

# XCS229i Problem Set 1

---

**Due Monday, 27 July 2020.**

**Notes:**

1. These questions require thought, but do not require long answers. Please be as concise as possible.
2. If you have a question about this homework, we encourage you to post your question on our Slack channel, at <http://xcs229i-scpd.slack.com/>
3. Familiarize yourself with the collaboration and honor code policy before starting work.
4. For the coding problems, you may not use any libraries except those defined in the provided `environment.yml` file. In particular, ML-specific libraries such as scikit-learn are not permitted.

All students must submit an electronic PDF version of the written questions. We highly recommend typesetting your solutions via L<sup>A</sup>T<sub>E</sub>X, though it is not required. If you choose to hand write your responses, please make sure they are well organized and legible when scanned. The source L<sup>A</sup>T<sub>E</sub>X for all problem sets is available on github. All students must also submit a zip file of their source code. Create a submission using the following bash command (Make sure your conda environment is activated for this to work):

```
zip -j ps1_submission.zip featuremaps/src/featuremap.py
```

You should make sure to (1) restrict yourself to only using libraries included in the `environment.yml` file, and (2) make sure your code runs without errors. Your submission may be evaluated by the auto-grader using a private test set, or used for verifying the outputs reported in the writeup.

**Honor code:** We strongly encourage students to form study groups. Students may discuss and work on homework problems in groups. However, each student must write down the solutions independently, and without referring to written notes from the joint session. In other words, each student must understand the solution well enough in order to reconstruct it by him/herself. In addition, each student should write on the problem set the set of people with whom s/he collaborated. Further, because we occasionally reuse problem set questions from previous years, we expect students not to copy, refer to, or look at the solutions in preparing their answers. It is an honor code violation to intentionally refer to a previous year's solutions.

---

### 1. [20 points] A Simple Neural Network

Let  $X = \{x^{(1)}, \dots, x^{(n)}\}$  be a dataset of  $n$  samples with 2 features, i.e  $x^{(i)} \in \mathbb{R}^2$ . The samples are classified into 2 categories with labels  $y^{(i)} \in \{0, 1\}$ . A scatter plot of the dataset is shown in Figure 1:

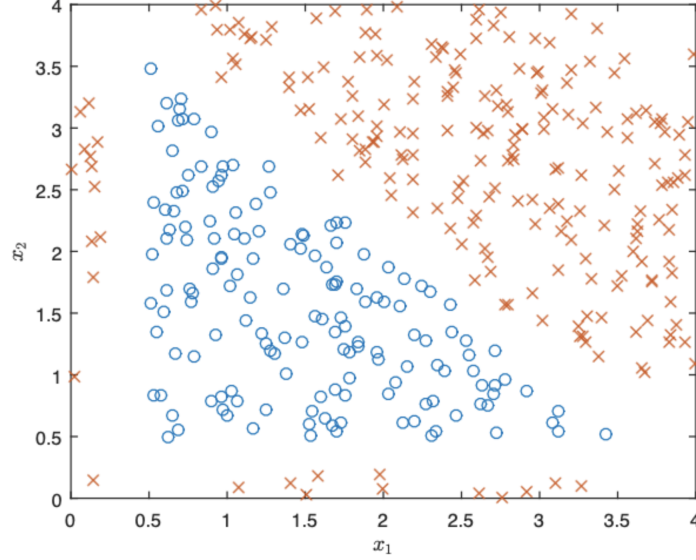


Figure 1: Plot of dataset  $X$ .

The examples in class 1 are marked as “ $\times$ ” and examples in class 0 are marked as “ $\circ$ ”. We want to perform binary classification using a simple neural network with the architecture shown in Figure 2:

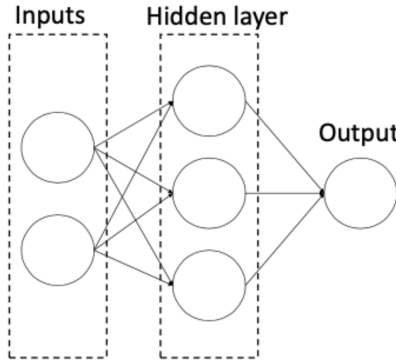


Figure 2: Architecture for our simple neural network.

Denote the two features  $x_1$  and  $x_2$ , the three neurons in the hidden layer  $h_1, h_2$ , and  $h_3$ , and the output neuron as  $o$ . Let the weight from  $x_i$  to  $h_j$  be  $w_{i,j}^{[1]}$  for  $i \in \{1, 2\}, j \in \{1, 2, 3\}$ , and the weight from  $h_j$  to  $o$  be  $w_j^{[2]}$ . Finally, denote the intercept weight for  $h_j$  as  $w_{0,j}^{[1]}$ , and the intercept weight for  $o$  as  $w_0^{[2]}$ . For the loss function, we'll use average squared loss instead of the usual negative log-likelihood:

$$l = \frac{1}{n} \sum_{i=1}^n \left( o^{(i)} - y^{(i)} \right)^2,$$

where  $o^{(i)}$  is the result of the output neuron for example  $i$ .

- (a) [5 points] Suppose we use the sigmoid function as the activation function for  $h_1, h_2, h_3$  and  $o$ . What is the gradient descent update to  $w_{1,2}^{[1]}$ , assuming we use a learning rate of  $\alpha$ ? Your answer should be written in terms of  $x^{(i)}$ ,  $o^{(i)}$ ,  $y^{(i)}$ , and the weights.

**BEGIN PROOF HERE**

Let  $g$  denote the sigmoid function  $g(x) = \frac{1}{1+e^{-x}}$ , and  $h_j^{(i)}$  denote the output of hidden neuron  $h_j$  for sample  $i$ . For shorthand, we denote  $\vec{w}_j^T x^{(i)} = w_{1,j}^{[1]}x_1^{(i)} + w_{2,j}^{[1]}x_2^{(i)} + w_{0,j}^{[1]}$  for  $j \in \{1, 2, 3\}$ , and  $\vec{w}_o^T h^{(i)} = w_1^{[2]}h_1^{(i)} + w_2^{[2]}h_2^{(i)} + w_3^{[2]}h_3^{(i)} + w_0^{[2]}$ . Using chain rule, we have

$$\frac{\partial l}{\partial w_{1,2}^{[1]}} =$$

$$\frac{\partial(o^{(i)})}{\partial w_{1,2}^{[1]}} =$$

$$\frac{\partial(h_2^{(i)})}{\partial w_{1,2}^{[1]}} =$$

Combining everything, we have

$$\frac{\partial l}{\partial w_{1,2}^{[1]}} =$$

and the update rule is

**END PROOF**

- (b) [8 points] Now, suppose instead of using the sigmoid function for the activation function for  $h_1, h_2, h_3$  and  $o$ , we instead used the step function  $f(x)$ , defined as

$$f(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

Is it possible to have a set of weights that allow the neural network to classify this dataset with 100% accuracy?

If it is possible, please provide a set of weights that enable 100% accuracy as a table shown below

If it is not possible, please explain your reasoning in the writeup.

**Hint 1:** There are three sides to a triangle, and there are three neurons in the hidden layer.

**Hint 2:** A solution can be found where all weight and bias parameters take values only in  $\{-1, -0.5, 0, 1, 3, 4\}$ . You are free to come up with other solutions as well.

Neuron	Bias	Value	Weight	Value	Weight	Value	Weight	Value
$h_1$	$w_{0,1}^{[1]}$		$w_{1,1}^{[1]}$		$w_{2,1}^{[1]}$		-	-
$h_2$	$w_{0,2}^{[1]}$		$w_{1,2}^{[1]}$		$w_{2,2}^{[1]}$		-	-
$h_3$	$w_{0,3}^{[1]}$		$w_{1,3}^{[1]}$		$w_{2,3}^{[1]}$		-	-
$o$	$w_0^{[2]}$		$w_1^{[2]}$		$w_2^{[2]}$		$w_3^{[2]}$	

- (c) [7 points] Let the activation functions for  $h_1, h_2, h_3$  be the linear function  $f(x) = x$  and the activation function for  $o$  be the same step function as before.

Is it possible to have a set of weights that allow the neural network to classify this dataset with 100% accuracy?

If it is possible, please provide a set of weights that enable 100% accuracy as a table shown below

If it is not possible, please explain your reasoning in the writeup.

**Hint:** The hints from the previous sub-question might or might not apply.

Neuron	Bias	Value	Weight	Value	Weight	Value	Weight	Value
$h_1$	$w_{0,1}^{[1]}$		$w_{1,1}^{[1]}$		$w_{2,1}^{[1]}$		-	-
$h_2$	$w_{0,2}^{[1]}$		$w_{1,2}^{[1]}$		$w_{2,2}^{[1]}$		-	-
$h_3$	$w_{0,3}^{[1]}$		$w_{1,3}^{[1]}$		$w_{2,3}^{[1]}$		-	-
$o$	$w_0^{[2]}$		$w_1^{[2]}$		$w_2^{[2]}$		$w_3^{[2]}$	

## 2. [20 points] Linear regression: linear in what?

In the first two lectures, you have seen how to fit a linear function of the data for the regression problem. In this question, we will see how linear regression can be used to fit non-linear functions of the data using feature maps. We will also explore some of its limitations, for which future lectures will discuss fixes.

### (a) [5 points] Learning degree-3 polynomials of the input

Suppose we have a dataset  $\{(x^{(i)}, y^{(i)})\}_{i=1}^n$  where  $x^{(i)}, y^{(i)} \in \mathbb{R}$ . We would like to fit a third degree polynomial  $h_\theta(x) = \theta_3 x^3 + \theta_2 x^2 + \theta_1 x + \theta_0$  to the dataset. The key observation here is that the function  $h_\theta(x)$  is still linear in the unknown parameter  $\theta$ , even though it's not linear in the input  $x$ . This allows us to convert the problem into a linear regression problem as follows.

Let  $\phi : \mathbb{R} \rightarrow \mathbb{R}^4$  be a function that transforms the original input  $x$  to a 4-dimensional vector defined as

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \end{bmatrix} \in \mathbb{R}^4 \quad (1)$$

Let  $\hat{x} \in \mathbb{R}^4$  be a shorthand for  $\phi(x)$ , and let  $\hat{x}^{(i)} \triangleq \phi(x^{(i)})$  be the transformed input in the training dataset. We construct a new dataset  $\{(\phi(x^{(i)}), y^{(i)})\}_{i=1}^n = \{(\hat{x}^{(i)}, y^{(i)})\}_{i=1}^n$  by replacing the original inputs  $x^{(i)}$ 's by  $\hat{x}^{(i)}$ 's. We see that fitting  $h_\theta(x) = \theta_3 x^3 + \theta_2 x^2 + \theta_1 x + \theta_0$  to the old dataset is equivalent to fitting a linear function  $h_\theta(\hat{x}) = \theta_3 \hat{x}_3 + \theta_2 \hat{x}_2 + \theta_1 \hat{x}_1 + \theta_0$  to the new dataset because

$$h_\theta(x) = \theta_3 x^3 + \theta_2 x^2 + \theta_1 x + \theta_0 = \theta_3 \phi(x)_3 + \theta_2 \phi(x)_2 + \theta_1 \phi(x)_1 + \theta_0 = \theta^T \hat{x} \quad (2)$$

In other words, we can use linear regression on the new dataset to find parameters  $\theta_0, \dots, \theta_3$ .

Please write down 1) the objective function  $J(\theta)$  of the linear regression problem on the new dataset  $\{(\hat{x}^{(i)}, y^{(i)})\}_{i=1}^n$  and 2) the update rule of the batch gradient descent algorithm for linear regression on the dataset  $\{(\hat{x}^{(i)}, y^{(i)})\}_{i=1}^n$ .

*Terminology:* In machine learning,  $\phi$  is often called the feature map which maps the original input  $x$  to a new set of variables. To distinguish between these two sets of variables, we will call  $x$  the input **attributes**, and call  $\phi(x)$  the **features**. (Unfortunately, different authors use different terms to describe these two things. In this course, we will do our best to follow the above convention consistently.)

$$J(\theta) =$$

Differentiating this objective, we get:

$$\nabla_\theta J(\theta) =$$

The gradient descent update rule is

$$\theta := \theta - \alpha \nabla_{\theta} J(\theta),$$

which reduces here to:

(b) [3 points] **Coding question: degree-3 polynomial regression**

For this sub-question question, we will use the dataset provided in the following files:

`src/featuremaps/{train,valid,test}.csv`

Each file contains two columns:  $x$  and  $y$ . In the terminology described in the introduction,  $x$  is the attribute (in this case one dimensional) and  $y$  is the output label.

Using the formulation of the previous sub-question, implement linear regression with **normal equations** using the feature map of degree-3 polynomials. Use the starter code provided in `src/featuremaps/featuremap.py` to implement the algorithm.

Create a scatter plot of the training data, and plot the learnt hypothesis as a smooth curve over it. Submit the plot in the writeup as the solution for this problem.

*Remark:* Suppose  $\hat{X}$  is the design matrix of the transformed dataset. You may sometimes encounter a non-invertible matrix  $\hat{X}^T \hat{X}$ . For a numerically stable code implementation, always use `np.linalg.solve` to obtain the parameters directly, rather than explicitly calculating the inverse and then multiplying it with  $\hat{X}^T y$ .

Your plot should look similar to the following:

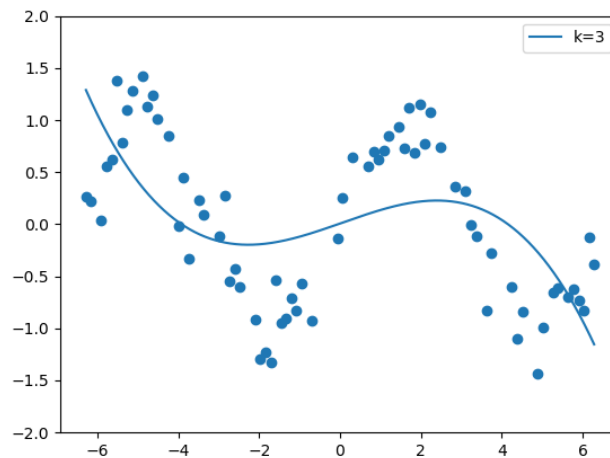


Figure 3: Polynomial regression with degree 3

(c) [4 points] **Coding question: degree- $k$  polynomial regression**

Now we extend the idea above to degree- $k$  polynomials by considering  $\phi : \mathbb{R} \rightarrow \mathbb{R}^{k+1}$  to be

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^k \end{bmatrix} \in \mathbb{R}^{k+1} \quad (3)$$

Follow the same procedure as the previous sub-question, and implement the algorithm with  $k = 3, 5, 10, 20$ . Create a similar plot as in the previous sub-question, and include the hypothesis curves for each value of  $k$  with a different color. Include a legend in the plot to indicate which color is for which value of  $k$ .

Submit the plot in the writeup as the solution for this sub-problem. Observe how the fitting of the training dataset changes as  $k$  increases. Briefly comment on your observations in the plot.

Your plot should look similar to the following:

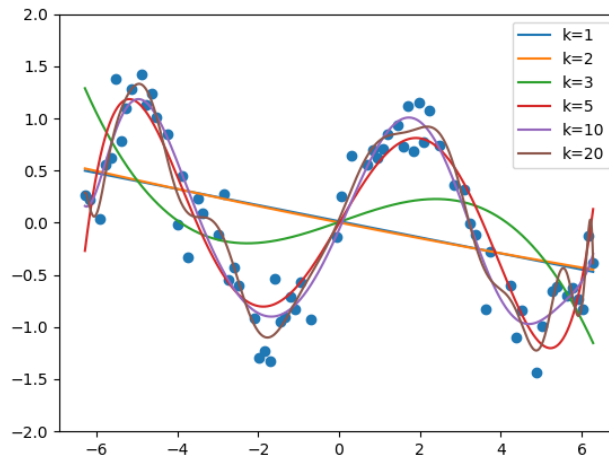


Figure 4: Polynomial regression with kernel sizes 1,2,3,5,10 and 20

Provide your explanation of the fit for each  $K$  value here:

(d) [4 points] **Coding question: other feature maps**

You may have observed that it requires a relatively high degree  $k$  to fit the given training data, and this is because the dataset cannot be explained (i.e., approximated) very well by low-degree polynomials. By visualizing the data, you may have realized that  $y$  can be approximated well by a sine wave. In fact, we generated the data by sampling from  $y = \sin(x) + \xi$ , where  $\xi$  is noise with Gaussian distribution. Please update the feature map  $\phi$  to include a sine transformation as follows:

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^k \\ \sin(x) \end{bmatrix} \in \mathbb{R}^{k+2} \quad (4)$$

With the updated feature map, train different models for values of  $k = 0, 1, 2, 3, 5, 10, 20$ , and plot the resulting hypothesis curves over the data as before.

Submit the plot as a solution to this sub-problem. Compare the fitted models with the previous sub-question, and briefly comment about noticeable differences in the fit with this feature map.

Your plot should look similar to the following:

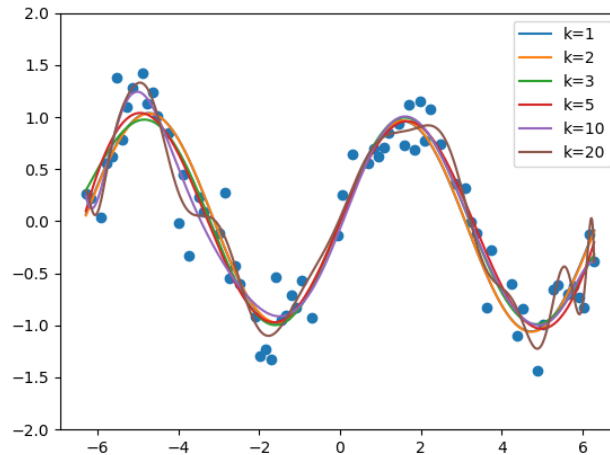


Figure 5: Polynomial regression with other features with kernel sizes 1,2,3,5,10 and 20

Provide your comparison of the fitted models here:

(e) [4 points] **Overfitting with expressive models and small data**

For the rest of the problem, we will consider a small dataset (a random subset of the dataset you have been using so far) with much fewer examples, provided in the following file:

`src/featuremaps/small.csv`

We will be exploring what happens when the number of features start becoming bigger than the number of examples in the training set. Run your algorithm on this small dataset using the



following feature map

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^k \end{bmatrix} \in \mathbb{R}^{k+1} \quad (5)$$

with  $k = 1, 2, 5, 10, 20$ .

Create a plot of the various hypothesis curves (just like previous sub-questions, both with and without the sinusoidal features). Observe how the fitting of the training dataset changes as  $k$  increases. Submit the plot in the writeup and comment on what you observe.

**Remark:** The phenomenon you observe where the models start to fit the training dataset very well, but suddenly “goes wild” is due to what is called *overfitting*. The intuition to have for now is that, when the amount of data you have is small relative to the expressive capacity of the family of possible models (that is, the hypothesis class, which, in this case, is the family of all degree  $k$  polynomials), it results in overfitting.

Loosely speaking, the set of hypothesis function is “very flexible” and can be easily forced to pass through all your data points especially in unnatural ways. In other words, the model explains the noises in the training dataset, which shouldn’t be explained in the first place. This hurts the predictive power of the model on test examples. We will describe overfitting in more detail in future lectures when we cover learning theory and bias-variance tradeoffs.

Your plot should look similar to the following (only polynomial features are shown):

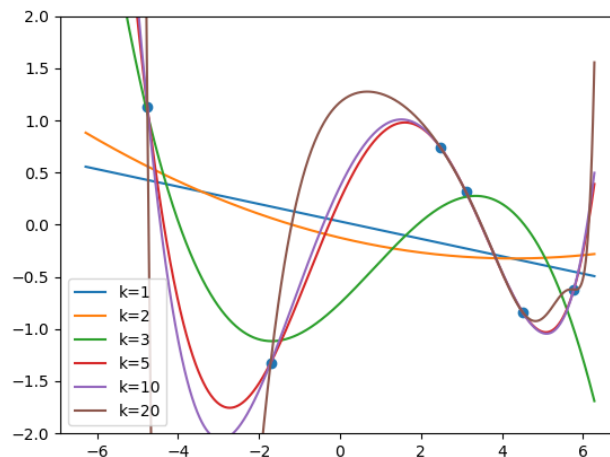


Figure 6: Polynomial regression with kernel sizes 1,2,3,5,10 and 20 on small dataset

Provide your observations of how the value of  $k$  fits the smaller training set here: