

4 Kernels

Course: COMPSCI 589 Machine Learning, Spring, 2021

Instructor: Justin Domke

Assignment: 4

Group work policy: You are allowed to complete this homework in teams of at most 3 students. However, each student must submit their own individual .pdf and .zip files to Gradescope. List your team members at the beginning of your [report.pdf](#). You may verbally discuss the assignment with course staff or students outside your group. Please also list any students or course staff (separately) at the beginning of your [report.pdf](#). However, you may not *look, copy, or show* any part of another student's assignment. Copying any part of another assignment — even a single sentence or line of code — from anyone outside your team is considered plagiarism. We use sophisticated tools to detect this. Please do not do it.

2

Group Finder

Due date: April 16, 2021, 5:00 PM

Submission instructions:

- For this assignment, you should prepare your solutions in one of three formats:
 - Latex (any style)
 - Markdown
 - Jupyter notebook
- Regardless of how you prepared the solutions, you should export a single .pdf file that you upload to Gradescope. The .pdf should be submitted to the [Assignment 4: Kernels](#) assignment. Most coding question will ask you to include your code as text in the solution .pdf.

- Additionally, you **must submit a .zip file** to [Assignment 4: ZIP File](#) in Gradescope. Your .zip file should contain four things:
 - `report.pdf` - Your report.
 - `report_src/` - A directory Assignment 4: ZIP File containing all source files for the report.
 - `code/` - A directory containing Python code for all parts of the assignment.
 - `code/run_me.py` - A single Python file that will generate all figures included in your report.
- If you use a Jupyter notebook, nothing changes. You still must put your code in external files, and you still must submit both a single .pdf and a .zip file containing the above components to the respective assignments in Gradescope.
- When you submit the .pdf to Gradescope, you must mark page numbers for the different questions. We hate to do it, but we will penalize anyone who does not do this, as it creates a huge amount of difficulty for the graders.
- For the purpose of late days, the later of your two submissions will be considered the submission time for your assignment. E.g., if you submit your .pdf on time, but the .zip is two days late, the assignment will be considered two days late.
- The assignment asks you to upload predictions to [Kaggle](#). Please create a Kaggle account using your [umass.edu](#) email address. Otherwise, it is very difficult for us to map Kaggle submissions to students.

Kernel Ridge Regression

Question 1 (5 points) Suppose that we wish to find the vector w that minimizes

$$\sum_{n=1}^N \left(w \cdot x^{(n)} - y^{(n)} \right)^2 + \lambda \|w\|^2.$$

Show that the optimal w is

2

$$w^* = \left(\sum_{n=1}^N \left(x^{(n)} \right) \left(x^{(n)} \right)^\top + \lambda I \right)^{-1} \sum_{n=1}^N x^{(n)} y^{(n)}.$$

Question 2 (5 points) Suppose now that we apply a basis expansion to x so that the goal is instead to minimize

2

$$\sum_{n=1}^N \left(w \cdot h(x^{(n)}) - y^{(n)} \right)^2 + \lambda \|w\|^2.$$

What now is the optimal w^* ? Argue why your answer is correct.

Question 3 (5 points) Take some new input x^{pred} . Naively, we would predict $y^{\text{pred}} = w^* \cdot h(x^{\text{pred}})$. Suppose, however, that we have access to some kernel function k such that $k(x, x') = h(x) \cdot h(x')$. Derive an expression for y^{pred} that makes no reference to $h(\cdot)$ or w^* . (Your expression can use "temporary" variables if you like. These must not reference $h(\cdot)$ or w^* either of course!)

Question 4 (5 points) Consider a 1-D input x . Consider the following polynomial basis expansion:

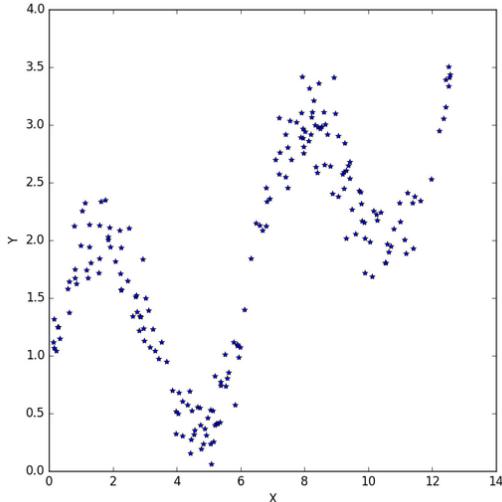
9+

$$h(x) = [c_0, c_1 x, c_2 x^2, \dots, c_P x^P], \quad c_p = \sqrt{\binom{P}{p}},$$

Derive a kernel function $k(x, x')$ such that $k(x, x') = h(x) \cdot h(x')$ for all x and x' . (Hint: your kernel function should be significantly simpler than the basis expansion you started with.)

You are given a file [data_synth.npz](#). This contains 200 1D inputs and 1D outputs, plotted below.

 [data_synth.npz](#) 7.2KB



As in previous assignments, the data can be loaded as follows:

```
stuff = np.load("data_synth.npz") X_trn = stuff['X_trn'] Y_trn =  
stuff['Y_train'] X_val = stuff['X_val'] Y_val = stuff['Y_val']
```

2

Here is a function that performs that basis expansion from the question above.
(You'd use this by doing `h=get_poly_expansion(5)` and then `h(X)`.)

```
from scipy.special import comb def get_poly_expansion(P): def  
expand(X): tmp = [np.sqrt(comb(P,p))*X**p for p in range(P+1)]  
return np.vstack(tmp).T return expand # example usage h =  
get_poly_expansion(5) expansion = h(X_trn[0])
```

4

Question 5 (5 points) Provide a function that will evaluate ridge-regression when using basis expanded data, i.e. evaluate $f_w(x) = w \cdot h(x)$. Your function should have the following signature:

```
def eval_basis_expanded_ridge(x,w,h): # do stuff return y
```

Here `x` is a single input, `w` is a vector of weights and `h` is a basis-expansion function (e.g. what you'd get from calling `get_poly_expansion(3)`). The return value `y` is just a scalar. Provide your function directly in your report.

Question 6 (5 points) Provide a function that will do ridge-regularization on basis-expanded data, i.e. minimize

$$\sum_{n=1}^N \left(f_w(x^{(n)}) - y^{(n)} \right)^2 + \lambda \|w\|^2,$$

over w where f_w is the function you implemented in the previous question. Your function should have the following signature.

4

```
def train_basis_expanded_ridge(X,Y,λ,h): # do stuff return w
```

Here X is a 1D array of inputs, Y is a 1D array of training outputs, λ is a regularization constant, and h is a basis-expansion function. Provide your function directly in your report.

1

Question 7 (5 points) For each value of $P \in \{1, 2, 3, 5, 10\}$, do basis-expanded ridge regression using a P th order basis expansion on the given dataset with $\lambda = 0.1$. For each value of P please:

- Report the vector w that you recovered.
- Make a plot of the final learned function $f_w(x)$ as a function of x . Plot this between $x = 0$ and $x = 15$, superimposed on the training data.

8

Question 8 (5 points) Implement a method to get a polynomial kernel. Your function should have the following signature

2

```
def get_poly_kernel(P): def k(x,xp): # do stuff return kernel_value
    return k
```

This kernel function should be equivalent to taking the inner-product of two basis expansions, i.e. that $k(x, x') = h(x) \cdot h(x')$. Your kernel function must **never** form or create a basis expansion, and must have a time complexity that doesn't depend on P (assume that you can take the power of a scalar in constant time.) Provide your function directly in your report.

Question 9 (5 points) Run the following code

```
x = 0.5 xp = 0.7 k = get_poly_kernel(5) h = get_poly_expansion(5)
out1 = k(x,xp) out2 = np.inner(h(x),h(xp)) print("output 1", out1)
print("output 2", out2)
```

what is the output?

Question 10 (5 points) Implement a function to train a kernel ridge regression model. Given a dataset of inputs $x^{(n)}$ and outputs $y^{(n)}$ you should compute

$$\alpha = (K + \lambda I)^{-1}y$$

where $K_{nm} = k(x^{(n)}, x^{(m)})$ is the kernel function evaluated on the n th and m th inputs. Your function should have the following signature:

```
def train_kernel_ridge(X,Y,λ,k): # do stuff return α
```

Here X is a 1D array of inputs, Y is a 1D array of training outputs, λ is a regularization constant, and k is a kernel function. Do *not* use `numpy.inv` in your solution. If you're tempted to do that, look into `numpy.linalg.solve` instead. Provide your solution directly in your report. You can/should call your kernel function from the previous question.

Question 11 (5 points) Implement a function to evaluate a kernel ridge regression model. Given a dataset of inputs $x^{(n)}$, a vector α ; a kernel function k , and a new input x , you should compute

$$\sum_{n=1}^N \alpha_n k(x^{(n)}, x).$$

Your function should have the following signature:

```
def eval_kernel_ridge(X_trn, x, α, k): # do stuff return y
```

Here `X_trn` is a 1D array of training inputs, `x` is the input to evaluate, `α` is a vector of learned components, and `k` is a kernel function. Provide your solution directly in your report.

Question 12 (5 points) For each value of $P \in \{1, 2, 3, 5, 10\}$, do kernel-expanded ridge regression using a P -th order polynomial kernel on the given dataset with $\lambda = 0.1$. For each value of P make a plot of the final learned function $f_w(x)$ as a function of x . Plot this between $x = 0$ and $x = 15$, superimposed on the training data. (You can give either 5 separate plots, one for each value of P or a single plot with the different values of P superimposed and labeled.)

7

Question 13 (5 points) How do your results using kernel ridge regression compare to those you obtained using basis-expanded ridge regression? Explain why in at most two sentences.

Support Vector Machines

0 data_real.npz 49.0KB

3

You are given a file with 686 training inputs of length 4 (`x_trn`) and corresponding outputs (`y_trn`). You are also given 686 test inputs (`x_tst`).

For this question, you will perform classification using support vector machines on this dataset. For this question you allowed (and encouraged) to use `sklearn`'s implementation. However, you may **not** use

`sklearn.model_selection.cross_val_score`.

6

In the questions below, we define a support vector machine to be the result of minimizing

$$\min_{\alpha} \sum_{n=1}^N L\left(y^{(n)}, f_{\alpha}(x^{(n)})\right) + \lambda \sum_{n=1}^N \sum_{m=1}^M \alpha_n \alpha_m k(x^{(n)}, x^{(m)}),$$

where

2

$$L(y, f) = \max(0, 1 - yf)$$

is the hinge loss, and

3

$$f_\alpha(x) = \sum_{n=1}^N \alpha_n k(x, x^{(n)}).$$

Question 14 (10 points) Train a support vector machine with a linear kernel with each of the following regularization penalties: $\lambda \in \{2, 20, 200\}$. Estimate the mean validation-set hinge loss using 5-fold cross validation. Give your results here as a table with one entry for each value of λ . (Note: Make sure you understand exactly what the `sklearn` arguments do!)

Question 15 (10 points) We can define a polynomial kernel with a constant term as

5

$$k(x, x') = (\gamma + x \cdot x')^P,$$

where P is the degree and γ is some constant. For each of $\gamma \in \{1, .01, .001\}$ and $\lambda \in \{2, 20, 200\}$, train a support vector machine using a polynomial kernel of degree $P = 3$ and a regularization penalty of λ . Estimate the validation-set hinge loss using 5-fold cross validation. Give your results as a 3-column table with one entry for each (γ, λ) pair. (Make sure it's clear what entry corresponds to which value.)

4

Question 16 (10 points) Repeat the previous question, but using a polynomial of degree $P = 5$ instead.

Question 17 (10 points) We can define a "radial basis function" kernel as

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

Again, for each of the $\gamma \in \{1, .01, .001\}$ and $\lambda \in \{2, 20, 200\}$ values, train and evaluate this model using 5-fold cross-validation. Report your estimated generalization error (validation-set hinge loss) in a 3x3 table. 💬 6

Question 18 (10 points) Which kernel (with which value γ , if applicable) and which regularization constant performed best? Fix this kernel, retrain on all the data, and then make predictions for the test data. Upload these to the [Assignment 4 Kaggle competition](#) (again, please use your UMass email so that we can identify your submission!) and report here your error on the public leaderboard. Please give: 💬 8

- What kernel you chose.
- Your estimated 0-1 generalization error (i.e., 1 - accuracy). 💬 1
- Your observed generalization error on the leaderboard.