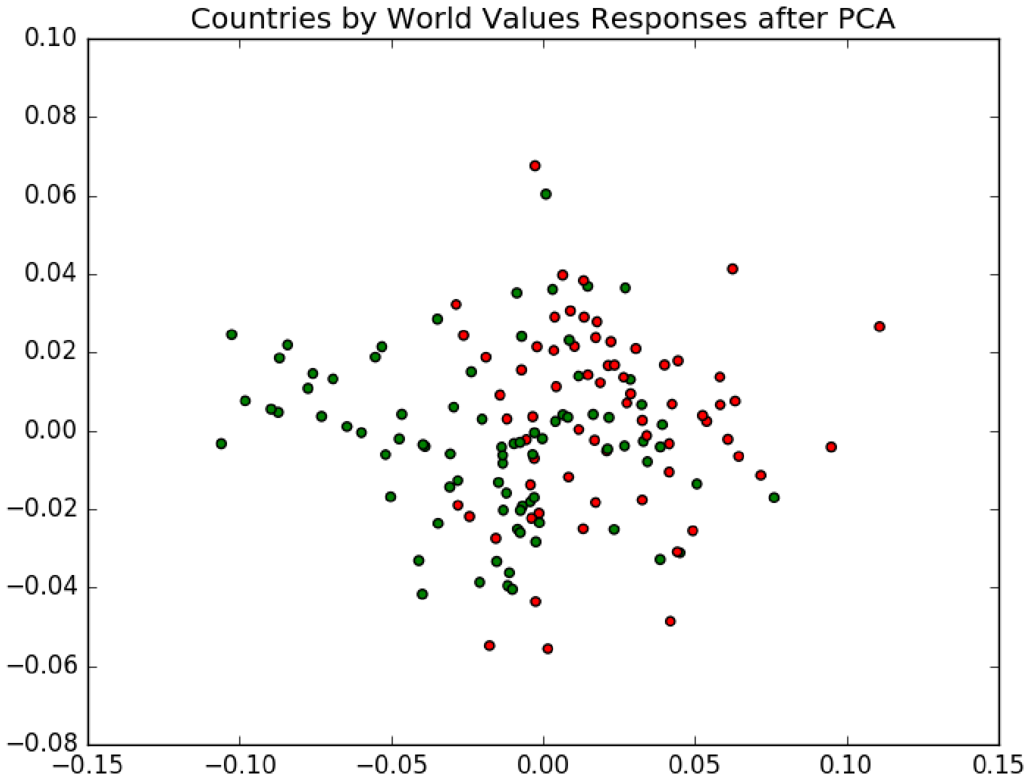
Solution: A good RMSE on the test set is less than 0.12. A great RMSE on the test set is less than 0.11. Prediction on this data set is relatively difficult since there is a lot of noise!

(q) So far we have dealt with the regression problem. Let's take a brief look at classification. A naive classifier is a classifier which disregards the features and just classifies everything as belonging to a single class. **In any classification problem with k classes, what accuracy are we guaranteed to get with the best naive classifier?** (Hint: there are k possible naive classifiers. Use the pigeonhole principle).

Solution: The accuracy is guaranteed to be at least $\frac{1}{k}$, because dividing n data points into k classes means at least one of the k classes contains $\frac{n}{k}$ points, meaning the naive classifier which predicts that class has accuracy $\frac{1}{k}$. In our case, with two classes, it is 50%. This is a good baseline to compare our classification results against.

(r) We will split countries into two groups: high HDI (more than 0.7) and low HDI (less than 0.7). **Plot the countries by their first two PCA dimensions again, but now color them by class.**

Solution: The code to do this was provided for you. You should see the following graph:



(s) Examine the graph generated in (r). **How well do you think a linear SVM would do in classification?**

Solution: In the left of the image, there appear to be a lot of green points (high HDI) that are well-separated and easy to classify with a linear separator. The reds are much more poorly separated, so we expect a lot more confusion there (where they are heavily mixed with the greens).

(t) We will use an SVM classifier to predict whether a country's HDI is "high" or "low" based on the responses of their citizens to the World Values Survey. **Use the code provided to train and cross-validate an SVM classifier using a linear kernel. What is the accuracy of the classifier?**

Solution: Run the code provided. The accuracy is ~ 0.75. Your value may differ slightly due to randomness.

(u) We are going to modify the classifier from (t). **Add a PCA step and Scaling step to the SVM pipeline. Your hyper-parameter search should now be over all possible dimensions for the PCA reduction. Does the accuracy improve?**

Solution: The accuracy improves to ~ 0.81. Notice that we need to adjust C since we have scaled everything up (review the optimization formulation of SVM to see why!) If you did not change the hyper-parameter C, you would have gotten a much lower accuracy.

(v) Change the kernel in (t) from linear to "radial basis function" (rbf). For this part, do not use PCA or Scaling. **What is the accuracy?**

Solution: The accuracy is ~ 0.80. Your value may differ slightly due to randomness.

(w) Now we are going to use k Nearest Neighbors for the same task. That is, we would like to predict whether a country's HDI is "high" or "low" based on the responses of their citizens to the World Values Survey. **Train and cross-validate a k Nearest Neighbors classifier using distance weighting. What is its accuracy? Does scaling help?**

Solution: The accuracy is ~ 0.76 without scaling. With scaling it improves slightly to 0.77. What is interesting here is that while the kNN performed best of the regression methods we tried, it did not perform as well as SVM on classification. Still, it was quite close. Not bad for such a simple algorithm!

(x) (Bonus): Towards the end of the week, we will post the "Berkeley F2017 Values Survey." **If Berkeley were an independent country, what do you predict its HDI would be?**

Solution: Depending on which methods and which data set you used, you will get a different answers. However, **you should have gotten an answer between 0.40 and 0.65**.

You might find this HDI surprisingly low. This is largely explained by a few things. First, a lot of Berkeley students answered "A good education" (in fact, almost everyone who filled out the survey did so!) Interestingly, this feature is somewhat *negatively* correlated with HDI. On the other hand, very few Berkeley students answered "Protecting forests, rivers, and oceans." This feature has the highest correlation with HDI, so the lack of responses penalized the HDI considerably.

If you got values that were wildly different, it is likely that you did not pre-process the data correctly.

(y) (Bonus): **Describe how you would use kNN to revisit the sensor location problem from previous homework. How well do you think it will work?**

Solution: Let's say you had 7 sensors, 100 training points, and 100 testing points. Using the 7 distances of the testing points to the sensors, you could predict their location by averaging the location of their nearest neighbors (taking the centroid) in the 7-dimensional feature space (of distances). It is important to make the distinction between sensor locations and measures distances: the former are labels and the latter are features. In kNN (and, indeed, all ML algorithms), you use features to predict labels.

(z) (Bonus): **What did you learn from this problem? Do you have any useful feedback for the problem author?**

Solution: N/A

4 Your Own Question

Write your own question, and provide a thorough solution.

Writing your own problems is a very important way to really learn material. The famous "Bloom's Taxonomy" that lists the levels of learning is: Remember, Understand, Apply, Analyze, Evaluate, and Create. Using what you know to create is the top-level. We rarely ask you any HW questions about the lowest level of straight-up remembering, expecting you to be able to do that yourself. (e.g. make yourself flashcards) But we don't want the same to be true about the highest level.

As a practical matter, having some practice at trying to create problems helps you study for exams much better than simply counting on solving existing practice problems. This is because thinking about how to create an interesting problem forces you to really look at the material from the perspective of those who are going to create the exams.

Besides, this is fun. If you want to make a boring problem, go ahead. That is your prerogative. But it is more fun to really engage with the material, discover something interesting, and then come up with a problem that walks others down a journey that lets them share your discovery. You don't have to achieve this every week. But unless you try every week, it probably won't happen ever.