EECS 189 Introduction to Machine Learning
Fall 2020

EXAM1

1 Regularization from the Augmentation Perspective (10 points)

Assume $\mathbf{w}$ is a d-dimensional Gaussian random vector $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$ and $\mathbf{\Sigma}$ is symmetric positive-definite. Our model for how the $\{y_i\}$ training data is generated is

$$y = \mathbf{w}^\top \mathbf{x} + Z, \quad Z \sim \mathcal{N}(0, 1), \tag{1}$$

where the noise variables Z are independent of $\mathbf{w}$ and iid across training samples. Notice that all the training $\{y_i\}$ and the parameters $\mathbf{w}$ are jointly normal/Gaussian random variables conditioned on the training inputs $\{\mathbf{x}_i\}$. Let us define the standard data matrix and measurement vector:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}.$$

In this model, the MAP estimate of $\mathbf{w}$ is given by the Tikhonov regularization counterpart of ridge regression:

$$\widehat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X} + \mathbf{\Sigma}^{-1})^{-1} \mathbf{X}^\top \mathbf{y}, \tag{2}$$

In this question, we explore Tikhonov regularization from the data augmentation perspective.

Define the matrix $\mathbf{\Gamma}$ as a $d \times d$ matrix that satisfies $\mathbf{\Gamma}^\top \mathbf{\Gamma} = \mathbf{\Sigma}^{-1}$. Consider the following augmented design matrix (data) $\hat{\mathbf{X}}$ and augmented measurement vector $\hat{\mathbf{y}}$:

$$\hat{\mathbf{X}} = \begin{bmatrix} \mathbf{X} \\ \mathbf{\Gamma} \end{bmatrix} \in \mathbb{R}^{(n+d)\times d}, \quad \text{and} \quad \hat{\mathbf{y}} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0}_d \end{bmatrix} \in \mathbb{R}^{n+d},$$

where $\mathbf{0}_d$ is the zero vector in $\mathbb{R}^d$. **Show that the ordinary least squares problem**

$$\arg\min_{\mathbf{w}} \|\hat{\mathbf{y}} - \hat{\mathbf{X}}\mathbf{w}\|_2^2$$

has the same solution as (2).

(HINT: Feel free to just use the formula you know for the OLS solution. You don't have to rederive that. This problem is not intended to be hard or time consuming.)

2 What is going on? (12 points)

This question relies on your understanding of the behavior of ridge regression and kernel ridge regression as the hyperparameters change. Part (a) uses a piecewise linear function which you have already seen in homework. Parts (b) and (c) show the results of (kernel) ridge regression using heatmaps which you should already be familiar with from homework. For parts (b) and (c) we used the circles dataset which you have already seen, with an 80% train/ 20% test split illustrated in Figure 1 below.

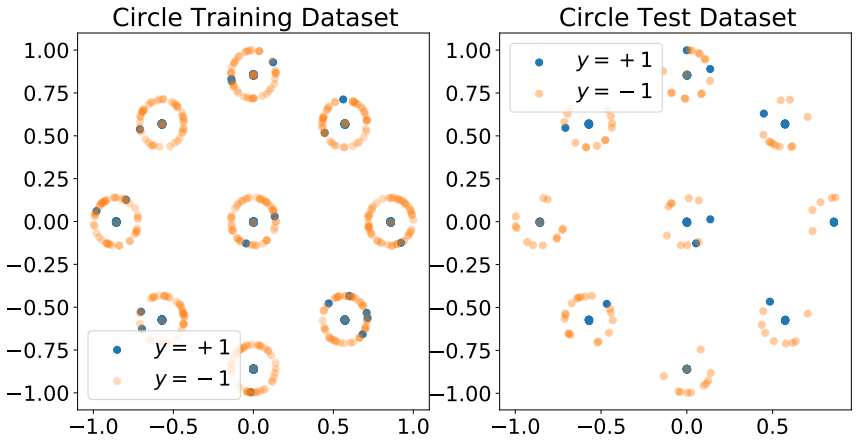


Figure 1: Training and test datasets, with partial transparency to show overlapping datapoints. Notice that there are "mislabeled" points in this dataset.

(a) (6 pts) In this part we have performed kernel ridge regression with the RBF kernel

$$k(x, x') = \exp\left(-\gamma \|x - x'\|^2\right)$$

on a piecewise linear 1D function which you should recognize from homework. Both the regularization parameter λ and the smoothing parameter γ have been varied in the plots on the next page. Your goal in this problem is to identify the qualitative level of the γ and λ parameters given the visible plots of what was learned.

Determine whether the regression was over- or under-regularized and whether the smoothing factor was too wide (small γ) or too narrow (large γ), and write the letter of each subfigure from Figure 2 in the appropriate location on your answer sheet following Figure 3. We have provided the location of (b) to get you started. Not all the marked spots will be used and each spot will correspond to at most one letter.

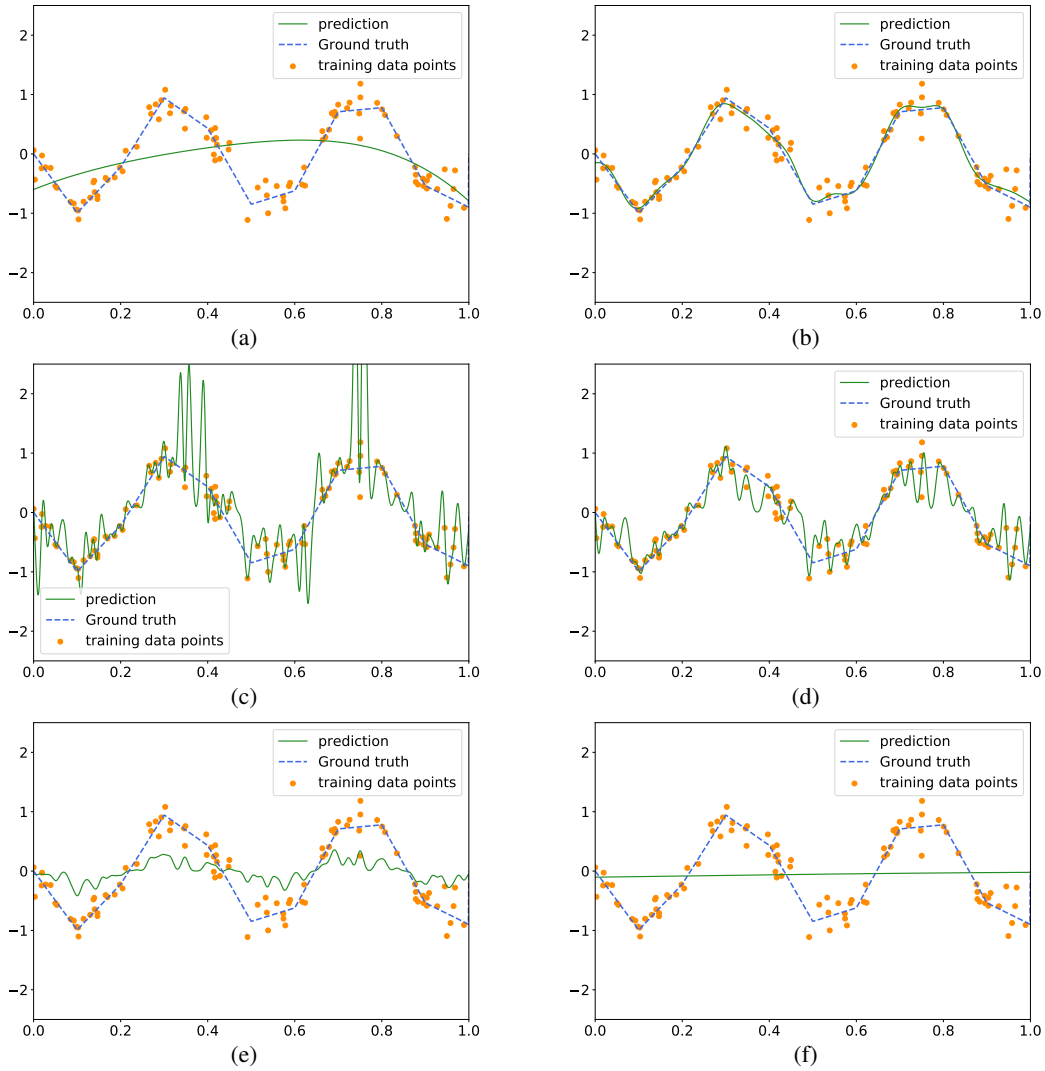


Figure 2: Match these subfigures to the (γ, λ) hyper-parameter space.

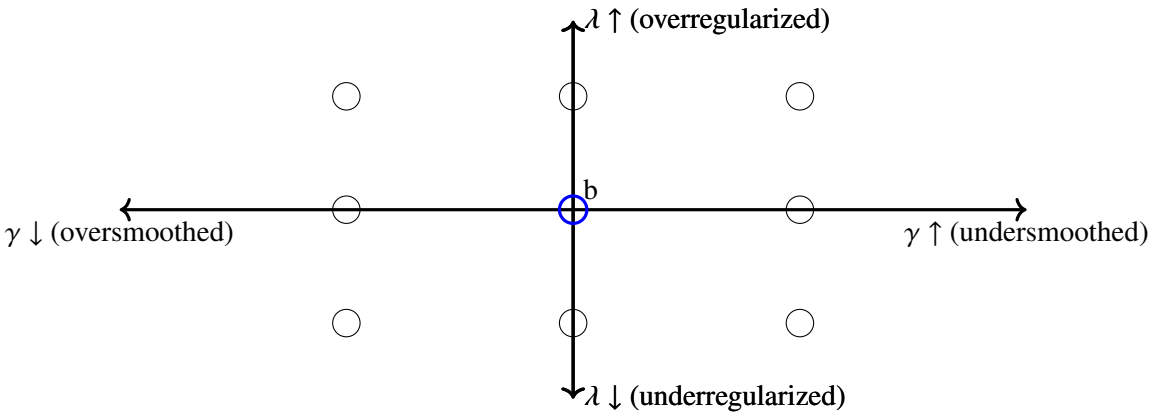
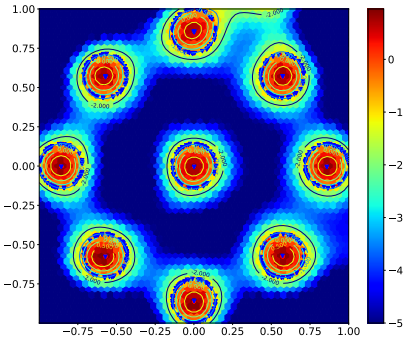
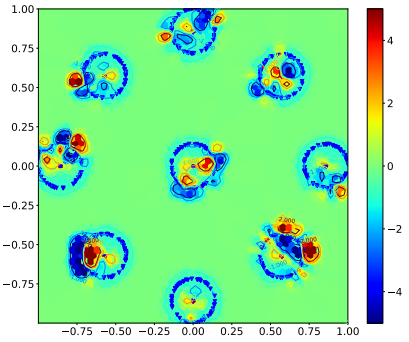


Figure 3: **Write subfigure letters from Figure 2 in the appropriate location on your answer sheet.**

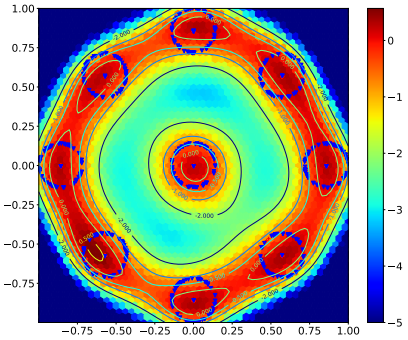
(b) (3 pts) In this part we have performed kernel ridge regression with the RBF kernel $\exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$ and $\lambda = 0.001$. **Match each subfigure (a), (b), and (c) of Figure 4 with the appropriate location (1), (2), or (3) on the test error curve of Figure 5.** For example, write: $(a) \rightarrow (1); (b) \rightarrow (2); (c) \rightarrow (3)$.



(a) Test Error: 0.21



(b) Test Error: 0.56



(c) Test Error: 0.58

Figure 4: Heatmaps corresponding to the functions learned by kernel ridge regression. Match these subfigures to the test error curve below. Don't forget to use the test-error information.

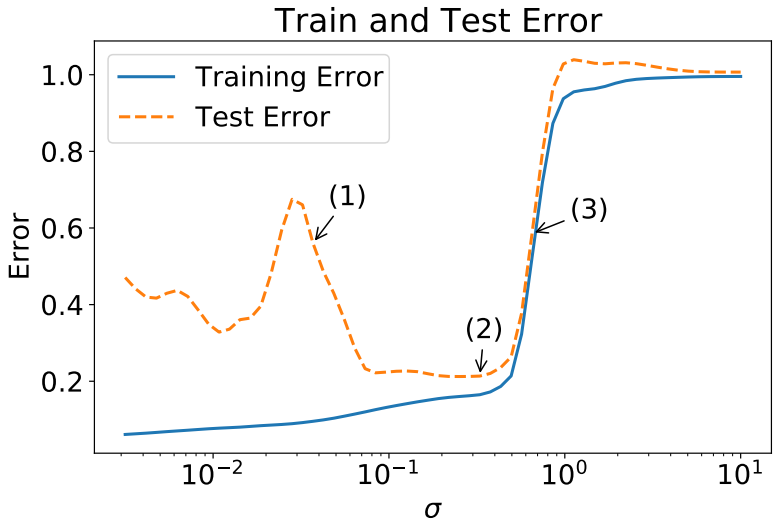
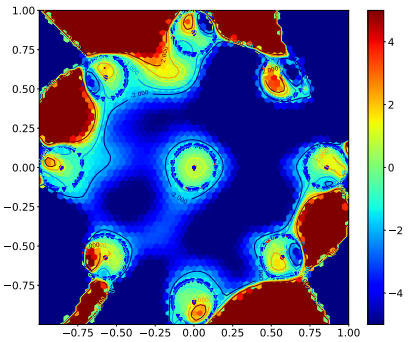
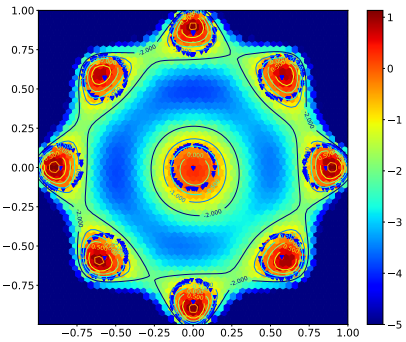


Figure 5: Train and test errors for varying RBF kernel parameter σ.

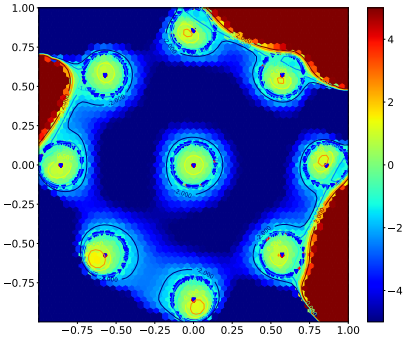
(c) (3 pts) In this part we have performed ridge regression on the circles dataset with polynomial features of increasing degree D. After examining Figure 6, **match each subfigure (a), (b), and (c) with the appropriate location (1), (2), or (3) on the train/test error plot in Figure 7.** For example, write: $(a) \rightarrow (1); (b) \rightarrow (2); (c) \rightarrow (3)$.



(a) Test Error: 0.29



(b) Test Error: 0.29



(c) Test Error: 0.22

Figure 6: Heatmaps corresponding to the functions learned by ridge regression. Match these subfigures to the test error curve below. Don't forget to use the test-error information.

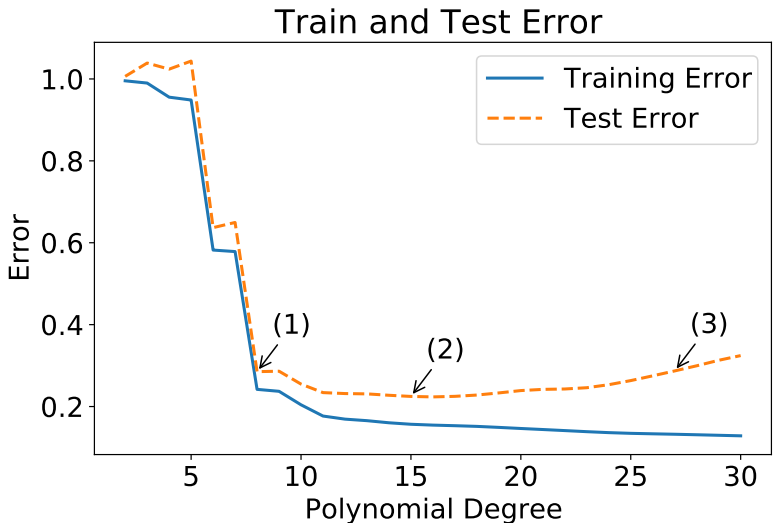


Figure 7: Train and test errors for increasing polynomial degrees.

3 Validation can sometimes fail (40 points)

This problem is a simplified setting where we investigate the probability that validation ends up picking a more complicated false model instead of the simpler true model. Consider the setting of learning a one-dimensional function from noisy samples.

In this problem, suppose the true underlying function is the constant zero function. We have access to noisy training data consisting of n_t data points that we arrange into a pair of n_t-dimensional vectors $(\mathbf{x}_t, \mathbf{y}_t)$. Because the underlying function is the zero function, we have:

$$\mathbf{y}_t = \boldsymbol{\epsilon}_t,$$

where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_{n_t})$. Further we have access to similarly noisy validation data that has been arranged into a pair of n_v-dimensional vectors $(\mathbf{x}_v, \mathbf{y}_v)$ where

$$\mathbf{y}_v = \boldsymbol{\epsilon}_v,$$

and $\boldsymbol{\epsilon}_v \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_{n_v})$. We assume $\boldsymbol{\epsilon}_v$ is independent of $\boldsymbol{\epsilon}_t$.

For ease of computation, assume

$$\|\mathbf{x}_t\|_2^2 = n_t, \qquad \|\mathbf{x}_v\|_2^2 = n_v.$$

We use the training data to learn models and then use the validation data to choose the better performing one.

In this simplified case, the 0-th model is just the constant zero. There are no parameters to learn for this model.

The 1-st model is a simple linear function $f(x) = wx$ where we need to learn the single parameter w from the training data.

For this problem, the various parts are largely independent of each other even though they collectively tell one story. Since this is an exam, don't get bogged down in any one part and lose easy points elsewhere.

(a) (4 pts) Suppose that we just decided to perform OLS on the *validation data* and computed

$$\widehat{w}_v = \arg\min_{w \in \mathbb{R}} \left\| \mathbf{y}_v - \mathbf{x}_v w \right\|_2^2.$$

Just use the formula for OLS to show that

$$\widehat{w}_v = \frac{1}{n_v} \mathbf{x}_v{}^\top \boldsymbol{\epsilon}_v.$$

and give the mean and variance of the Gaussian random variable $\widehat{w}_v$.

Here, we are viewing this hypothetical $\widehat{w}_v$ as a random variable because the noise in the validation data is random.

(b) (4 pts) By doing a calculation analogous to the previous part, you can understand the actually learned parameter $\widehat{w}_t$ obtained by performing OLS on the training data. This gives

$$\widehat{w}_t = \frac{1}{n_t}\mathbf{x}_t^\top \boldsymbol{\epsilon}_t.$$

and this has a Gaussian distribution $\widehat{w}_t \sim \mathcal{N}(0, \frac{\sigma^2}{n_t})$. Here, the randomness in $\widehat{w}_t$ comes from the random noise in the training data.

You now have the marginal distributions for $\widehat{w}_t$ and $\widehat{w}_v$. **Find the joint distribution of $\widehat{w}_t$ and $\widehat{w}_v$**, i.e. find the distribution of

$$\mathbf{w} = \begin{bmatrix} \widehat{w}_t \\ \widehat{w}_v \end{bmatrix}.$$

(c) (12 pts) Suppose we have the following two choices for the predictor:
Choice 0: $\widehat{f}(x) = 0$.
Choice 1: $\widehat{f}(x) = x\widehat{w}_t$.
We use the validation data to determine which choice to use by computing the validation errors:

$$\mathcal{E}_0^{\text{val}} = \left\|\mathbf{y}_v\right\|_2^2,$$
$$\mathcal{E}_1^{\text{val}} = \left\|\mathbf{y}_v - \mathbf{x}_v\widehat{w}_t\right\|_2^2.$$

If $\mathcal{E}_0^{\text{val}} \leq \mathcal{E}_1^{\text{val}}$, we correctly choose the zero predictor.
If $\mathcal{E}_0^{\text{val}} > \mathcal{E}_1^{\text{val}}$, we choose the linear model (Choice 1).

Since the true function is actually zero, the probability that validation fails is given by:

$$P(\text{Validation fails}) = P(\mathcal{E}_1^{\text{val}} < \mathcal{E}_0^{\text{val}}).$$

Show that:

$$P(\mathcal{E}_1^{\mathbf{val}} < \mathcal{E}_0^{\mathbf{val}}) = P\left((\widehat{w}_v - \widehat{w}_t)^2 < \widehat{w}_v^2\right).$$

(Hint 1: Start expanding $\mathcal{E}_1^{val} - \mathcal{E}_0^{val}$.
Hint 2: At some point, you'll need to expand out squares $\|\mathbf{v}\|_2^2 = \mathbf{v}^T\mathbf{v}$.
Hint 3: At an opportune point, use the trick

$$\mathbf{y}_v = \mathbf{y}_v - \mathbf{x}_v\widehat{w}_v + \mathbf{x}_v\widehat{w}_v.$$

Hint 4: Remember that the orthogonality principle in ordinary least squares tells us that the residual vector $\mathbf{y}_v - \mathbf{x}_v\widehat{w}_v$ is orthogonal to the vector $\mathbf{x}_v$.)

(d) (4 pts) Simple algebra takes the previous result $P(\mathcal{E}_1^{\text{val}} < \mathcal{E}_0^{\text{val}}) = P\left((\widehat{w}_v - \widehat{w}_t)^2 < \widehat{w}_v^2\right)$ and simplifies it to

$$P(\mathcal{E}_1^{\text{val}} < \mathcal{E}_0^{\text{val}}) = P\left(\widehat{w}_t(\widehat{w}_t - 2\widehat{w}_v) < 0\right).$$

This lets us illustrate the realizations of $\widehat{w}_t, \widehat{w}_v$ for which validation fails as the shaded region in Fig. 8:

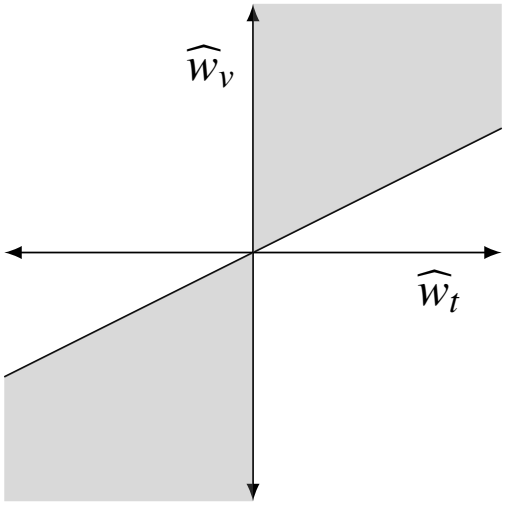


Figure 8: Illustrating the region where validation makes the wrong choice.

Now suppose we had some particular value of $\widehat{w}_t > 0$. **Draw a one-dimensional region corresponding to values of $\widehat{w}_v$ for which validation fails.**

Your drawing should have a 1d real line with both zero and $\widehat{w}_t$ marked, and a shaded region that corresponds to those values of $\widehat{w}_v$ for which validation would make the wrong choice. Be sure to label the boundary of the region.

(e) (12 pts) Use the illustration in Fig. 8 and the fact that $P(\mathcal{E}_1^{\text{val}} < \mathcal{E}_0^{\text{val}}) = P\left(\widehat{w}_t(\widehat{w}_t - 2\widehat{w}_v) < 0\right)$ to **show that the probability validation fails is given by**

$$P(\mathcal{E}_1^{\textbf{val}} < \mathcal{E}_0^{\textbf{val}}) = \frac{1}{\pi}\cos^{-1}\left(\frac{1}{\sqrt{1 + \frac{4n_t}{n_v}}}\right). \tag{3}$$

(Hint: Recall from Homework 6, Question 4 that if $\mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$ *where* $\boldsymbol{\Sigma}$ *is a positive definite*

$$\boldsymbol{\Sigma} = \begin{bmatrix} a & c \\ c & b \end{bmatrix},$$

then

$$P(X_1 > 0, X_2 > 0) = \frac{1}{2\pi}\cos^{-1}\left(\frac{-c}{\sqrt{ab}}\right).$$

and ask yourself what the relevant 2d Gaussian random variables are here that you need to think about to understand the probability of the event we want to understand.)

(f) (4 pts) Use what you proved in the previous part, $P(\mathcal{E}_1^{\text{val}} < \mathcal{E}_0^{\text{val}}) = \frac{1}{\pi}\cos^{-1}\left(\frac{1}{\sqrt{1+\frac{4n_t}{n_v}}}\right)$, to

evaluate $P(\mathcal{E}_1^{\textbf{val}} < \mathcal{E}_0^{\textbf{val}})$ **when** $n_t = 50, n_v \to \infty$ as well as
evaluate $P(\mathcal{E}_1^{\textbf{val}} < \mathcal{E}_0^{\textbf{val}})$ **when** $n_t \to \infty, n_v = 50$**.**

Feel free to use the facts that $\cos^{-1}(1) = 0$ and $\cos^{-1}(0) = \frac{\pi}{2}$.

4 Kernel Controversy (42 points)

Your friend rushes up to you with an excited tone. They say that they've got a more flexible way of understanding Kernel Ridge Regression:

"At the end of the day, we represent the learned function using the following representation:

$$\widehat{f}(\mathbf{x}) = \sum_{i=1}^{n} \beta[i] k(\mathbf{x}_i, \mathbf{x}) \tag{4}$$

where $\mathbf{x}_i$ are the training inputs. If we take that as a given, all we are doing is giving ourselves a new set of n features. I can just think about the i-th feature as being $\phi_i(\mathbf{x}) = k(\mathbf{x}_i, \mathbf{x})$. Instead of doing something special like kernel ridge regression, we can just use our $\{\phi_i(\cdot)\}$ features and do ordinary ridge regression to get our weights $\boldsymbol{\beta}$. If we think about things this way, we don't even have to worry about things like positive-semi-definiteness and anything like that. It just works."

This problem is about exploring your friend's idea. You'll do traditional kernel ridge regression first. Throughout this problem you will use the linear kernel only.

(a) (10 pts) Suppose we have training pairs $\{(\mathbf{x}_i, y_i)\}$ and arrange the input data into the standard data matrix $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix}$ and the training outputs into a vector $\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$.

Consider the "linear kernel" $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T\mathbf{z}$. For kernel ridge regression, we have the standard way of learning weights

$$\widehat{\boldsymbol{\beta}} = (\mathbf{K} + \lambda \mathbf{I})^{-1}\mathbf{y} \tag{5}$$

where the symmetric Kernel Gram matrix $\mathbf{K}$ is defined by

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \cdots & k(\mathbf{x}_1, \mathbf{x}_n) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \cdots & k(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \ddots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & k(\mathbf{x}_n, \mathbf{x}_2) & \cdots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}. \tag{6}$$

If we use kernel ridge regression with the linear kernel, **give an expression for what the final learned function $\widehat{f}(\mathbf{x})$ would be if we followed (5) to learn weights for use in (4).**

Please give your expression in the matrix form $\widehat{y}_{\text{test}} = \mathbf{x}_{\text{test}}^T\big(\ \cdot\ \big)\mathbf{y}$.

In other words, tell us what goes inside those parentheses. You can use $\mathbf{X}$, $\mathbf{I}$, λ, and mathematical operations.

(HINT: What is the relationship between $\mathbf{X}$ and $\mathbf{K}$?)

(b) (10 pts) Now we will follow the approach given by your friend. For each potential input $\mathbf{x}$, you can compute an $n-$dimensional feature vector $\boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} \phi_1(\mathbf{x}) \\ \phi_2(\mathbf{x}) \\ \vdots \\ \phi_n(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}) \\ k(\mathbf{x}_2, \mathbf{x}) \\ \vdots \\ k(\mathbf{x}_n, \mathbf{x}) \end{bmatrix}$ and you can define a new $n \times n$ dimensional feature matrix from your training data in the usual way:

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\phi}(\mathbf{x}_1)^T \\ \boldsymbol{\phi}(\mathbf{x}_2)^T \\ \vdots \\ \boldsymbol{\phi}(\mathbf{x}_n)^T \end{bmatrix}. \tag{7}$$

Using the "linear kernel" $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T\mathbf{z}$ to construct our features above, do standard ridge regression using this feature data matrix and the training outputs $\mathbf{y}$ to learn feature weights $\boldsymbol{\beta} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda\mathbf{I})^{-1}\boldsymbol{\Phi}^T\mathbf{y}$ for our new learned function $\widehat{f_{\text{new}}}(\mathbf{x}) = \sum_{i=1}^{n} \beta[i]\phi_i(\mathbf{x}) = \sum_{i=1}^{n} \beta[i]k(\mathbf{x}_i, \mathbf{x})$.

You can use this function to make predictions for a test point $\widehat{y}_{\text{test, new}} = \widehat{f_{\text{new}}}(\mathbf{x}_{\text{test}})$.

Give an expression in matrix form involving $\mathbf{X}, \lambda, \mathbf{I}$ for: $\widehat{y}_{\mathbf{test, new}} = \mathbf{x}_{\mathbf{test}}^T\big(\ \cdot\ \big)\mathbf{y}$.

In other words, tell us what goes inside those parentheses.

(HINT: What is the relationship between the $\mathbf{K}$ *matrix from the previous part and the* $\boldsymbol{\Phi}$ *matrix here?)*

(c) (18 pts) Suppose $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$ where we are using the full-sized SVD and so $\mathbf{U}$ is $n \times n$ and the matrix $\mathbf{V} = m \times m$ where m is the size of the input vectors $\mathbf{x}$.

We know in this case that the ordinary-least-squares prediction based on the input data would just be

$$\sum_{i=1}^{\min(n,m)} (\mathbf{u}_i^T\mathbf{y})\Big(\frac{1}{\sigma_i}\Big)\mathbf{v}_i^T\mathbf{x}_{test} \tag{8}$$

where the matrix $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_n]$ and $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_m]$ and σ_i are the singular values of the matrix $\mathbf{X}$.

Give counterpart expressions for what would be between the central parentheses (where OLS has $\frac{1}{\sigma_i}$) in (8) for three approaches for learning:

- **What you did in part (a): kernel ridge regression with a linear kernel.**
- **What you did in part (b): your friend's approach to ridge regression with kernel features for the linear kernel.**
- **What you would have gotten had you done $k-$PCA+OLS to learn the pattern.**

Your answers should only have a dependence on the singular values and the chosen λ for the two approaches above (*this can help you check your work for the earlier parts*), and the singular values and k for the case of using PCA for dimensionality reduction followed by OLS.

(d) (4 pts) **Comment on whether the two approaches to ridge regression with kernels are the exactly the same, similar, or in what sense they are different in what they end up doing.**

(HINT: What is the overall behavior as you sweep through possible values for the hyperparameter λ*? Although this part doesn't ask you to comment on the relationship to* $k-$*PCA+OLS, you might find it helpful to remember what happens there as you sweep through* k*.)*

5 Regularization with one-hot coupons (18 points)

The first p features in our input data are regular input features. The last d features are a one-hot encoding of a coupon that can have d different types, each equally likely. For each input x, a d-sided die is independently rolled and an integer ξ is picked uniformly at random from $p+1$ to $p+d$. Then, because we are using one-hot encoding, the ξ-th feature is a 1 while all the other coupon-features are 0. Mathematically:

$$x[j] = \begin{cases} 0, & \text{if } j \geq p+1 \text{ and } j \neq \xi, \\ 1, & \text{if } j = \xi. \end{cases}$$

Depending on whether or not we include coupon features, we get two different training data matrices:

$$\mathbf{X}_{\text{train}} = \begin{bmatrix} x_1[1] & x_1[2] & \cdots & x_1[p] \\ x_2[1] & x_2[2] & \cdots & x_2[p] \\ \vdots & \vdots & \ddots & \vdots \\ x_n[1] & x_n[2] & \cdots & x_n[p] \end{bmatrix}; \quad \mathbf{X}_{\text{train, long}} = \begin{bmatrix} x_1[1] & x_1[2] & \cdots & x_1[p+d] \\ x_2[1] & x_2[2] & \cdots & x_2[p+d] \\ \vdots & \vdots & \ddots & \vdots \\ x_n[1] & x_n[2] & \cdots & x_n[p+d] \end{bmatrix}.$$

Your training data set has n i.i.d. training samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, where all $\mathbf{x}_i$ are drawn as described above. You also collect an independent test set with m i.i.d. test samples $\{(\mathbf{x}_i, y_i)\}_{i=n+1}^{n+m}$ from the same distribution.

Two data analysts are planning the following experiment:

- The first data analyst thinks that the coupons are irrelevant, so she will remove the d coupon features. Then she will perform ridge regression with the remaining p-dimensional data with $\lambda = 1$. In other words, she just uses the matrix $\mathbf{X}_{\text{train}}$ above along with the $\mathbf{y}$ to do ridge regression to learn p weights.
- The second data analyst will simply find the minimum norm interpolating solution while including the coupon-features. So, the second data analyst uses the much wider training data matrix $\mathbf{X}_{\text{train, long}}$ together with the $\mathbf{y}$ and learns the $p+d$ weights by computing:

$$\widehat{\mathbf{w}} = \underset{\mathbf{w} \text{ such that } \mathbf{X}_{\text{train, long}}\mathbf{w}=\mathbf{y}}{\arg\min} \|\mathbf{w}\|^2$$

Both analysts use their learned models to compute predictions on the test set and compare results.

With probability $\prod_{i=0}^{m+n-1}(1-\frac{i}{d})$, the $m+n$ coupons you get for the training and test set will all be distinct: in this event, you won't get the same coupon twice across all $m+n$ samples. **Show that the predictions of both data analysts on the entire test data will be exactly the same if all $m+n$ coupons turn out to be distinct.**

(Hint: Draw a picture of what that looks like for the training data and for the test data that we have. Which perspective on ridge regression is likely to be useful here?)

6 Local approximation using neighbors (20 points)

Suppose we are learning a one-dimensional function f. We have access to noisy samples (x_i, y_i) with $x_i, y_i \in \mathbb{R}$, We consider the estimator

$$\widehat{f}(x) = \sum_i y_i h(x - x_i),$$

where $h(x - x_i)$ is the local influence of the i-th training sample on the value of $\widehat{f}$ at x.

(a) (5 pts) Suppose our training samples x_i were equally spaced (Δ apart: so $x_{i+1} = x_i + \Delta$) and ordered (i.e., $x_i < x_j$ for $i < j$), with i going through the integers from $-\infty$ to $+\infty$. If $\widehat{f}$ is a k-nearest neighbor estimator (i.e., the prediction for a test point is the average of the closest k samples from the training data), and $k = 2\ell$ for some strictly positive integer ℓ, **what is the relevant local influence function $h(\cdot)$ that would give this estimator?** Your answer should involve ℓ and Δ in some way.

(HINT: Draw what is going on to help.)

(b) (15 pts) Now, let us consider an explicit function that we want to learn: the absolute value function $f(x) = |x|$. Suppose we have training samples (x_i, y_i) that are separated by $\Delta = 1$ with $x_i = i + \frac{1}{2}$ and $y_i = f(x_i) + \epsilon_i$ *for all integers i.* The training noise is i.i.d. with $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$.

If our test point is just $x = 0$, our goal is to understand the best value of the hyperparameter ℓ so that our mean squared error on this test point would be minimized when using a $k = 2\ell$-nearest neighbor estimator. **What ℓ should we choose as a function of the training noise standard deviation σ?**

(HINT: The only randomness here is in the noise on the training points. To understand the dependence of terms you might want to make sure you understand how the mean-squared error behaves when $\ell = 1$ and $\ell = 2$ before working with a formula. What gets worse as you increase ℓ? What might be getting better?)

Depending on how you you choose to solve this, you might find the formula for the sum of an arithmetic series useful $\sum_{i=1}^{\ell} i = \frac{1}{2}\ell(\ell + 1)$.

Feel free to compute a possibly continuous value for ℓ — on the exam, don't worry about what the right way to round it might be.

Doodle page! Draw us something if you want or give us suggestions or complaints.
You can also use this page to report anything suspicious that you might have noticed.