2022-06-01 Wednesday

NAME: Ravi Moelchand, 7463332

Topics covered since the last tutorial:

- Bias and Variance (Lecture 8)
- Overfitting (Lecture 9)

And we will also cover this mathematics topic:

- Marginalization, Random Variables (week 5)

If you write your answers directly into the notebook, it is preferred that you generate a .pdf file for submission

This tutorial contains 10 problems. Please submit one solution per person

Bias and Variance

Are the following propositions true or false? Explain.

1. Someone comes to you with a learning problem, and hands you the associated data set. Obviously, you do not know the target function. A good way to analyse the performance of the learning model you want to apply, is to calculate the exact bias and variance for the given learning problem.

answer

True, bias and variance are both indicators for the performance of a learning model. So, calculating those is a very good way to analyse the performance of learning model.

2. Suppose a learning model has hypothesis set $\mathcal{H} = \{h_1(x) = 0, h_2(x) = 2\}$. So, the hypothesis set contains two constant functions. It is possible that the average hypothesis in a bias-variance analysis of the learning model: $\bar{g}(x) = 2$.

answer

True, we base the average hypothesis on different datasets in a bias-variance analysis. So it depends on what is learned, which in this case is $\bar{g}(x) = 2$.

3. The bias-variance trade-off implies that the bias and variance cannot be low at the same time.

answer

True, the negative feedback relation between bias and variance makes that they cannot be low at the same time.

Overfitting

Carry out the following exercises from Learning From Data (Abu-Mostafa et al, 2012):

4. Exercise 4.3

answer

a. The algorithm will try to fit the noise. Which means that there is a higher tendency to overfit. So, as the complexity of f increases, the deterministic noise will rise as well.

b. -

5. Problem 4.1

6. Problem 4.4 parts a-c. This question is quite hard, and assumes some knowledge about non-linear transforms, e.g. in (b). We do not expect you to write any code here, but rather to explain using words and mathematics.

---

7. Carry out exercise 1 from the online Homework 6 exercises from Learning From Data (Abu-Mostafa et al, 2012): https://work.caltech.edu/homework/hw6.pdf

Marginalization, Random Variables

8. Two (distinct) dice, die 1 and die 2, are thrown. What is the set of possible outcomes?

9. Consider two random variables $X = D_1$ the number on die 1 and the difference between the die values $Y = |D_1 - D_2|$. The joint probability distribution for (X, Y) is shown on the following table:

drawing

What are the marginal probabilities for the random variable Y as defined above?

10. A box contains 8 tickets. Two are marked 1, two marked 2, two marked 3, and two marked 4. Tickets are drawn at random from the box without replacement until a number appears that has appeared before. Let X be the number of draws that are made. Make a table to display the probability distribution of X. You should create a table like this one:

drawing

Recommended

Once you've finished the previous part, we encourage you to practice your understanding of the subjects covered in this tutorial with the following recommended exercises. Most of these exercises involve a little programming, so we encourage you to try them out so that you can apply these principles in practice.

Bias and Variance

The following assignments are variants on Problem 2.23 of Learning From Data (Abu-Mostafa et al, 2012).

Note In the assignments you will be using Python-code (see the end of this document). However, these are **not** programming assignments. You will be applying the code as a kind of advanced calculator, and not do (much) coding yourself.

In the following assignments, we are going to explore regression learning problems for which the following things hold:

- The output space is, self-evidently, real-valued, i.e. $\mathcal{Y} = \mathbb{R}$.
- The target function f_{target} is known.
- The input probability distribution is uniform on $\mathcal{X} = [x_{\min}, x_{\max}]$.
- The training set $\mathcal{D}$ has only two data points (picked independently).
- The learning algorithm picks the hypothesis that minimises the in sample mean squared error.

Now, consider the following list of notions, as explained in Learning From Data (Abu-Mostafa et al, 2012), section 2.3.1:

> **List 1**
>
> 1. h_{real}: the best hypothesis that approximates f_{target} in the mean squared error sense,
> 2. $\mathbf{g}(x)$: the expected value (with respect to $\mathcal{D}$) of the hypothesis that the learning algorithm produces for f_{target}.
> 3. $\mathbb{E}(E_{\text{out}}(g^{(\mathcal{D})}))$: the expected out of sample error
> 4. var: the variance
> 5. bias: the bias

In [ ]:

```
### The code is an adaptation by Chide Groenouwe of code from https://github.com/niuers

import matplotlib.pyplot as plt
import numpy as np
from functools import partial
import math
from matplotlib.font_manager import FontProperties

def sample(lb, ub, sz):
    # sample randomly from a uniform distribution
    return lb + np.random.random_sample((sz,))*(ub-lb)

# Learning models for different hypothesis sets. Each function below picks the hypothesis
# with the lowest mean squared error for the given data set with input points (x1, x2).
# Note, x1 and x2 here represent the first and the second input datapoint, not for the first and second inputva

# hypothesis set h(x) = 1
def gd_1(x, target_func, x1, x2):
    return 1

# hypothesis set h(x) = ax + b
def gd_ax_plus_b(x, target_func, x1, x2):
    # NOTE: this function only works if x1 != x2
    #hypothesis found by learning algorithm
    a = (target_func(x2)-target_func(x1))/(x2-x1)
    b = (x2*target_func(x1)-x1*target_func(x2))/(x2-x1)
    return a*x + b

# hypothesis set h(x) = ax
def gd_ax(x, target_func, x1, x2):
    #hypothesis found by learning algorithm
    a = (x1*target_func(x1)+x2*target_func(x2))/(x1**2+x2**2)
    return a*x

# hypothesis set h(x) = b
def gd_b(x, target_func, x1, x2):
    #hypothesis found by learning algorithm
    return 0.5*(target_func(x1) + target_func(x2))

# )

def avg_g(x, gdfunc, num_samples, target_func, x_min, x_max):
    #compute the average hypothesis \bar{g} at given point x
    bias_at_x = 0
    gd_funcs = []
    for i in range(num_samples):
        #generate 2 sample data each time
        x1, x2 = sample(x_min, x_max, 2)
        v = gdfunc(x, target_func, x1, x2)
        gd_funcs.append(v)

    average_gfunc_at_x = np.mean(gd_funcs)
    #print('x: ', x, 'average_gfunc_at_x: ', average_gfunc_at_x)
    variance_gfunc_at_x = np.var(gd_funcs)
    bias_at_x = (average_gfunc_at_x - target_func(x))**2
    return average_gfunc_at_x, variance_gfunc_at_x, bias_at_x

# Numerically approximate the expected value of variance, bias and out-of-sample error
def calc_bias_var_eout(gd_func, target_func, x_min, x_max, num_data_samples, num_x_samples):
    variances, biases, eouts = [], [], []
    for i in range(num_x_samples):
        x = sample(x_min, x_max, 1)
        _, variance, bias = avg_g(x, gd_func, num_data_samples, target_func, x_min, x_max)
        variances.append(variance)
        biases.append(bias)

        # Compute the expected value of out-of-sample error w.r.t. data
        eout_on_data = []
        for i in range(num_data_samples):
            x1, x2 = sample(x_min, x_max, 2)
            v= gd_func(x, target_func, x1, x2)
            eout_on_data.append((v-target_func(x))**2) # (g^((\mathcal{D}))(x) - f(x))**2

        eout_data_avg = np.mean(eout_on_data)
        eouts.append(eout_data_avg)



    variance = np.mean(variances)
    bias = np.mean(biases)
    eout = np.mean(eouts)
    print('The variance is: ', variance)
    print('The bias is: ', bias)
    print('The expected out-of-sample error is: ', eout)
    print('The variance+bias is: ', variance+bias)


    xs = np.arange(x_min, x_max, 0.01)
    true_f, avg_gf, var_gf, ubs, lbs = [],[], [], [], []
    for x in xs:
        true_f.append(target_func(x))
        mean_g, var_g, bias_g = avg_g(x, gd_func, num_data_samples, target_func, x_min, x_max)
        avg_gf.append(mean_g)
        var_gf.append(var_g)
        ubs.append(mean_g + np.sqrt(var_g))
        lbs.append(mean_g - np.sqrt(var_g))

    plt.plot(xs, true_f, color='red', label='Problem 2.23: True Function')
    plt.plot(xs, avg_gf, color='green', label='Problem 2.23: Average Hypothesis g_bar')
    plt.plot(xs, ubs, color='blue', label='Problem 2.23: Upper bound of the average hypothesis')
    plt.plot(xs, lbs, color='blue', label='Problem 2.23: Lower bound of the average hypothesis')
    legend_x = 2.0
    legend_y = 0.5
    plt.legend(('Problem 2.23: Target function',
                'Problem 2.23: Average hypothesis g_bar',
                'Problem 2.23: g_bar(x) + standard_deviation(x)',
                'Problem 2.23: g_bar(x) - standard_deviation(x)'),
               loc='center right', bbox_to_anchor=(legend_x, legend_y))
```

Suppose: $\mathcal{H} = \{h|h(x) = 1\}$, so a hypothesis set that consists of only one hypothesis: $h(x) = 1$. $f_{\text{target}}(x) = e^x$ and $\mathcal{X} = [0, 4]$.

Rec1. Use `plot_function` on the given domain $\mathcal{X}$ to see what f_{target} looks like. Intuitively try to estimate what the values will be for each of the expressions in List 1. Now determine each of the expressions in List 1 analytically. So not numerically, nor with code, but with mathematical derivations. Each time, start with the definition of each of the expressions as they are presented in Section 2.3.1 of Learning From Data (Abu-Mostafa et al, 2012). Substitute what is known in this particular case, and work your way towards the solution. Hint 1: Hint 2: at some steps, you can use the change-of-variable rule for expectations: see A Modern Introduction to Probability and Statistics Understanding Why and How- Dekking et al (2005): https://utrechtuniversity-on-worldcat-org.proxy.library.uu.nl/oclc/262680588, Section 7.3, the gray block, the rule for continuous variables.

Rec2. Now, determine these values also experimentally, by using the Python-function `calc_bias_var_eout` in the code. Show the function-calls you make and the resulting plots.

Rec3. Compare the analytical and the experimental results: are they close to each other? Also, compare with your intuitions, do they tally? Now, conceptually (and not in terms of formulas or derivations) explain why you get the values you did for each of the items in List 1.

Now, suppose $\mathcal{H} = \{h|h(x) = ax + b \text{ and } a, b \in \mathbb{R}\}$ (the set of hypotheses of the form $ax + b$, where a and b are real numbers), with the same f_{target} and $\mathcal{X}$ as in the previous assignment.

Rec4. Compare your intuitions with the experiments. Do they tally? Now, conceptually (and not in terms of formulas or derivations) explain the differences between these results and those of the previous question.

Rec5. In the last problem we worked with 2 datapoints per data set. Suppose you would have worked with 3 datapoints instead. What do you think would happen with the bias and variance in the answer to the previous assignment? Explain. Tip: look at the changes you have to make to the learning models, what hypothesis will it pick based on 3 datapoints instead of 2?

We return the original setting with 2 datapoints. In the code there are also some other hypothesis sets available in addition to the ones used so far. For each of the following states of affairs: define a target function f_{target}, define an input space interval $\mathcal{X} = [x_{\min}, x_{\max}]$ and choose one of the hypothesis sets, such that the state of affairs becomes a reality. Each time, choose a different hypothesis set. You can define any target function you want. So, use four hypothesis sets in total. Explain your answer conceptually (no formulas needed), and show the function-call that you made to verify your answer.

Rec6. low bias and a low variance

Rec7. high bias and a low variance

Rec8. low bias and a high variance

Rec9. a high bias and a high variance

Overfitting

Rec10. Carry out problem 4.4, parts d-f, from Learning From Data (Abu-Mostafa et al, 2012). This question is quite hard, and assumes some knowledge about non-linear transforms, e.g. in (b). Therefore, when you are not able to solve a sub-question that is needed for answer the sub-questions by which it is followed, look up its solution, or use a Python library that also solves it. However, try to minimise the amount of peeking and solve as much as possible yourself.

Carry out the following exercises from Google's machine-learning course:

Rec11. https://developers.google.com/machine-learning/crash-course/validation/check-your-intuition

Rec12. https://developers.google.com/machine-learning/crash-course/training-and-test-sets/playground-exercise

Rec13. https://developers.google.com/machine-learning/crash-course/validation/programming-exercise (see the link "Validation Sets and Test Sets Colab exercise" on the page.)