MAD 2019, Assignment 2

Kasper Rasmussen - wdq486

December 3, 2019

1 Weighted Average Loss

1.a)

$$L = \frac{1}{N}\sum_{n=1}^{N}\alpha_n(\mathbf{w}^T\mathbf{x}_n - t_n)^2 = \frac{1}{N}\sum_{n=1}^{N}\alpha_n(\mathbf{w}^T\mathbf{x}_n - t_n)(\mathbf{w}^T\mathbf{x}_n - t_n)$$

Let $\mathbf{A}$ be defined as in the hint. Then $(\mathbf{Xw} - \mathbf{t})^T\mathbf{A} = [\alpha_1(\mathbf{w}^T\mathbf{x}_1 - t_1), ..., \alpha_n(\mathbf{w}^T\mathbf{x}_n - t_n)]$ and $(\mathbf{Xw} - \mathbf{t})^T\mathbf{A}(\mathbf{Xw} - \mathbf{t}) = \alpha_1(\mathbf{w}^T\mathbf{x}_1 - t_1)(\mathbf{w}^T\mathbf{x}_1 - t_1) + ... + \alpha_n(\mathbf{w}^T\mathbf{x}_n - t_n)(\mathbf{w}^T\mathbf{x}_n - t_n)$. Therefore

$$\frac{1}{N}\sum_{n=1}^{N}\alpha_n(\mathbf{w}^T\mathbf{x}_n - t_n)^2 = \frac{1}{N}(\mathbf{Xw} - \mathbf{t})^T\mathbf{A}(\mathbf{Xw} - \mathbf{t})$$

Now we manipulate this to make it easier to find the gradient.

$$\frac{1}{N}(\mathbf{Xw} - \mathbf{t})^T\mathbf{A}(\mathbf{Xw} - \mathbf{t}) = \frac{1}{N}((\mathbf{Xw})^T - \mathbf{t}^T)\mathbf{A}(\mathbf{Xw} - \mathbf{t}) =$$

$$\frac{1}{N}(\mathbf{w}^T\mathbf{X}^T - \mathbf{t}^T)\mathbf{A}(\mathbf{Xw} - \mathbf{t}) = \frac{1}{N}(\mathbf{w}^T\mathbf{X}^T - \mathbf{t}^T)(\mathbf{AXw} - \mathbf{At}) =$$

$$\frac{1}{N}(\mathbf{w}^T\mathbf{X}^T\mathbf{AXw} - \mathbf{w}^T\mathbf{X}^T\mathbf{At} - \mathbf{t}^T\mathbf{AXw} + \mathbf{t}^T\mathbf{At})$$

With constant and sum rule

$$\nabla L = \frac{1}{N}(\frac{\partial}{\partial \mathbf{w}}\mathbf{w}^T\mathbf{X}^T\mathbf{AXw} - \frac{\partial}{\partial \mathbf{w}}\mathbf{w}^T\mathbf{X}^T\mathbf{At} - \frac{\partial}{\partial \mathbf{w}}\mathbf{t}^T\mathbf{AXw} + \frac{\partial}{\partial \mathbf{w}}\mathbf{t}^T\mathbf{At})$$

$\mathbf{X}^T\mathbf{AX}$ is symmetric (A is symmetric), so for the first term we use rule 4 in Table 1.4, second term rule 1, third term uses rule 2 and the fourth term does not vary with $\mathbf{w}$, so it is $\mathbf{0}$

$$\nabla L = \frac{1}{N}(2(\mathbf{X}^T\mathbf{AX})\mathbf{w} - \mathbf{X}^T\mathbf{At} - (\mathbf{t}^T\mathbf{AX})^T + \mathbf{0})$$

Since $\mathbf{A}$ is symmetric $(\mathbf{t}^T\mathbf{AX})^T = \mathbf{X}^T\mathbf{At}$, so

$$\nabla L = \frac{2}{N}\mathbf{X^T}\mathbf{AXw} - \frac{1}{N}\mathbf{X}^T\mathbf{At} - \frac{1}{N}\mathbf{X}^T\mathbf{At} = \frac{2}{N}\mathbf{X}^T\mathbf{AXw} - \frac{2}{N}\mathbf{X}^T\mathbf{At}$$

Now we solve $\nabla L = \mathbf{0}$ for $\mathbf{w}$

$$\frac{2}{N}\mathbf{X}^T\mathbf{AXw} - \frac{2}{N}\mathbf{X}^T\mathbf{At} = \mathbf{0}$$

$$\frac{2}{N}\mathbf{X}^T\mathbf{AXw} = \frac{2}{N}\mathbf{X}^T\mathbf{At}$$

$$\mathbf{X}^T\mathbf{AXw} = \mathbf{X}^T\mathbf{At}$$

Then, if $\mathbf{X}^T\mathbf{AX}$ is invertible,

$$\mathbf{w} = (\mathbf{X}^T\mathbf{AX})^{-1}\mathbf{X}^T\mathbf{At}$$

1.b)

Assuming the derivation above is correct, we can use the equation $\mathbf{X}^T\mathbf{AXw} = \mathbf{X}^T\mathbf{At}$, for making weighted linear regression. We modify `fit` method in `LinearRegression` class to have an additional parameter `alpha` corresponding to α. The weights w are then found by solving the equation. Line 5-6 in Listing 1 sets up the numpy matrix corresponding to $\mathbf{A}$ in the previous exercise and lines 14-16 solves the system for w.

Listing 1: linweighreg.py fit method

```
def fit(self, X, t, alpha):
    X = np.array(X).reshape((len(X), -1))
    t = np.array(t).reshape((len(t), 1))
    # Make array A with diagonal = alpha vector
    A = np.zeros((len(t), len(t)))
    np.fill_diagonal(A, alpha)

    # prepend a column of ones
    ones = np.ones((X.shape[0], 1))
    X = np.concatenate((ones, X), axis=1)

    # compute weights (solve system)
    # X^TAXw = X^TAt
    lefthand_side = np.dot(np.dot(X.T, A), X)
    right_handside = np.dot(np.dot(X.T, A), t)
    self.w = np.linalg.solve(lefthand_side,right_handside)
```

Now this `LinearWeightedRegression` class is used to fit a model on all the features of in the training set `boston_train.csv` with $\alpha_n = t_n^2$ This is done using the initial part of `housing_2.py` from the previous assignment and extending it as shown in Listing 2. Line 6 in the Listing 2 shows the construction of the `alpha` corresponding to α and the line 9-10 fits the model. The RMSE error on the test data is calculated and a scatter plot is drawn where a line 22 is included `plt.plot(t_test, t_test, 'C1')` to make a line showing what would have been the perfect prediction.

Listing 2: housing_3.py

```
[...]

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

alpha = t_train**2

# (c) fit linear regression model using all features
model = linweighreg.LinearWeightedRegression()
model.fit(X_train, t_train, alpha)

print('Weights:')
print(model.w)

# (d) evaluation of results
prediction = model.predict(X_test)
rmse_testset = rmse(prediction, t_test)
print('\nRMSE on test set: ',rmse_testset)

# Scatter plots
plt.scatter(t_test, prediction)
plt.plot(t_test, t_test, 'C1')
plt.xlabel('True price')
plt.ylabel('Predicted price')
plt.title('Boston House Prices with weighted regression')
plt.show()

[...]
```

The result of running `housing_3.py` is shown in Figure 1 and the scatter plot is shown in Figure 2.

- *What do you expect to happen?* Since the vector t of target values in this example is the vector of house prices, and $\alpha_n = t_n^2$, I expect errors on houses with higher prices would be penalized more and thus the model would model prices of expensive houses *better* than cheaper houses.

```
> python3 housing_3.py
Number of training instances: 253
Number of test instances: 253
Number of features: 13
Weights:
[[ 1.75083507e+01]
 [ 1.40916656e-01]
 [ 2.88253875e-02]
 [-2.53140059e-02]
 [ 4.16450373e+00]
 [-1.63031856e+01]
 [ 5.41621155e+00]
 [ 2.97581647e-02]
 [-1.40915587e+00]
 [ 5.43436040e-01]
 [-1.54552624e-02]
 [-5.87852626e-01]
 [ 1.72833594e-02]
 [-7.77887298e-01]]

RMSE on test set:  6.21287919995914

Root mean weighted square error on test set:  144.86182104542675

RMWSE for prices > 40:  189.77343677279686
RMSE for prices > 40:  7.041359480805906

RMWSE for price < 30:  148.60104968149648
RMSE for prices < 30:  6.4584899968109095
```

Figure 1: Output from running `housing_3.py`

Figure 2: Scatter plot for true and predicted prices on test set with weighted regression

Figure 3: Scatter plot for true and predicted prices on test set with non-weighted regression

- *What do you observe? Do the additional weights have an influence on the outcome?* We can compare the scatter plot for weighted regression with a scatter plot from non-weighted regression generated using code from assignment 1. First we see that for housing with true price 0-30, some of the predicted prices are very far from the true price. For the most expensive houses with true price 40-50, the model seems to be a little better. Defining better in terms of RMSE, it can be shown by using the code shown in listing 3 and equivalent code for the unweighted regression. By doing this, it can be shown that the RMSE for prices above 40 on the test set is 7.04 for weighted and 12.27 for non-weighted (see housing_2b.py). For house prices below 30 we have 3.61 for non-weighted and 6.45 for weighted.

Listing 3: housing_3.py continued

```
[...]

t_test_high = []
prediction_high = []
alpha_high = []
for i in range(len(t_test)):
    if(t_test[i][0] > 40):
        t_test_high.append(t_test[i])
        prediction_high.append(prediction[i])
        alpha_high.append(alpha[i])
t_test_high = np.array(t_test_high)
prediction_high = np.array(prediction_high)
alpha_high = np.array(alpha_high)
rmwse_high = rmwse(prediction_high, t_test_high, alpha_high)
rmse_high = rmse(prediction_high, t_test_high)
print('\nRMWSE for prices > 40: ', rmwse_high)
print('RMSE for prices > 40: ', rmse_high)

[...]
```

2 Regularization and LOOCV

2.a)

We use the code from the notebooks on Absalon on regularized linear regression as seen in `Exercise2.ipynb`. Listing 4 shows a Python function `LOOCV` which performs LOOCV. It takes test data and a model. It loops through the rows in test data, generating `t_train` and `X_train` by deleting the row, and generated `t_validate` and `X_validate` by taking just the iterated row. It then fits the model to the training data and makes a prediction on the validation data. It finds the mean squared error and adds it to a list of losses. After doing this for every row it computes the average loss.

Listing 4: housing_3.py continued

```
def LOOCV(X, t, model):
    mseList = []
    for i in range(len(X)):
        t_train = np.copy(t)
        t_train = np.delete(t_train, i)
        X_train = np.copy(X)
        X_train = np.delete(X_train, i, 0)
        t_validate = t[i:i+1]
        X_validate = X[i:i+1]

        model.fit(X_train,t_train)
        prediction = model.predict(X_validate)
        mse = model.mse(prediction, t_validate)
        mseList.append(mse)
```

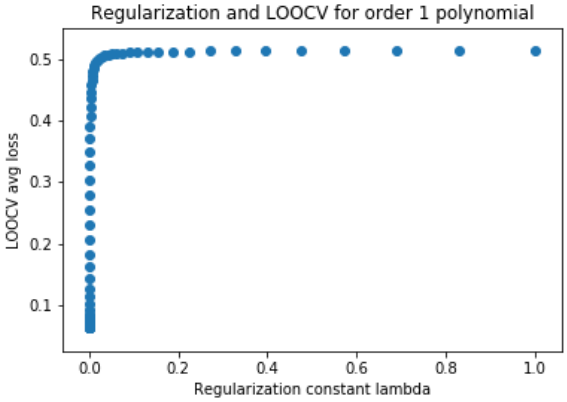

Figure 4: LOOCV loss as function of λ regularization constant for order 1

```
    avgMSE = np.array(mseList).mean()
    return avgMSE
```

We use this function to find the regularization constant which minimizes the LOOCV average loss on the Olympics data when fitting a first order polynomial. A plot of the LOOCV error as a function of λ is shown in Figure 4.

Listing 5: Excerpt 2 from Exercise2.ipynb

```
lambdas = np.logspace(-8, 0, 100, base=10)
# For each regularization constant do LOOCV
errors = []
for lam in lambdas:
    reglinreg = regularized_linreg.LinearRegression(lam)
    errors.append(LOOCV(X,t,reglinreg))
errors = np.array(errors)
```

We find the λ which minimises the LOOCV error with the code in Listing 6, which gives a $\lambda = 0.000000343046928631 49$ and LOOCV loss 0.06243120397875392968

Listing 6: Excerpt 3 from Exercise2.ipynb

```
# Finding regularization constant minimising the LOOCV avg loss
argmin = np.argmin(errors)
optimalLam = lambdas[argmin]
print("lam=%.20f and loss=%.20f" % (optimalLam, errors[argmin]))
```

Now we fit 2 models to the full data set, one using the best lambda and the other using no regularization (which is equivalent to $= 0$ since the regularization term is added to the MSE term in the loss function). Doing so, we get the following weights shown below. We see that the weights are almost identical and the plot in Figure 5 shows the same.

```
Optimal regularization weights:
w0 = 3.63776282e+0
w1 = -1.33110046e-02
No regularization weights:
w0 = 3.64164559e+01
w1 = -1.33308857e-02
```

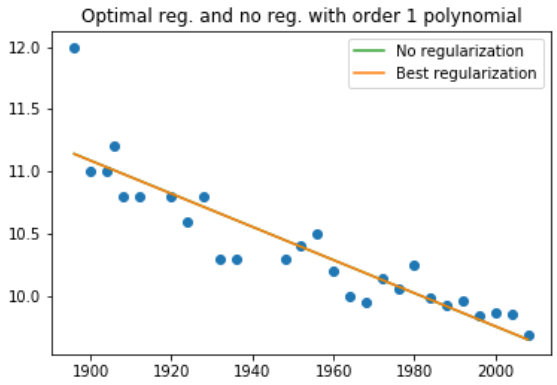

Figure 5: Order 1

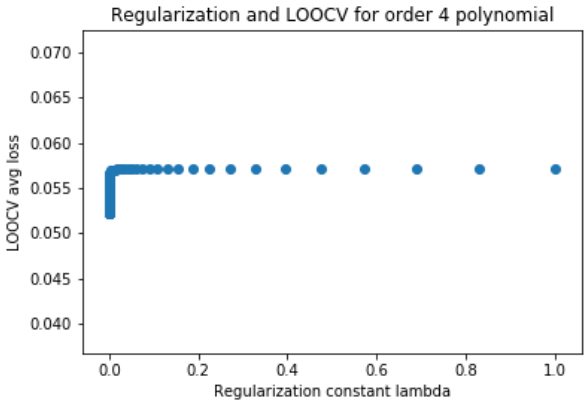

Figure 6: LOOCV loss as function of λ regularization constant for order 4

2.b)

Copying the code from lecture for making augmenting feature matrices with columns for polynomials, we make a new data matrix `X4` and follow the same procedure as above. This gives a plot of LOOCV as function λ as shown in Figure 6 and here the optimal $\lambda = 0.00002056512308348652$ which gives LOOOCV 0.05214287521969054873. Fitting on the whole data set with optimal regularization and none the weights are as follows, with plot shown in Figure 7.

```
Best regularization weights:
[[ 1.88290267e-02]
 [ 9.11460072e+00]
 [-1.38628671e-02]
 [ 7.03520268e-06]
 [-1.19059495e-09]]
No regularization weights:
[[ 1.00847612e+06]
 [-2.05490385e+03]
 [ 1.57007061e+00]
 [-5.33124319e-04]
 [ 6.78783436e-08]]
```

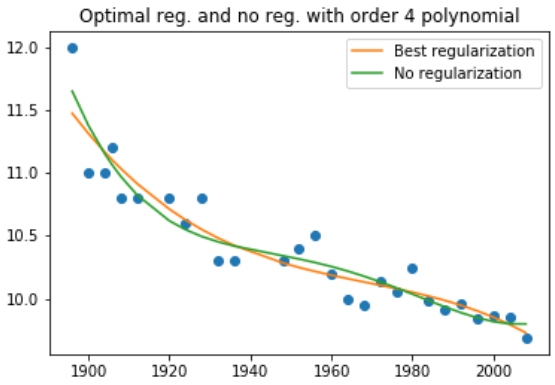

Figure 7: Order 4

3 Continuous random variables

a) $f(x) = F'(x)$ For $x \leq 0$, $f(x) = F'(x) = \frac{d}{dx}0 = 0$. For $x > 0$, $f(x) = F'(x) = \frac{d}{dx}(1 - exp(-\beta x^\alpha)) =$

$$\frac{d}{dx}1 - \frac{d}{dx}exp(-\beta x^\alpha)$$

By chain rule and power rule

$$-exp(-\beta x^\alpha)\frac{d}{dx}(-\beta x^\alpha) = \beta \exp(-\beta x^\alpha)\frac{d}{dx}(x^\alpha) = \beta\alpha x^{\alpha-1}exp(-bx^\alpha)$$

b) Since X is the lifespan of the chip, we are looking for $P(X > 4) = 1 - P(X \leq 4) = 1 - F(4) = 1 - (1 - exp(-\frac{1}{4} \cdot 4^2)) = 0.018316$. The probability that it will be working in interval [5,10] years is $P(X \in [5, 10]) = P(X \leq 10) - P(X < 5)$. Since $P(X = 5) = 0$, $P(X < 5) = P(P \leq 5)$, so we have $P(X \leq 10) - P(X \leq 5) = F(10) - F(5) = (1 - exp(-\frac{1}{4} \cdot 10^2)) - (1 - exp(-\frac{1}{4} \cdot 5^2)) = 0.0019305$

c) m is a median of X if $P(X \leq m) \geq \frac{1}{2}$ and $P(X \geq m) \leq \frac{1}{2}$

$P(X \leq m) \geq \frac{1}{2} \iff F(m) \geq \frac{1}{2} \iff 1 - exp(-\beta x^\alpha) \geq \frac{1}{2} \iff \frac{1}{2} \geq exp(-\beta x^\alpha) \iff \ln(\frac{1}{2}) \geq -\beta x^\alpha \iff \frac{\ln 1 - \ln 2}{-\beta} \leq x^\alpha \iff \frac{\ln 2}{\beta} \leq x^\alpha \iff (\frac{\ln 2}{\beta})^\alpha \leq x$

$P(X \geq m) \leq \frac{1}{2} \iff 1 - P(X \leq m) \leq \frac{1}{2} \iff 1 - F(m) \leq \frac{1}{2} \iff 1 - (1 - exp(-\beta x^\alpha) \leq \frac{1}{2} \iff exp(-\beta x^\alpha) \leq \frac{1}{2} \iff -\beta x^\alpha \leq \ln(\frac{1}{2}) \iff x^\alpha \geq \frac{\ln 1 - \ln 2}{-\beta} \iff x \geq (\frac{\ln 2}{\beta})^{\frac{1}{\alpha}}$

So $(\frac{\ln 2}{\beta})^{\frac{1}{\alpha}}$ satisfies the conditions for being a median.

4 Right to remain silent

Based on the assignment text, some probabilistic events can be defined together with their probabilities. First NC means the person has no history of convictions and C means the person has a history of convictions. We do not use the probabilities $P(NC)$ or $P(C)$. *court* means that the person goes to court. *convicted* will mean that the person is convicted in court and

- $P(court|NC, silent) = 0.001$ and $P(court|C, silent) = 0.001$, so $P(-court|silent) = 0.999$
- $P(court|NC, talks) = 0.002$, so $P(-court|NC, talks) = 0.998$
- $P(court|C, talks) = 0.005$, so $P(-court|C, talks) = 0.995$
- Naturally $P(-convicted| - court) = 1$ and $P(convicted| - court) = 0$
- $P(-convicted|court, NC, talks) = 0.5$, so $P(convicted|court, NC, talks) = 0.5$
- $P(-convicted|court, C, talks) = 0.1$, so $P(convicted|court, C, talks) = 0.9$

Since the probability of not being convicted falls 4 times if ones stays silent:

- $P(-convicted|court, NC, silent) = 0.5/4 = 0.125$, so $P(convicted|court, NC, silent) = 0.875$
- $P(-convicted|court, C, silent) = 0.1/4 = 0.025$, so $P(convicted|court, C, silent) = 0.975$

a) First we find the expected sentence duration for a person with no history of convictions which talks with the police.

Let X be a discrete random variable describing a possible sentence duration. If the person is not convicted then $X = 0$. If the person is convicted then, since they talked to the police, the sentence duration will be reduced by factor 0.75, so in that case $X = 5 \cdot 0.75 = 3.75$

Now the problem is to find

$$EX = \sum_x xp(x) = 3.75 \cdot P(X = 3.75) + 0 \cdot P(X = 0) = 3.75 \cdot P(X = 3.75)$$

As it was defined $X = 3.75$ only when the person is convicted, so

$$P(X = 3.75) = P(convicted|NC, talks)$$

By Conditional Law of Total Probability

$$P(convicted|NC, talks) = P(convicted|court, NC, talks)P(court|NC, talks) + P(convicted| - court, NC, talks)P(-court|NC, talks) = 0.5 \cdot 0.002 + 0 \cdot 0.998 = 0.001$$

So $EX = 3.75 \cdot 0.001 = 0.00375$ years.

Next, the person has no history of convictions and remains silent, so either $X' = 0$ or $X' = 5$.

$$EX' = \sum_x xp(x) = 5 \cdot P(X' = 5) + 0 \cdot P(X' = 0) = 5 \cdot P(X' = 5)$$

$$P(X' = 5) = P(convicted|NC, silent)$$

By Conditional Law of Total Probability

$$P(convicted|NC, silent) = P(convicted|court, NC, silent)P(court|NC, silent) + P(convicted| - court, NC, silent)P(-court|NC, silent) = 0.875 \cdot 0.001 + 0 \cdot 0.999 = 0.00175$$

So $EX' = 5 \cdot 0.00175 = 0.00875$ years. Since $EX < EX'$, a person with no history of convictions would rationally choose to talk with the police.

b) Now we use the same mode of computation for a person with a history of conviction who talks with the police.

$EX = 3.75 \cdot P(convicted|C, talks)$

$$P'(convicted|C, talks) = P(convicted|court, C, talks)P(court|C, talks) + P(convicted| - court, C, talks)P(-court|C, talks) = 0.9 \cdot 0.005 + 0 \cdot 0.995 = 0.0045$$

So $EX = 3.75 \cdot 0.0045 = 0.016875$

Lastly, when the person stays silent

$EX = 5 \cdot P(convicted|C, silent)$

$$P(convicted|C, silent) = P(convicted|court, C, silent)P(court|C, silent)+$$
$$P(convicted| - court, C, silent)P(-court|C, silent) =$$
$$0.975 \cdot 0.001 + 0 \cdot 0.999 = 0.000975$$

$EX = 5 \cdot 0.000975 = 0.004875$ Thus, a person with a history of convictions will minimise expected duration of sentence by staying silent.