Benjamin Ye

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

November 6, 2023

**Homework 6**

|  |  |
| --- | --- |
| **Problem** | **Answer** |
| 1 | [b] |
| 2 | [a] |
| 3 | [d] |
| 4 | [e] |
| 5 | [d] |
| 6 | [b] |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |

**Overfitting and Deterministic Noise**

1. Deterministic noise depends on , as some models approximate better than others. Assume that and that is fixed. In general (but not necessarily in all cases), if we use instead of , how does deterministic noise behave?

**Answer: [b] In general, deterministic noise will increase.**

In general, there are fewer hypotheses in than because is a subset of . As such, we expect that will model less of the target function correctly in the input space . The deterministic noise (or the bias) is expected to increase since it is higher for smaller models.

**Regularization with Weight Decay**

In the following problems, use the data provided in the files

<http://work.caltech.edu/data/in.dta>

<http://work.caltech.edu/data/out.dta>

as a training and test set, respectively. Each line of the files corresponds to a two-dimensional input , so that , followed by the corresponding label from . We are going to apply linear regression with a non-linear transformation for classification. The nonlinear transformation is given by

Recall that the classification error is defined as the fraction of misclassified points.

1. Run linear regression on the training set after performing the non-linear transformation. What values are closest (in Euclidean distance) to the in-sample and out-of-sample classification errors, respectively?

**Answer: [a] 0.03, 0.08**

1. Now add weight decay to linear regression, that is, add the term to the squared in-sample error, using . What are the closest values to the in-sample and out-of-sample classification errors, respectively, for ? Recall that the solution for linear regression with weight decay was derived in class.

**Answer: [d] 0.03, 0.08**

1. Now, use . What are the closest values to the new in-sample and out-of-sample classification errors, respectively?

**Answer: [e] 0.4, 0.4**

1. What value of , among the following choices, achieves the smallest out-of-sample classification error?

**Answer: [d]**

1. What value is closest to the minimum out-of-sample classification error achieved by varying (limiting to integer values)?

**Answer: [b] 0.06**

See the next page for the linear regression results and a brief discussion.

The output used to answer problems 2–6 is

[HW6 P2–6]

Linear regression (without regularization) statistics:

**E\_in=0.029**, **E\_out=0.084**

Linear regression (with weight decay regularization using lambda=10^k) statistics:

k=-5: E\_in=0.029, E\_out=0.084

k=-4: E\_in=0.029, E\_out=0.084

k=-3: **E\_in=0.029**, E\_out=**0.080**

k=-2: E\_in=0.029, E\_out=0.084

**k=-1**: E\_in=0.029, **E\_out=0.056**

k=0: E\_in=0.000, E\_out=0.092

k=1: E\_in=0.057, E\_out=0.124

k=2: E\_in=0.200, E\_out=0.228

k=3: **E\_in=0.371**, **E\_out=0.436**

k=4: E\_in=0.429, E\_out=0.452

k=5: E\_in=0.429, E\_out=0.456

k=6: E\_in=0.429, E\_out=0.456

For problem 6, the out-of-sample classification error stops changing for and .

For , we get overfitting as , while for , we get underfitting as . The expected for the weight decay regularization should have a minimum in the range , which is exactly what we see for .

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

import numpy as np

import requests

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, rng=None,

        seed=None, hyp=False, \*\*kwargs):

    if rng is None:

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

    if y is None:

        if x is None:

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

        else:

            N = x.shape[0]

            y = f(x)

    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 y\_test is None:

        if x\_test is None:

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

        else:

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

            y\_test = f(x\_test)

    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])

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

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

    (CWD / "data").mkdir(exist\_ok=True)

    data = {"train": "in.dta", "test": "out.dta"}

    for dataset, file in data.items():

        if not (CWD / "data" / file).exists():

            r = requests.get(f"http://work.caltech.edu/data/{file}")

            with open(CWD / "data" / file, "wb") as f:

                f.write(r.content)

        data[dataset] = np.loadtxt(CWD / "data" / file)

    transform = lambda x: np.hstack((

        np.ones((len(x), 1), dtype=float),

        x,

        x[:, :1] \*\* 2,

        x[:, 1:] \*\* 2,

        x[:, :1] \* x[:, 1:],

        np.abs(x[:, :1] - x[:, 1:]),

        np.abs(x[:, :1] + x[:, 1:])

    ))

    print("\n[HW6 P2–6]\nLinear regression (without regularization) "

        "statistics:")

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

        vf=validate\_binary, x=data["train"][:, :-1], y=data["train"][:, -1],

        transform=transform, x\_test=data["test"][:, :-1],

y\_test=data["test"][:, -1]

    )

    print(f"  {E\_in=:.3f}, {E\_out=:.3f}")

    print("Linear regression (with weight decay regularization using "

          "lambda=10^k) statistics:")

    for k in (ks := np.arange(-5, 7)):

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

            vf=validate\_binary, x=data["train"][:, :-1], y=data["train"][:, -1],

            transform=transform, regularization="weight\_decay",

wd\_lambda=10.0 \*\* k, x\_test=data["test"][:, :-1],

y\_test=data["test"][:, -1]

        )

        print(f"  {k=}: {E\_in=:.3f}, {E\_out=:.3f}")

**Regularization for Polynomials**

Polynomial models can be viewed as linear models in a space , under a nonlinear transform . Here, transforms the scalar into a vector of Legendre polynomials, . Our hypothesis set will be expressed as a linear combination of these polynomials,

where .

1. Consider the following hypothesis set defined by the constraint:

Which of the following statements is correct?

**Answer:**

**Neural Networks**

1. A fully connected neural network has ; , , . If only products of the form , , and count as operations (even for ), without counting anything else, which of the following is the closest to the total number of operations in a single iteration of backpropagation (using SGD on one data point)?

**Answer:**

Let us call every “node” in a neural network a unit, whether that unit is an input variable or a neuron in one of the layers. Consider a neural network that has 10 input units (the constant is counted here as a unit), one output unit, and 36 hidden layers (each is also counted as a unit). The hidden units can be arranged in any number of layers , and each layer is fully connected to the layer above it.

1. What is the minimum possible number of weights that such a network can have?

**Answer:**

1. What is the maximum possible number of weights that such a network can have?

**Answer:**