Homework 2

Austin Gardiner

September 29, 2023

1 Avoiding Machine Learning Pitfalls

Summaries of sections 2 and 3 from the article.

1.1 2: Before you start to build models

This section is focused on evaluating data before making a machine learning model. The first step is to examine both the quality and quantity of your data. Bad data will generate a bad model, and not having enough data will result in a model that does not generalize well. There are circumstances where enough data does not exist, and in these cases it is often possible to use data augmentation techniques, which is usually effective for small data sets. In the case of classification, sometimes some classes will have significantly more training points than others, leading to a class imbalance. In these cases it is important to use data augmentation techniques and to avoid using accuracy to test the model.

Additionally, it is important to reach out to domain experts and to read literature before determining the type of model that will be trained. This can help avoid wasting time with types of models that will perform poorly for a desired task. Domain experts can also help you determine which features will be most important in training your model. Surveying the literature can also help you to avoid repeating work that has already been done, and can help you discover new unexplored areas. It is also useful to think of how the model will be deployed. Applications where a quick classification is required or where little processing power is available will limit the complexity of your model.

1.2 3: How to reliably build models

The first focus of this section is to separate test data from training data. It is important that test data is kept separate so that the model is not training itself to a specific dataset. One way to help avoid this is to split data into training, validation, and testing datasets. This should be done before the feature selection and hyperparameter training steps. This way the test dataset does not leak into the training step. Once the data has been segmented, a bunch of different types of models should be tested. Because all perform differently for different tasks

and types of data, a large range of applicable model types should be tried. However, some model types should be avoided, depending on the types of data used in training the model. For example, a timeseries model requires a model that uses past states to predict future ones would require a model that can use sequential data. Last, it is important to optimize hyperparameters, and it is best to do this using a structured optimization method.

2 Convexity

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y) \tag{1}$$

2.1 Global Minimizer of a Strictly Convex Function

A strictly convex function satisfies the following strict inequality for all x, y, and t.

$$f(tx + (1-t)y) < tf(x) + (1-t)f(y) \tag{2}$$

Assuming that there are two global minimizers, x and y, and using a value of $t = \frac{1}{2}$, we get

$$f(\frac{x+y}{2}) < \frac{f(x)+f(y)}{2} \tag{3}$$

This new formula states that the value of f at the midpoint of x and y must be less than the average of f(x) and f(y). Because x and y are both global minimizers, f(x) and f(y) must be equal, so the right hand side of the equation simplifies to f(x). Therefore, the value of the function at the midpoint of the two minimizers must be less than the global minimum, which means x and y are not actually global minimizers. Therefore, for a strictly convex function there can only be one global minimizer.

2.2 Sum of two Convex Functions

N/A

2.3 Conditions of Convex vs Strictly Convex

For the following function where A is a symmetric dxd matrix, derive the Hessian. Under what conditions on A is f convex and strictly convex? Because A is symmetric, $A^T = A$.

$$f(x) = \frac{1}{2}x^T Ax + b^T x + c \tag{4}$$

$$\frac{\partial f}{\partial x} = x^T A + b^T \tag{5}$$

$$\frac{\partial^2 f}{\partial x^2} = \nabla^2 f = A \tag{6}$$

The hessian of f is derived as shown above in eq. 6. For a function to be convex, its hessian must be positive semi-definite and for it to be strictly convex, its hessian must be positive definite and this must be true at all values of x. Because the hessian of f is just the A matrix and therefore has no dependence on x, this means that A must be positive semi-definite or positive definite for convex or strictly convex respectively. This can be checked in the normal ways: the definition, checking eigenvalues, or the leading principle minors test.

2.4 Prove that $f(x) = x^3$ is not convex using the definition

Intuitively, it can be seen that the function x^3 is not convex below x = 0. This can be proven using the definition of convexity shown above in eq. 1. Using values of x = -1, y = 0, and t = $\frac{1}{2}$ the left and right sides of the equation are solved below.

$$f(tx + (1-t)y) = f(\frac{1}{2}(-1) + (1-\frac{1}{2})(0)) = f(-\frac{1}{2}) = -\frac{1}{8} \tag{7}$$

$$tf(x) + (1-t)f(y) = \frac{1}{2}f(-1) + (1-\frac{1}{2})f(0)) = \frac{1}{2}f(-1) = -\frac{1}{2} \tag{8}$$

Comparing these solutions, we get the following equation is not true.

$$\frac{-1}{8} \leq \frac{-1}{2} \tag{9}$$

Because this is not true, we know that the function x^3 is not convex.

2.5 Proof using twice continuously differentiable

One of the necessary conditions to prove convexity is that the hessian must be positive for all values of x in the domain. For the function x^3, the hessian is equal to the second derivative, which is $f'' = 6x$. f'' is negative for all values less than 0 and is equal to zero at the critical point found at $x = 0$.

$$f''(x) < 0 : x < 0 \tag{10}$$

Therefore, x^3 is not convex.

2.6 Concavity of ln(x)

Using the fact that a function f is concave if -f is convex, we are proving that ln(x) is concave for $x > 0$. The first step is to set g = -f = -ln(x) and then calculate the first and second derivative.

$$f(x) = ln(x), f'(x) = \frac{1}{x}, f''(x) = \frac{-1}{x^2}$$

$$g(x) = -ln(x), g'(x) = \frac{-1}{x}, g''(x) = \frac{1}{x^2}$$

Looking at g(x), if we set g'(x) = 0, we find that $lim_{x\to\infty} g'(x) = 0$, but excluding infinity, there is no true global min. However, testing the second derivative, we see that for $x > 0, g''(x) > 0$. Using these two conditions, it can be seen that -ln(x) is convex for $x > 0$. This can also seen by plugging any values for x, y, and t into the definition shown in eq. 1.

With this proven, we know that ln(x) is concave, according to the principle that the negative of a convex function is concave.

2.7 Affine Function

$$f(x) = ax + b \tag{11}$$

Using the definition of convex functions shown in eq. 1 and using a similar definition for concavity with the $\leq$ replaced with $\geq$, the affine function is proved to be both convex and concave. The left hand side is expanded first, followed by the right hand side of the convex definition formula. The following equations work for any combination of x, y, and t.

$$LHSConv : f(tx + (1-t)y) = a(tx + (1-t)y) + b = atx + (1-t)ay + b$$

$$RHSConv : tf(x) + (1-t)f(y) = atx + tb + (1-t)(ay + b) = atx + (1-t)ay + b$$

$$atx + (1-t)ay + b = atx + (1-t)ay + b \tag{12}$$

Because both sides of the equation are exactly equal, the function is both convex and concave, but not strictly convex or strictly concave.

By checking the first derivative, we see that f'(x) = a, which is never equal to zero unless a = 0. Therefore there is no global min or max, which makes intuitive sense as this is a linear function. However, if a = 0, every point is both a global min and and global max.

There are no other functions that are both concave and convex and twice continuously differentiable as the second derivative of such a function must be exactly equal to zero at all points. The only functions for which this is true are the affine function and a constant value function, which is just the affine function with a = 0.

3 Sigmoids and Perceptrons

$$z = wx + b \tag{13}$$

3.1 Scalar Multiplication in a Perceptron Network

In a perceptron network, the output of a neuron depends only on the sign of z, which as defined in eq. 13 is the sum of the inputs (or outputs from the previous layer) multiplied by their corresponding weights, plus the bias for the neuron. If this value is greater than or equal to zero, the output of the neuron is a 1, if z is negative, the output is 0.

Now, a constant scalar, $c > 0$, is brought in and multiplies with all weights and biases in the network and we want to know the effect on the network. Starting by looking at the effect on a single neuron, the scalar is shown as multiplied by the weight vector and bias and results in eq. 14. Because the weights and bias are both multiplied by the same scalar, that value can be brought outside of the parentheses, and the equation becomes equal to cz.

$$cwx + cb = c(wx + b) = cz \tag{14}$$

Because c is non-zero and positive, it will not change the sign of z. Therefore the output of the neuron remains unchanged, as shown in eq. 15.

$$p(cz) = p(z) \tag{15}$$

Because the scalar does not change the output of any individual neuron, the inputs to the next layer of neurons remain unchanged. This pattern of unchanged inputs and outputs continues through all layers of the network. Thus the overall output of the network will remain unchanged as there are no changes to cascade through the network.

3.2 Scalar Multiplication in a Sigmoid Network

The sigmoid activation function is a smooth function, as such, a scalar will change the output of individual neurons. Now the goal is to see what happens as the scalar, c, approaches infinity. Now the input to the sigmoid function, z, is determined according to eq. 16, where the sign is dependent on the sign of z (if z is positive, $cz \to +\infty$, if z is negative $cz \to -\infty$). This equation assumes that $z \neq 0$, with this case described later.

$$\lim_{c\to\infty} cwx + cb = \lim_{c\to\infty} cz = \pm\infty \tag{16}$$

Plugging this value into the sigmoid function, we see the following behavior.

$$s(z) = \frac{1}{1 + e^{-z}} \tag{17}$$

$$s(-\infty) = \frac{1}{1 + e^{\infty}} = 0, s(\infty) = \frac{1}{1 + e^{-\infty}} = 1 \tag{18}$$

Based on eq. 18 this new model will behave just like a standard perceptron network, where the output is based solely on the sign of z.

This behavior does not hold when z = 0. Plugging 0 into eq. 17, we get

$$s(0) = \frac{1}{1+e^0} = \frac{1}{2} \tag{19}$$

For a perceptron network, with our definition that p(0) = 1 (this can sometimes be defined to be equal to 0) this value does not equal the $\frac{1}{2}$ found in eq. 19. In this case the network does not behave like a perceptron and this change will cascade through the network. But if $z \neq 0$ for all neurons, the output will be the same as a perceptron network with the same structure.

3.3 Calculating Output in a Multi-Layer Perceptron and Sigmoid Network

Using the MLP setup shown in Figure 1 of the assignment document, the perceptron and sigmoid outputs are calculated for all possible combinations of $x_1, x_2, and x_3$. Perceptron values are shown in the p column and sigmoid values rounded to five decimals are shown in the s column. The procedure for generating this table is included in the code block below.

```
Table
x1 x2 x3| p s
------------------
0  0  0 | 0 0.56927
0  0  1 | 1 0.58501
0  1  0 | 1 0.62246
0  1  1 | 1 0.63314
1  0  0 | 1 0.56987
1  0  1 | 0 0.57508
1  1  0 | 1 0.61733
1  1  1 | 1 0.62831
```

Figure 1: Perceptron and Sigmoid Outputs for each x combination

```
import math

inputs = [0, 1]
wi = [0.6, -0.7]
wj = [0.5, 0.4]
wk = [-0.6, 0.8]
w21 = 1
```

```
w22 = 1

b1 = -0.4
b2 = -0.5
b3 = -0.5

def sigmoid(z):
    return 1 / (1 + math.exp(-z))

def perceptron(z):
    if z >= 0:
        return 1
    else:
        return 0


x = []
p = []
s = []

for i in inputs:
    for j in inputs:
        for k in inputs:
            z1 = i * wi[0] + j * wj[0] + k * wk[0] + b1
            z2 = i * wi[1] + j * wj[1] + k * wk[1] + b2

            # For perceptron
            p1 = perceptron(z1)
            p2 = perceptron(z2)
            pz3 = p1 * w21 + p2 * w22 + b3
            p3 = perceptron(pz3)

            # For sigmoid
            n1 = sigmoid(z1)
            n2 = sigmoid(z2)
            nz3 = n1 * w21 + n2 * w22 + b3
            n3 = sigmoid(nz3)

            # Store results
            x.append([i, j, k])
            p.append(p3)
            s.append(n3)

print("Perceptron")
for index, xval in enumerate(x):
    print(f"x=[{xval[0]}, {xval[1]}, {xval[2]}]: {p[index]}")
```

```
print("\nSigmoid")
for index, xval in enumerate(x):
    print(f"x=[{xval[0]}, {xval[1]}, {xval[2]}]: {round(s[index], 5)}")

print("\nTable")
print("x1 x2 x3| p s")
print("-------------------")
for index, xval in enumerate(x):
    print(f"{xval[0]}  {xval[1]}  {xval[2]} | {p[index]} {round(s[index], 5)
```

4 Regression, Classification, or Both?

4.1 Determine which kind of tree is in a picture

Classification. This is a classic classification problem as it deals with assigning images a label that fit into different classes.

4.2 Predict when someone will die based on their current health conditions

Regression. This task deals with taking condition data and predicting a numerical label on a continuous spectrum.

4.3 Predict the likelihood that someone will survive for more than one year after a cancer diagnosis based on their current health conditions

Both. This is primarily a classification task where you are classifying them as likely to live more than a year or not. But predicting the likelihood is a regression task as it is a continuous spectrum.

4.4 Predict how many stars out of five a person will rate a product on Amazon based on their previous reviews and browsing history

Both. Assuming that the rating must be a discrete star value, a model built for this task would classify the product as receiving a 1, 2, 3, 4, or 5 star rating, therefore fitting better as a classification problem as there are 5 distinct possible classes. This would also hold true if half star ratings were allowed. However, if the rating is on a continuous scale, it would be a regression problem.

4.5 Text prediction

Classification. If there are a set number of possible words to be used next, this becomes a more standard classification problem. However, if any word may be used, it become more complicated, but still fits better as a classification problem than a regression problem as there are a finite number of discrete words that can be placed next in the sequence.

5 Building a Regression Model

5.1 Linear Regression Model

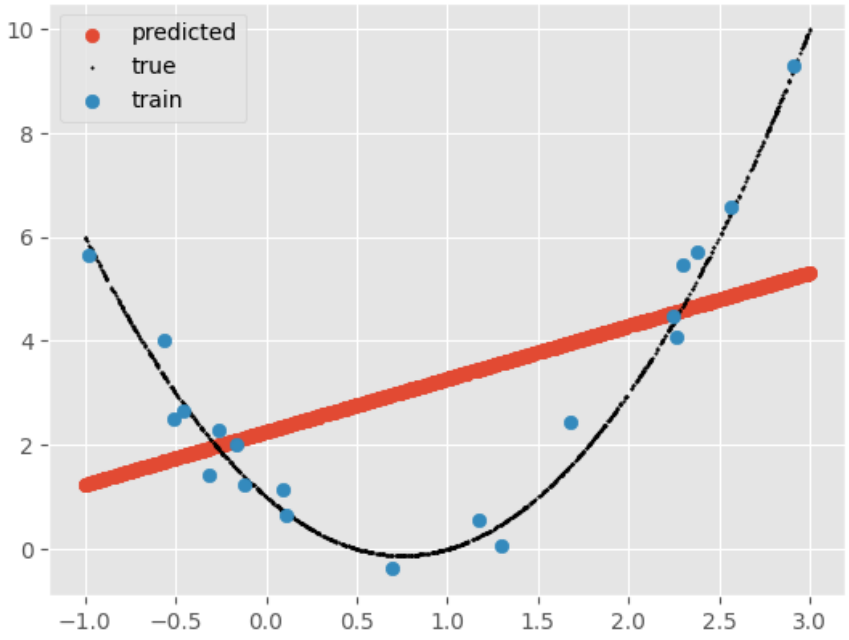

Figure 2: Training Data with Linear Regression fit on Test Data

5.2 Second Order Regression Model

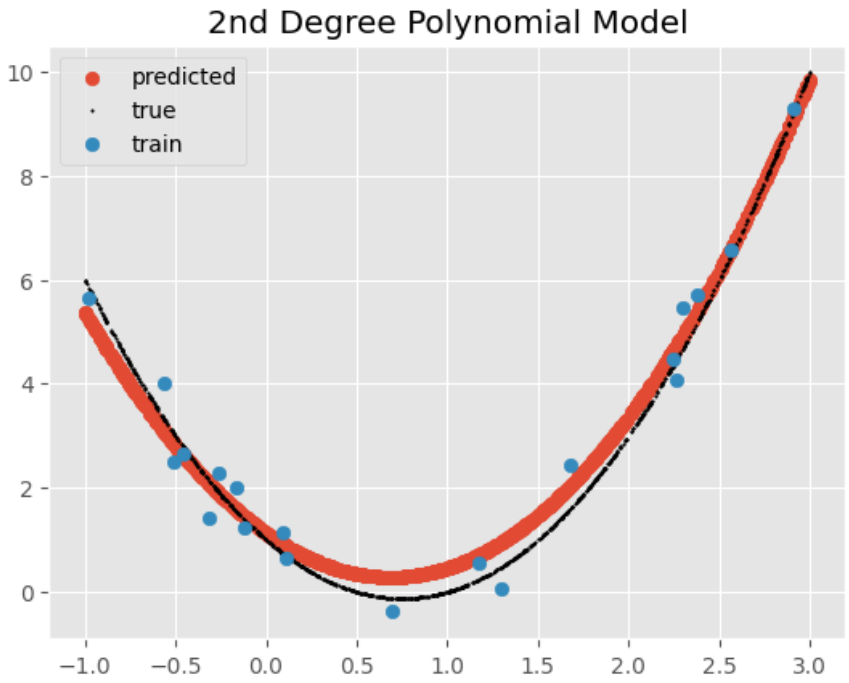

Figure 3: Second Order Regression fit on Test Data

$$Actual : y = 2x^2 - 3x + 1 \tag{20}$$

$$Predicted : y = 1.78884x^2 - 2.46064x + 1.12957 \tag{21}$$

While the weights and biases aren't perfectly predicted, the model gets close. As the number of epochs increases, the predictions get closer to the true values.

5.3 5th Order Regression Model

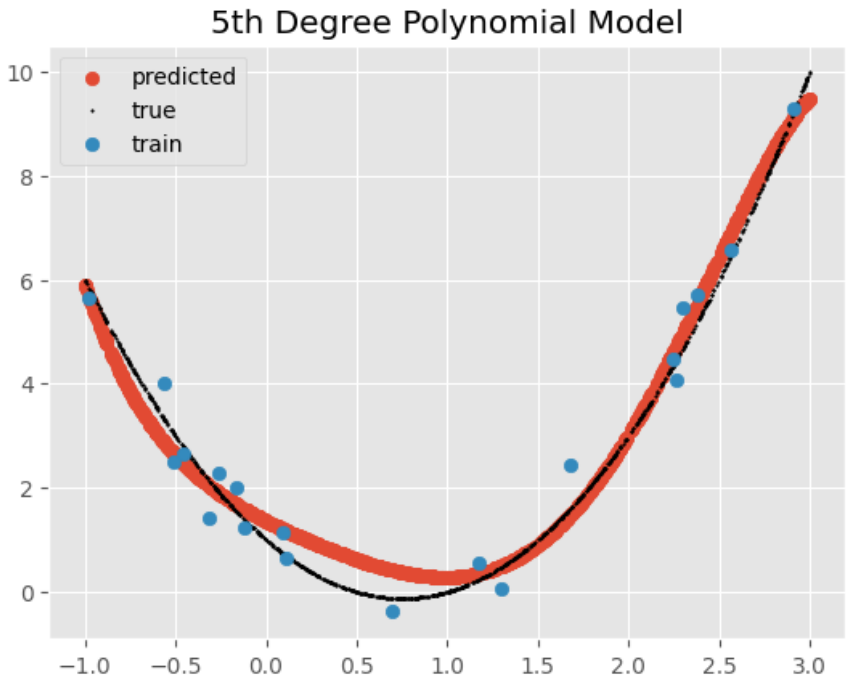

Figure 4: 5th Order Regression fit on Test Data

The learning rate had to be lowered to allow the model to actually train. However, even with a tuned learning rate, the model performance was not great with just 5,000 epochs for training. Using a learning rate of 2.2e-4 and 25,000 epochs, a better fit was obtained, as shown above.

5.4 Regularization at Multiple Noise Settings

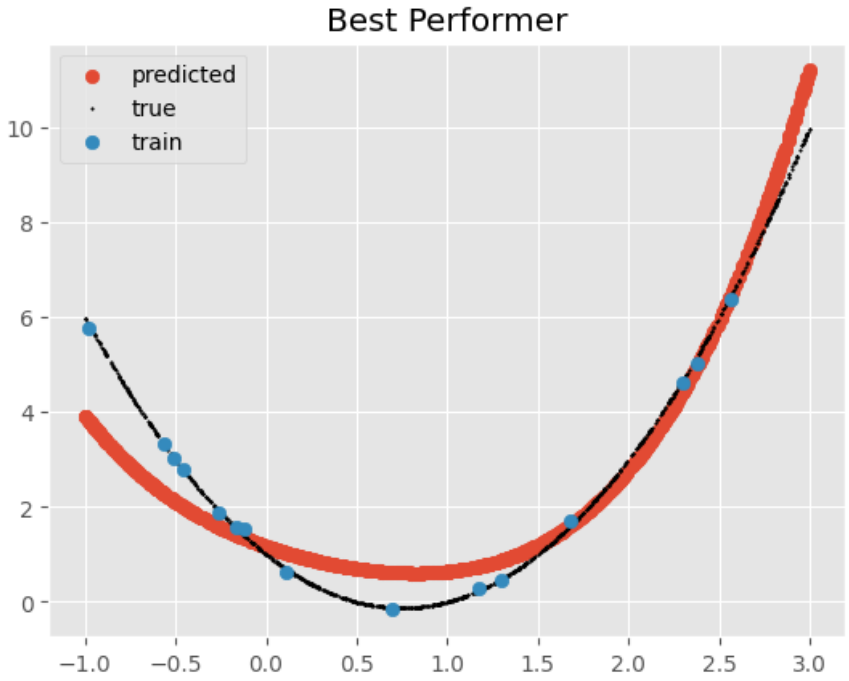

Figure 5: Best Performer in the Regularization Test

$$\sigma = 0.1, N = 15, MSE = 0.4965 \tag{22}$$

At lower values of σ, the test error, measured using Mean Squared Error (MSE), was lower when trained on less samples (N=15). However, at $\sigma = 1$ the model performed noticeably better with N=100 samples. In all cases, the model performed better when the weight decay parameter was set to 0. More results are shown in the code.