Benjamin Ye

CS/CNS/EE 156a: Learning Systems (Fall 2023)

December 1, 2023

**Final Exam**

|  |  |  |  |
| --- | --- | --- | --- |
| **Problem** | **Answer** | **Problem** | **Answer** |
| 1 | [e] | 11 | [c] |
| 2 | [d] | 12 | [c] |
| 3 | [d] | 13 | [a] |
| 4 | [d] | 14 |  |
| 5 | [a] | 15 |  |
| 6 | [b] | 16 |  |
| 7 | [d] | 17 |  |
| 8 | [b] | 18 |  |
| 9 | [e] | 19 |  |
| 10 | [a] | 20 |  |

**Nonlinear Transforms**

1. The polynomial transform of order applied to of dimension resuilts in a space of what dimensionality (not counting the constant coordinate or )?

**Answer: [e] None of the above**

For a second-order polynomial transformation, we have :

For a third-order polynomial transformation, we have :

And from Homework 5 Problem 3, we have for a fourth-order polynomial transformation.

By interpolating the dimensionalities of these lower-order examples, it is clear to see that

That is, each additional order adds terms. Therefore,

**Bias and Variance**

1. Recall that the average hypothesis was based on training the same model on different data sets to get , and taking the expected value of with respect to to get . Which of the following models could result in ?

**Answer: [d] is the logistic regression model**

Choices [a] through [c] cannot give :

1. The average of a singleton hypothesis is itself, so .
2. The average of constant, real-valued hypotheses is a constant, real-valued number, so .
3. In the linear regression model, a hypothesis has the form of a linear combination with real-valued coefficients. The average of these linear combinations is another linear combination with real-valued coefficients, so .

However, in the logistic regression model, a hypothesis has the form of a logistic (sigmoidal) function with real-valued coefficients. The average of these logistic functions is not necessarily another logistic function, so could be true for [d].

(Consider the simple hypothesis set containing only

The average is

which cannot be reduced to a logistic function.)

**Overfitting**

1. Which of the following statements is false?

**Answer: [d] We can always determine if there is overfitting by comparing the values of .**

Choices [a] through [c] are all true:

1. Overfitting occurs when a hypothesis with the lowest is selected, even though it produces a higher . Therefore, there must be at least two hypotheses with different values.
2. As an extension of [a], the two hypotheses must have different values for overfitting to happen.
3. For [a] and [b] to be true, the two hypotheses must have different values. As an example, consider two hypotheses with in-sample and out-of-sample error pairs and , with hypothesis 1 erroneously selected for its lower value. If the two hypotheses have the same values, then for hypothesis 2 must be 0.4, which would then mean no overfitting occurred.

Leveraging the logic in [c], it is easy to see that [d] must be false.

In the example in [c], for hypothesis 1 is greater than for hypothesis 2.

Consider another example where and . The values for the two hypotheses are the same, but there is no longer overfitting. Therefore, the magnitudes of the values cannot be used to determine if there is overfitting, and [d] is false.

1. Which of the following statements is true?

**Answer: [d] Stochastic noise does not depend on the hypothesis set.**

Choices [a], [b], [c], and [e] are all false:

1. Deterministic and stochastic noise can be present simultaneously; the data set can have inherent fluctuations or measurement errors, and certain parts of the target function may be impossible to reproduce using the chosen model.
2. As described in [a] above, deterministic noise is caused by the inability of the hypothesis set to model a target function (e.g., hypothesis set of linear combinations used to fit a quadratic target function).
3. Same explanation as [b] above.
4. Stochastic noise is dependent on the target distribution, as a distribution with a lower variance will have less stochastic noise.

From [a] and [e], it is evident that stochastic noise is only affected by the input data and does not change with the hypothesis set used to model the data, so [d] is true.

**Regularization**

1. The regularized weight is a solution to:

where is a Tikhonov matrix. If , where is the linear regression solution, then what is ?

**Answer: [a]**

As the linear regression solution minimizes the squared error and satisfies the same constraint used to determine , it is equal to the regularized weight .

1. Soft-order constraints that regularize polynomial models can be

**Answer: [b] translated into augmented error**

Soft-order constraints prevent overfitting by encouraging the weights to be small but not necessarily zero (which is what a hard constraint is) and does not change the order of the polynomial. This means that

1. is false by definition,
2. is false since the VC dimension does not change despite the effectively smaller model with regularization, and
3. is false since is expected to increase for better generalization (lower ).

A soft-order constraint yields the optimization problem

For to be optimal,

By grouping into the undetermined variable , would satisfy

That is, locally minimizes , which is the augmented error where the second term represents a penalty term. Therefore, [b] is true.

**Regularized Linear Regression**

We are going to experiment with linear regression for classification on the processed U.S. Postal Service Zip Code data set from Homework 8. Download the data (extracted features of intensity and symmetry) for training and testing:

<http://www.amlbook.com/data/zip/features.train>

<http://www.amlbook.com/data/zip/features.test>

(The format of each row is: **digit, intensity, symmetry**.) We will train two types of binary classifiers; one-versus-one (one digit is class +1 and another digit is class , with the rest of the digits disregarded), and one-versus-all (one digit is class +1 and the rest of the digits are class ). When evaluating and of the resulting classifier, use binary classification error. Implement the regularized least-squares linear regression for classification that minimizes

where includes .

1. Set and dop not apply a feature transform (i.e., use ). Which among the following classifiers has the lowest ?

**Answer: [d] 8 versus all**

1. Now, apply a feature transform , and set . Which among the following classifiers has the lowest ?

**Answer: [b] 1 versus all**

1. If we compare using the transform versus not using it, and applying that to “0 versus all” through “9 versus all”, which of the following statements is correct for ?

**Answer: [e] The transform improves the out-of-sample performance of “5 versus all”, but by less than 5%.**

The transform improves the out-of-sample error from 0.079721 to 0.079223, a 0.6% decrease, for the “5 versus all” classifier.

1. Train the “1 versus 5” classifier with with and . Which of the following statements is correct?

**Answer: [a] Overfitting occurs (from to ).**

From to , decreases from 0.005125 to 0.004484, but increases from 0.025943 to 0.028302. This signifies overfitting since the hypothesis will be selected due for its lower in-sample error despite its worse out-of-sample performance.

(The program output and Python 3 source code is available on the following pages.)

The output used to answer Problems 7–10 is

[FE P7–9]

Linear regression with regularization:

0 vs. all:

X: E\_in=0.109313, E\_out=0.115097

Z: E\_in=0.102318, E\_out=0.106627

1 vs. all:

X: E\_in=0.015224, E\_out=0.022422

Z: E\_in=0.012344, **E\_out=0.021923**

2 vs. all:

X: E\_in=0.100261, E\_out=0.098655

Z: E\_in=0.100261, E\_out=0.098655

3 vs. all:

X: E\_in=0.090248, E\_out=0.082711

Z: E\_in=0.090248, E\_out=0.082711

4 vs. all:

X: E\_in=0.089425, E\_out=0.099651

Z: E\_in=0.089425, E\_out=0.099651

5 vs. all:

X: E\_in=0.076258, **E\_out=0.079721**

Z: E\_in=0.076258, **E\_out=0.079223**

6 vs. all:

X: E\_in=0.091071, E\_out=0.084704

Z: E\_in=0.091071, E\_out=0.084704

7 vs. all:

X: E\_in=0.088465, E\_out=0.073244

Z: E\_in=0.088465, E\_out=0.073244

8 vs. all:

X: **E\_in=0.074338**, E\_out=0.082711

Z: E\_in=0.074338, E\_out=0.082711

9 vs. all:

X: E\_in=0.088328, E\_out=0.088191

Z: E\_in=0.088328, E\_out=0.088191

[FE P10]

Linear regression with transform and regularization for 1 vs. 5 classifier:

lambda=0.01: **E\_in=0.004484**, **E\_out=0.028302**

lambda=1: **E\_in=0.005125**, **E\_out=0.025943**

The Python 3 source code is below and continues on the following pages:

import pathlib

import sys

import numpy as np

import requests

CWD = pathlib.Path(\_\_file\_\_).resolve()

DATA\_DIR = (CWD.parents[1] / "data").resolve()

def validate\_binary(w, x, y):

    return np.count\_nonzero(np.sign(x @ w) != y, axis=0) / x.shape[0]

def linear\_regression(

        N=None, f=None, vf=None, \*, x=None, y=None, transform=None, noise=None,

        regularization=None, N\_test=1\_000, x\_test=None, y\_test=None,

        x\_validate=None, y\_validate=None, rng=None, seed=None, hyp=False,

        \*\*kwargs):

    if rng is None:

        rng = np.random.default\_rng(seed)

    if x is None:

        x, y = generate\_data(N, f, bias=True, rng=rng)

    elif y is None:

        N = x.shape[0]

        y = f(x)

    else:

        N = x.shape[0]

    if transform:

        x = transform(x)

    if noise:

        i = rng.choice(N, round(noise[0] \* N), False)

        y[i] = noise[1](y[i])

    if regularization is None:

        w = np.linalg.pinv(x) @ y

    elif regularization == "weight\_decay":

        w = np.linalg.inv(

            x.T @ x + kwargs["wd\_lambda"] \* np.eye(x.shape[1], dtype=float)

        ) @ x.T @ y

    if vf is None:

        return w

    if x\_test is None or y\_test is None:

        if f is None:

            return (w, vf(w, x, y))[1 - hyp:]

        if x\_test is None:

            x\_test, y\_test = generate\_data(N\_test, f, bias=True, rng=rng)

        elif y\_test is None:

            N\_test = x\_test.shape[0]

            y\_test = f(x\_test)

    else:

        N\_test = x\_test.shape[0]

    if transform:

        x\_test = transform(x\_test)

    if noise:

        i = rng.choice(N\_test, round(noise[0] \* N\_test), False)

        y\_test[i] = noise[1](y\_test[i])

    if x\_validate is None or y\_validate is None:

        return (w, vf(w, x, y), vf(w, x\_test, y\_test))[1 - hyp:]

    N\_validate = len(y\_validate)

    if transform:

        x\_validate = transform(x\_validate)

    if noise:

        i = rng.choice(N\_validate, round(noise[0] \* N\_validate), False)

        y\_validate[i] = noise[1](y\_validate[i])

    return (w, (vf(w, x, y), vf(w, x\_validate, y\_validate)),

            vf(w, x\_test, y\_test))[1 - hyp:]

if \_\_name\_\_ == "\_\_main\_\_":

    # Regularized Linear Regression

    DATA\_DIR.mkdir(exist\_ok=True)

    data = {}

    for dataset in ["train", "test"]:

        file = f"features.{dataset}"

        if not (DATA\_DIR / file).exists():

            r = requests.get(f"http://www.amlbook.com/data/zip/{file}")

            with open(DATA\_DIR / file, "wb") as f:

                f.write(r.content)

        data[dataset] = np.loadtxt(DATA\_DIR / file)

    # Problems 7–10

    print("\n[FE P7–9]\nLinear regression with regularization:")

    transform = lambda x: np.hstack((x, x[:, 1:2] \* x[:, 2:], x[:, 1:2] \*\* 2,

                                     x[:, 2:] \*\* 2))

    for digit in range(10):

        x = np.hstack((np.ones((len(data["train"]), 1), dtype=float),

                    data["train"][:, 1:]))

        y = 2 \* (data["train"][:, 0] == digit) - 1

        x\_test = np.hstack((np.ones((len(data["test"]), 1), dtype=float),

                            data["test"][:, 1:]))

        y\_test = 2 \* (data["test"][:, 0] == digit) - 1

        print(f"  {digit} vs. all:")

        for t, l in zip((None, transform), ("X", "Z")):

            E\_in, E\_out = linear\_regression(

                vf=validate\_binary, x=x, y=y, transform=t,

                regularization="weight\_decay", wd\_lambda=1,

                x\_test=x\_test, y\_test=y\_test

            )

            print(f"    {l}: {E\_in=:.6f}, {E\_out=:.6f}")

    print("\n[FE P10]\nLinear regression with transform and "

          "regularization for 1 vs. 5 classifier:")

    subset = data["train"][np.isin(data["train"][:, 0], (1, 5))]

    x = np.hstack((np.ones((len(subset), 1), dtype=float), subset[:, 1:]))

    y = (subset[:, 0] == 1).astype(int) - (subset[:, 0] == 5)

    subset\_test = data["test"][np.isin(data["test"][:, 0], (1, 5))]

    x\_test = np.hstack((np.ones((len(subset\_test), 1), dtype=float),

                        subset\_test[:, 1:]))

    y\_test = (subset\_test[:, 0] == 1).astype(int) - (subset\_test[:, 0] == 5)

    for wd\_lambda in (0.01, 1):

        E\_in, E\_out = linear\_regression(

            vf=validate\_binary, x=x, y=y, transform=transform,

            regularization="weight\_decay", wd\_lambda=wd\_lambda,

            x\_test=x\_test, y\_test=y\_test

        )

        print(f"    lambda={wd\_lambda}: {E\_in=:.6f}, {E\_out=:.6f}")

**Support Vector Machines**

1. Consider the following training set generated from a target function , where :

Transform this training set into another two-dimensional space :

Using geometry (not quadratic programming), what values of (without ) and specify the separating plane that maximizes the margin in the space? The values of , , and are:

**Answer: [c]**

The data set is visualized in and space below:

A diagram of a number of numbers

Description automatically generated

In space, the separating plane is best placed extending into and out of the page between the blue squares and orange circles above at , suggesting that and since the value is irrelevant.

Then, we can solve for using a point that lies on the optimal plane, like ,

1. Consider the same training set of the previous problem, but instead of explicitly transforming the input space , apply the hard-margin SVM algorithm with the kernel

(which corresponds to a second-order polynomial transformation). Set up the expression for and solve for the optimal (numerically, using a quadratic programming package). The number of support vectors you get is in what range?

**Answer: [c] 4–5**

The output used to answer Problem 12 is

[FE P12]

Second-order polynomial support vector machine: N\_sv=5

The Python 3 source code for Problems 12–13 is below and continues on the next page:

import matplotlib as mpl

import matplotlib.pyplot as plt

import numpy as np

from sklearn import svm

mpl.rcParams.update(

    {

        "axes.labelsize": 14,

        "figure.autolayout": True,

        "figure.figsize": (4.875, 3.65625),

        "font.size": 12,

        "legend.columnspacing": 1,

        "legend.edgecolor": "1",

        "legend.framealpha": 0,

        "legend.fontsize": 12,

        "legend.handlelength": 1.25,

        "legend.labelspacing": 0.25,

        "xtick.labelsize": 12,

        "ytick.labelsize": 12,

        "text.usetex": True

    }

)

if \_\_name\_\_ == "\_\_main\_\_":

    # Problems 11–12

    x = np.array(((1, 0), (0, 1), (0, -1), (-1, 0), (0, 2), (0, -2),

              (-2, 0)), dtype=float)

    y = np.array((-1, -1, -1, 1, 1, 1, 1), dtype=int)

    z = np.hstack((x[:, 1:] \*\* 2 - 2 \* x[:, :1] - 1,

                   x[:, :1] \*\* 2 - 2 \* x[:, 1:] + 1))

    \_, ax = plt.subplots()

    ax.scatter(\*z[y == 1].T, marker="s", label="$+1$")

    ax.scatter(\*z[y == -1].T, marker="o", label="$-1$")

    ax.set\_aspect("equal", "box")

    ax.set\_xlabel("$z\_1$")

    ax.set\_xlim(-6, 6)

    ax.set\_ylabel("$z\_2$")

    ax.set\_ylim(-6, 6)

    ax.legend(title="Classification", loc="lower left")

    plt.show()

    clf = svm.SVC(C=np.finfo(float).max, kernel="poly", degree=2, gamma=1,

                  coef0=1)

    clf.fit(x, y)

    print("[FE P12]\nSecond-order polynomial support vector machine: "

        f"N\_sv={clf.n\_support\_.sum()}")

**Radial Basis Functions**

We experiment with the RBF model, both in regular form (Lloyd + pseudo-inverse) with centers:

(notice that there is a bias term), and in kernel form (using the RBF kernel in hard-margin SVM):

The input space is with uniform probability distribution, and the target is

which is slightly nonlinear in the space. In each run, generate 100 training points at random using this target, and apply both forms of RBF to these training points. Here are some guidelines:

* Repeat the experiment for as many runs as needed to get the answer to be stable (statistically away from flipping to the closest competing answer).
* In case a data set is not separable in the “ space” by the RBF kernel using hard-margin SVM, discard the run but keep track of how often this happens, if ever.
* When you use Lloyd’s algorithm, initialize the centers to random points in and iterate until there is no change from iteration to iteration. If a cluster becomes empty, discard the run and repeat.

1. For , how often do you get a data set that is not separable by the RBF kernel (using hard-margin SVM)? Hint: Run the hard-margin SVM, then check if the solution has .

**Answer: [a] of the time**

1. If we use for regular RBF and take , how often does the kernel form beat the regular form (excluding runs mentioned in Problem 13 and runs with empty clusters, if any) in terms of ?

**Answer:**

1. If we use for regular RBF and take , how often does the kernel form beat the regular form (excluding runs mentioned in Problem 13 and runs with empty clusters, if any) in terms of ?

**Answer:**

1. Now we focus on regular RBF only, with . If we go from clusters to clusters (only 9 and 12), which of the following 5 cases happens most often in your runs (excluding runs with empty clusters, if any)? Up or down means strictly so.

**Answer:**

1. For regular RBF with , if we go from to (only 1.5 and 2), which of the following 5 cases happens most often in your runs (excluding runs with empty clusters, if any)? Up or down means strictly so.

**Answer:**

1. What is the percentage of time that regular RBF achieves with and (excluding runs with empty clusters, if any)?

**Answer:**

**Bayesian Priors**

1. Let be the unknown probability of getting a heart attack for people in a certain population. Notice that is just a constant, not a function, for simplicity. We want to model using a hypothesis . Before we see any data, we assume that is uniform over (the prior). We pick one person from the population, and it turns out that he or she had a heart attack. Which of the following is true about the posterior probability that given this sample point?

**Answer:**

**Aggregation**

1. Given two learned hypotheses and , we construct the aggregate hypothesis given by for all . If we use the mean-squared error, which of the following statements is true?

**Answer:**