A1 - MAD

Julian Wulff

November 2019

Exercise 1

a)

First we find $\frac{\partial f}{\partial x}$

$$\frac{\partial}{\partial x}\left(x^4y^3 + x^5 - e^y\right) = \frac{\partial}{\partial x}\left(x^4y^3\right) + \frac{\partial}{\partial x}\left(x^5\right) + \frac{\partial}{\partial x}\left(-e^y\right) \tag{1}$$

1) Here the sum rule is applied

$$= \frac{\partial}{\partial x}\left(x^4y^3\right) + \frac{\partial}{\partial x}\left(x^5\right) + \frac{\partial}{\partial x} \tag{2}$$

2) Here we applied the constant rule

$$= y^3\frac{\partial}{\partial x}\left(x^4\right) + \frac{\partial}{\partial x}\left(x^5\right) \tag{3}$$

3) Here the constantmultiple rule is applied

$$= 4y^3x^3 + \frac{\partial}{\partial x}\left(x^5\right) \tag{4}$$

$$= 4y^3x^3 + 5x^4 \tag{5}$$

4) and 5) here the power rule is applied.

We now find $\frac{\partial f}{\partial y}$ by

$$\frac{\partial}{\partial y}\left(x^4y^3 + x^5 - e^y\right) = \frac{\partial}{\partial y}\left(x^4y^3\right) + \frac{\partial}{\partial y}\left(x^5\right) + \frac{\partial}{\partial y}\left(-e^y\right) \tag{6}$$

6) Here we apply the sum rule.

$$= \frac{\partial}{\partial y}\left(x^4y^3\right) + \frac{\partial}{\partial y}\left(-e^y\right) \tag{7}$$

7) here the constant rule is applied.

$$= x^4\frac{\partial}{\partial y}\left(y^3\right) + \frac{\partial}{\partial y}\left(-e^y\right) \tag{8}$$

8) Here the constantmultiple rule is applied.

$$= 3x^4y^2 + \frac{\partial}{\partial y}\left(-e^y\right) \tag{9}$$

9) here the power rule is applied

$$= 3x^4y^2 - \frac{\partial}{\partial y}\left(e^y\right) \tag{10}$$

$$= 3x^4y^2 - e^y \tag{11}$$

10) and 11) here the constantmultiple and exponential rule is applied.

b)

First we find $\frac{\partial f}{\partial x}$

$$\frac{\partial}{\partial x}\left(\frac{1}{\sqrt{x^3+xy+y^2}}\right) = \frac{\partial}{\partial x}\left(\left(x^3+xy+y^2\right)^{-\frac{1}{2}}\right) \tag{12}$$

By applying the exponent rule.
We then apply the chain rule and get

$$\frac{\partial f(u)}{\partial x} = \frac{\partial f}{\partial u} \cdot \frac{\partial u}{\partial x} \tag{13}$$

where

$$f = u^{-\frac{1}{2}} \qquad u = (x^3 + xy + y^2) \tag{14}$$

$$\tag{15}$$

We then solve

$$\frac{\partial}{\partial u}\left(u^{-\frac{1}{2}}\right) = -\frac{1}{2u^{\frac{3}{2}}} \tag{16}$$

$$\tag{17}$$

by applying the power rule and simplifying
We now solve

$$\frac{\partial}{\partial x}\left(x^3+xy+y^2\right) = 3x^2 + y \tag{18}$$

using the sum rule, the power rule and the constant rule. Now we have

$$\left(-\frac{1}{2u^{\frac{3}{2}}}\right)\left(3x^3+y\right) \tag{19}$$

and substitute back $u = (x^3 + xy + y^2)$ and get

$$\left(-\frac{1}{2(x^3+xy+y^2)^{\frac{3}{2}}}\right)\left(3x^3+y\right) = -\frac{3x^3+y}{2(x^3+xy+y^2)^{\frac{3}{2}}} \tag{20}$$

Now we find $\frac{\partial f}{\partial y}$

$$\frac{\partial}{\partial x}\left(\frac{1}{\sqrt{x^3+xy+y^2}}\right)=\frac{\partial}{\partial x}\left(\left(x^3+xy+y^2\right)^{-\frac{1}{2}}\right) \tag{21}$$

by applying the exponent rule.
We then apply the chain rule and get

$$\frac{\partial f(u)}{\partial x}=\frac{\partial f}{\partial u}\cdot\frac{\partial u}{\partial x} \tag{22}$$

where

$$f=u^{-\frac{1}{2}} \qquad u=(x^3+xy+y^2) \tag{23}$$
$$\tag{24}$$

We then solve

$$\frac{\partial}{\partial u}\left(u^{-\frac{1}{2}}\right)=-\frac{1}{2u^{\frac{3}{2}}} \tag{25}$$
$$\tag{26}$$

by applying the power rule and simplifying
We now solve

$$\frac{\partial}{\partial x}\left(x^3+xy+y^2\right)=2y+x \tag{27}$$

using the sum rule, the power rule and the constant rule. Now we have

$$\left(-\frac{1}{2u^{\frac{3}{2}}}\right)(2y+x) \tag{28}$$

and substitute back $u=(x^3+xy+y^2)$ and get

$$\left(-\frac{1}{2(x^3+xy+y^2)^{\frac{3}{2}}}\right)(2y+x)=-\frac{2y+x}{2(x^3+xy+y^2)^{\frac{3}{2}}} \tag{29}$$

c)

First we find $\frac{\partial f}{\partial x}$

$$\frac{\partial f}{\partial x}\left(\frac{x^3+y^2}{x+y}\right) \tag{30}$$

$$= \frac{\frac{\partial}{\partial x}(x^3+y^2)(x+y) - (x^3+y^2)\frac{\partial}{\partial x}(x+y)}{(x+y)^2} \tag{31}$$

$$= \frac{\left(\frac{\partial}{\partial x}(x^3) + \frac{\partial}{\partial x}(y^2)\right)(x+y) - (x^3+y^2)\left(\frac{\partial}{\partial x}(x) + \frac{\partial}{\partial x}(y)\right)}{(x+y)^2} \tag{32}$$

$$= \frac{\frac{\partial}{\partial x}(x^3)(x+y) - (x^3+y^2)\frac{\partial}{\partial x}(x)}{(x+y)^2} \tag{33}$$

$$= \frac{3x^2(x+y) - (x^3+y^2)\frac{\partial}{\partial x}(x)}{(x+y)^2} \tag{34}$$

$$= \frac{3x^2(x+y) - x^3 - y^2}{(x+y)^2} \tag{35}$$

In equation 31 the quotient rule is applied
In equation 32 the sum rule is applied
In equation 33 the constant rule is applied
In equation 34 the power rule is applied
In equation 35 the identity rule is applied

Now we find $\frac{\partial f}{\partial y}$

$$\frac{\partial f}{\partial y}\left(\frac{x^3+y^2}{x+y}\right) \tag{36}$$

$$= \frac{\frac{\partial}{\partial y}(x^3+y^2)(x+y) - (x^3+y^2)\frac{\partial}{\partial x}(x+y)}{(x+y)^2} \tag{37}$$

$$= \frac{\left(\frac{\partial}{\partial y}(x^3) + \frac{\partial}{\partial y}(y^2)\right)(x+y) - (x^3+y^2)\left(\frac{\partial}{\partial y}(x) + \frac{\partial}{\partial y}(y)\right)}{(x+y)^2} \tag{38}$$

$$= \frac{\frac{\partial}{\partial y}(y^2)(x+y) - (x^3+y^2)\frac{\partial}{\partial y}(y)}{(x+y)^2} \tag{39}$$

$$= \frac{2y(x+y) - (x^3+y^2)\frac{\partial}{\partial x}(y)}{(x+y)^2} \tag{40}$$

$$= \frac{2y(x+y) - x^3 - y^2}{(x+y)^2} \tag{41}$$

In equation 37 the quotient rule is applied
In equation 38 the sum rule is applied
In equation 39 the constant rule is applied
In equation 40 the power rule is applied
In equation 41 the identity rule is applied

Exercise 2

a)

In order to compute the gradient ∇f of $f(\bar{x}) = \bar{x}^T\bar{x}+c$ we will find the derivative of f with respect to $\bar{x}$.
First we apply the constant rule and get $\bar{x}^T\bar{x}$. Then we observer that computing $\bar{x}^T\bar{x}$ is equal to $x_1x_1+x_2x_2+\cdots+x_nx_n$. Rewriting this expression in summation form we get:

$$\sum_{i=1}^{n} x_ix_i = \sum_{i=1}^{n} x_i^2$$

. Differentiating this we get:

$$\sum_{i=1}^{n} 2x_i$$

which gives us the gradient

$$2\bar{x}$$

b)

In order to compute the gradient ∇f of $f(\bar{x}) = \bar{x}^T\bar{b}$ we will find the derivative of f with respect to $\bar{x}$.
First we observer that computing $\bar{x}^T\bar{b}$ is equal to $x_1b_1 + x_2b_2 + \cdots + x_nb_n$. Rewriting this expression in summation form we get:

$$\sum_{i=1}^{n} x_ib_i$$

. Differentiating this we get:

$$\sum_{i=1}^{n} b_i$$

which gives us the gradient

$$\bar{b}$$

c)

In calculating the gradient ∇f of $f(\bar{x}) = \bar{x}^T A\bar{x} + \bar{b}^T\bar{x} + c$ i was able to derive the $\bar{b}^T\bar{x} + c$ part with the same methods as in a) and b) but i was not able to derive the last part $\bar{x}^T A\bar{x}$. Therefore i just some identities in order to calculate the gradient

$$\nabla f = A\bar{x} + A^T\bar{x} + b$$

Exercise 3

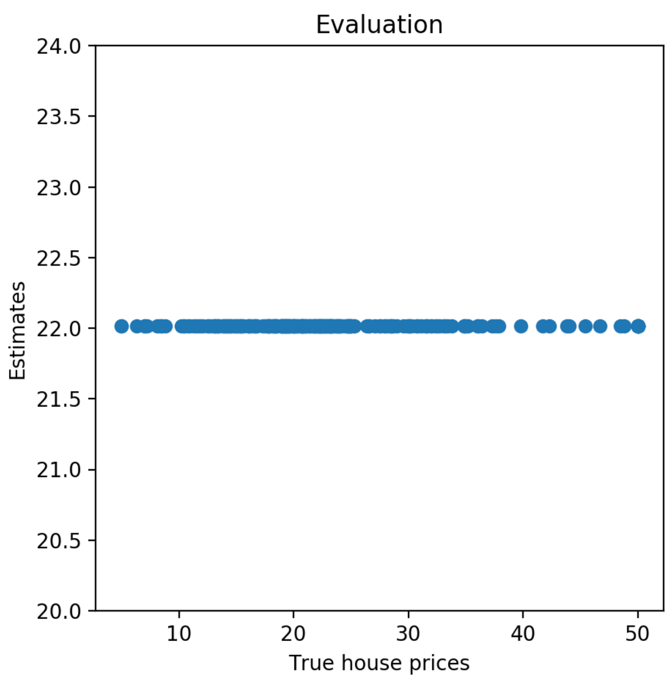
a)

The mean of the house prices in the training set is: 22.016601

b)

The RMSE between the true house prices and the estimates obtained via the simple 'mean' model for the test set is: 9.672478

c)



Exercise 4

a)

See the files `housing_2.py` and `linreg.py`

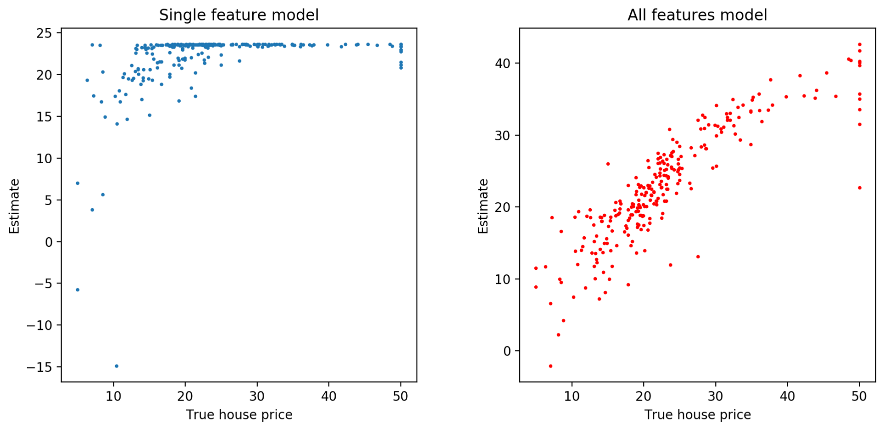


Figure 1: Figures containing scatter plot for true house prices vs estimated house prices. One figure for the single feature model and one for the all features model.

b)

The two weights $\hat{w}_0$ and $\hat{w}_1$ obtained from fitting the linear regression model on the training set only containing the feature (CRIM) are $\hat{w}_0 = 23.635062$ and $\hat{w}_1 = -0.432793$. These values suggest that if the per capita crime rate by town is 0 then the median value of owner-occupied homes in \$1000's is equal to 23.635 and that this is affected by the crime rate, such that median value decreases as the crime rate increses.

c)

From fitting the linear regression model on the training set containing all the features we obtain the values (for easier reading i have round these value to 3 decimals. The full values can be found when running the code in `housing_2.py`):

$$\hat{w}_0 = 31.389 \quad \hat{w}_1 = -0.06 \quad \hat{w}_2 = 0.029 \quad \hat{w}_3 = -0.029 \quad \hat{w}_4 = 2.293$$
$$\hat{w}_5 = -17.326 \quad \hat{w}_6 = 3.994 \quad \hat{w}_7 = 0.003 \quad \hat{w}_8 = -1.287 \quad \hat{w}_9 = 0.355$$
$$\hat{w}_{10} = -0.016 \quad \hat{w}_{11} = -0.815 \quad \hat{w}_{12} = 0.012 \quad \hat{w}_{13} = -0.465$$

d)

RMSE of the first feature only model is: 8.954860
RMSE of the all feature model is: 4.688334

1 Code

housing_1.py

```
import numpy as np
import matplotlib.pyplot as plt

np.set_printoptions(precision=3)
np.set_printoptions(suppress=True)

# load data
train_data = np.loadtxt("boston_train.csv", delimiter=",")
test_data = np.loadtxt("boston_test.csv", delimiter=",")

X_train, t_train = train_data[:,:-1], train_data[:,-1]
X_test, t_test = test_data[:,:-1], test_data[:,-1]

# make sure that we have N-dimensional Numpy arrays (ndarray)
t_train = t_train.reshape((len(t_train), 1))
t_test = t_test.reshape((len(t_test), 1))
print("Number of training instances: %i" % X_train.shape[0])
print("Number of test instances: %i" % X_test.shape[0])
print("Number of features: %i" % X_train.shape[1])

# (a) compute mean of prices on training set
t_mean = np.mean(t_train)
print("The mean of the house prices in the training set is: %f" %
    t_mean)
tp = np.full((len(t_train), 1), t_mean)

# (b) RMSE function
def rmse(t, tp):
    return np.sqrt(((t - tp) ** 2).mean())

print("The RMSE between the true house prices and the estimates
    obtained via the simple \'mean\' model for the test set is: %f"
     % (rmse(t_test, tp)))

# (c) visualization of results
plt.figure(figsize=(5,5))
plt.scatter(t_test, tp)
plt.xlabel("True house prices")
plt.ylabel("Estimates")
plt.ylim([20,24])
plt.title("Evaluation")
plt.show()
```

housing_1.py

housing_2.py

```
import numpy as np
import pandas
import linreg
```

```python
import matplotlib.pyplot as plt

np.set_printoptions(precision=3)
np.set_printoptions(suppress=True)

# load data
train_data = np.loadtxt("boston_train.csv", delimiter=",")
test_data = np.loadtxt("boston_test.csv", delimiter=",")
X_train, t_train = train_data[:,:-1], train_data[:,-1]
X_test, t_test = test_data[:,:-1], test_data[:,-1]
# make sure that we have N-dimensional Numpy arrays (ndarray)
t_train = t_train.reshape((len(t_train), 1))
t_test = t_test.reshape((len(t_test), 1))
print("Number of training instances: %i" % X_train.shape[0])
print("Number of test instances: %i" % X_test.shape[0])
print("Number of features: %i" % X_train.shape[1])

# (b) fit linear regression using only the first feature
model_single = linreg.LinearRegression()
model_single.fit(X_train[:,0], t_train)
print("Single feature model weights w0 = %f and w1 = %f " % (
    model_single.w[0], model_single.w[1]))

# (c) fit linear regression model using all features
model_all = linreg.LinearRegression()
model_all.fit(X_train, t_train)
print("Weights for all features model:")
print(model_all.w)

# (d) evaluation of results
def rmse(t, tp):
    return np.sqrt(((t - tp) ** 2).mean())

pred_single = model_single.predict(X_test[:,0])
rmse_single = rmse(t_test, pred_single)
print("RMSE of first feature only model: %f" % rmse_single)

plt.figure(figsize=(5,5))
plt.scatter(t_test, pred_single, s=3)
plt.xlabel("True house price")
plt.ylabel("Estimate")
plt.title("Single feature model")
plt.show()

pred_all = model_all.predict(X_test)
rmse_all = rmse(t_test, pred_all)
print("RMSE of all feature model: %f" % rmse_all)

plt.figure(figsize=(5,5))
plt.scatter(t_test, pred_all, s=3, color='red')
plt.xlabel("True house price")
plt.ylabel("Estimate")
plt.title("All features model")
plt.show()
```

housing_2.py

linreg.py

```
import numpy

# NOTE: This template makes use of Python classes. If
# you are not yet familiar with this concept, you can
# find a short introduction here:
# http://introtopython.org/classes.html

class LinearRegression():
    """
    Linear regression implementation.
    """

    def __init__(self):

        pass

    def fit(self, X, t):
        """
        Fits the linear regression model.

        Parameters
        ----------
        X : Array of shape [n_samples, n_features]
        t : Array of shape [n_samples, 1]
        """

        # TODO: YOUR CODE HERE
        X = numpy.array(X).reshape((len(X), -1))
        t = numpy.array(t).reshape((len(t), 1))

        ones = numpy.ones((X.shape[0], 1))
        X = numpy.concatenate((ones, X), axis=1)

        left = numpy.dot(X.T, X)
        right = numpy.dot(X.T, t)
        self.w = numpy.linalg.solve(left, right)

    def predict(self, X):
        """
        Computes predictions for a new set of points.

        Parameters
        ----------
        X : Array of shape [n_samples, n_features]

        Returns
        -------
        predictions : Array of shape [n_samples, 1]
        """

        # TODO: YOUR CODE HERE
        X = numpy.array(X).reshape((len(X), -1))

        # prepend a column of ones
        ones = numpy.ones((X.shape[0], 1))
```

```
        X = numpy.concatenate((ones, X), axis=1)

        # compute predictions
        predictions = numpy.dot(X, self.w)

        return predictions
```

linreg.py