MAD 2019, Assignment A4

Kasper Rasmussen - `wdq486`

november 2019

1 Partial derivatives

a)

$$f(x,y) = x^4y^3 + x^5 - e^y$$

$$\frac{\partial f}{\partial x} = \frac{\partial}{\partial x}x^4y^3 + \frac{\partial}{\partial x}x^5 - \frac{\partial}{\partial x}e^y = y^3\frac{\partial}{\partial x}x^4 + \frac{\partial}{\partial x}x^5 - 0 = y^3(4x^3) + (5x^4) = 4y^3x^3 + 5x^4$$

$$\frac{\partial f}{\partial y} = \frac{\partial}{\partial y}x^4y^3 + \frac{\partial}{\partial y}x^5 - \frac{\partial}{\partial y}e^y = x^4\frac{\partial}{\partial y}y^3 + 0 - e^y = 3x^4y^2 - e^y$$

b)

$$f(x,y) = \frac{1}{\sqrt{x^3 + xy + y^2}} = (x^3 + xy + y^2)^{-\frac{1}{2}}$$

.

Using chain rule with outer function $g(s) = \frac{1}{\sqrt{s}}$. Then $g'(s) = -\frac{1}{2}(s)^{-\frac{3}{2}}$

$$\frac{\partial f}{\partial x} = -\frac{1}{2}(x^3 + xy + y^2)^{-\frac{3}{2}} \cdot \frac{\partial}{\partial x}(x^3 + xy + y^2) = -\frac{1}{2}(x^3 + xy + y^2)^{-\frac{3}{2}} \cdot (3x^2 + y)$$

$$\frac{\partial f}{\partial y} = -\frac{1}{2}(x^3 + xy + y^2)^{-\frac{3}{2}} \cdot \frac{\partial}{\partial y}(x^3 + xy + y^2) = -\frac{1}{2}(x^3 + xy + y^2)^{-\frac{3}{2}} \cdot (x + 2y)$$

c)

$$f(x,y) = \frac{x^3 + y^2}{x + y}$$

Using quotient rule

$$\frac{\partial f}{\partial x} = \frac{(x+y)\frac{\partial}{\partial x}(x^3 + y^2) - (x^3 + y^2)\frac{\partial}{\partial x}(x+y)}{(x+y)^2} = \frac{(x+y)(3x^2) - (x^3+y^2)(1)}{(x+y)^2} =$$

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

$$\frac{\partial f}{\partial y} = \frac{(x+y)\frac{\partial}{\partial y}(x^3 + y^2) - (x^3 + y^2)\frac{\partial}{\partial y}(x+y)}{(x+y)^2} = \frac{(x+y)(2y) - (x^3+y^2)(1)}{(x+y)^2} =$$

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

2 Gradients

a)

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{x} + c$$

By showing partial derivative w.r.t. each component

$$\frac{\partial f}{\partial x_i} = \frac{\partial}{\partial x_i}\mathbf{x}^T\mathbf{x} + \frac{\partial}{\partial x_i}c = \frac{\partial}{\partial x_i}(x_1x_1 + x_2x_2 + ... + x_nx_n) + 0 =$$

$$\frac{\partial}{\partial x_i}x_1^2 + \frac{\partial}{\partial x_i}x_2^2 + ... + \frac{\partial}{\partial x_i}x_i^2 + ... + \frac{\partial}{\partial x_i}x_n^2 = 0 + 0 + ... + 2x_i + ... + 0 = 2x_i$$

So gradient $\frac{\partial f}{\partial \mathbf{x}} = (2x_1, 2x_2, ..., 2x_n)^T = 2 \cdot (x_1, x_2, ..., x_n)^T = 2\mathbf{x}$

b)

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{b} = x_1b_1 + x_2b_2 + ... + x_nb_n$$

Partial derivative w.r.t. to x_i. Sum rule

$$\frac{\partial f}{\partial x_i} = \frac{\partial}{\partial x_i}x_1b_1 + \frac{\partial}{\partial x_i}x_2b_2 + ... + \frac{\partial}{\partial x_i}x_ib_i + ... + \frac{\partial}{\partial x_i}x_nb_n$$

$$= 0 + 0 + ... + b_i + ... + 0 = b_i$$

So gradient $\frac{\partial f}{\partial \mathbf{x}} = (b_1, b_2, ..., b_n)^T = \mathbf{b}$

c)

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c$$

$$\frac{\partial f}{\partial \mathbf{x}} = \frac{\partial}{\partial \mathbf{x}}\mathbf{x}^T\mathbf{A}\mathbf{x} + \frac{\partial}{\partial \mathbf{x}}\mathbf{b}^T\mathbf{x} + \frac{\partial}{\partial \mathbf{x}}c$$

Starting with $\frac{\partial}{\partial \mathbf{x}}\mathbf{b}^T\mathbf{x}$. Since

$$\mathbf{b}^T\mathbf{x} = \sum_{k=1}^{n} b_kx_k = \mathbf{x}^T\mathbf{b}$$

We know from above that $\frac{\partial}{\partial \mathbf{x}}\mathbf{b}^T\mathbf{x} = \mathbf{b}$ and $\frac{\partial}{\partial \mathbf{x}}c = 0$. This leaves the first term $\frac{\partial}{\partial \mathbf{x}}\mathbf{x}^T\mathbf{A}\mathbf{x}$, where we first look at each x_i

$$\frac{\partial}{\partial x_i}\mathbf{x}^T\mathbf{A}\mathbf{x} = \frac{\partial}{\partial x_i}\sum_{k=1}^{n} x_k \sum_{m=1}^{n} \mathbf{A}_{km}x_m = \frac{\partial}{\partial x_i}\sum_{k=1}^{n}\sum_{m=1}^{n} \mathbf{A}_{km}x_mx_k = \sum_{k=1}^{n}\sum_{m=1}^{n}\frac{\partial}{\partial x_i}\mathbf{A}_{km}x_mx_k$$

Looking at terms in 4 cases.

$k \neq i$ and $m \neq i$:

$$\frac{\partial}{\partial x_i}\mathbf{A}_{km}x_mx_k = 0$$

$k = i$ and $m \neq i$:

$$\frac{\partial}{\partial x_i}\mathbf{A}_{km}x_mx_k = \mathbf{A}_{km}x_m = \mathbf{A}_{im}x_m$$

$k \neq i$ and $m = i$:

$$\frac{\partial}{\partial x_i}\mathbf{A}_{km}x_mx_k = \mathbf{A}_{km}x_k = \mathbf{A}_{ki}x_k$$

$k = i$ and $m = i$

$$\frac{\partial}{\partial x_i}\mathbf{A}_{km}x_mx_k = \frac{\partial}{\partial x_i}\mathbf{A}_{ii}x_i^2 = 2\mathbf{A}_{ii}x_i$$

So

$$\frac{\partial}{\partial x_i}\mathbf{x}^T\mathbf{A}\mathbf{x} = 2A_{ii}x_i + \sum_{m=1, m\neq i}^{n} A_{im}x_m + \sum_{k=1, k\neq i}^{n} A_{ki}x_k = \sum_{m=1}^{n} A_{im}x_m + \sum_{k=1}^{n} A_{ki}x_k =$$

$$\mathbf{x}^T(\mathbf{A}^T)_i + \mathbf{x}^T\mathbf{A}_i = \mathbf{x}^T((\mathbf{A}^T)_i + \mathbf{A}_i) = \mathbf{x}^T(\mathbf{A}^T + \mathbf{A})_i$$

So

$$\frac{\partial}{\partial x}\mathbf{x}^T\mathbf{A}\mathbf{x} = (\mathbf{x}^T(\mathbf{A}^T + \mathbf{A})_1, ..., \mathbf{x}^T(\mathbf{A}^T + \mathbf{A})_n) =$$

$$(\mathbf{A}^T + \mathbf{A})^T\mathbf{x} = ((\mathbf{A}^T)^T + \mathbf{A}^T)\mathbf{x} = (\mathbf{A} + \mathbf{A}^T)\mathbf{x}$$

The conclusion is that

$$\frac{\partial f}{\partial \mathbf{x}} = (\mathbf{A} + \mathbf{A}^T)\mathbf{x} + \mathbf{b}$$

3 House Prices I

The mean of the prices in the training set is computed and is the prediction for every house in the training set. The root mean squared error for this model is computed and a scatter plot is produced, showing the relationship betwen the true and estimated prices of the houses in the test set. This is how housing_1.py has been extended:

Listing 1: Extension of housing_1.py

```python
# (a) compute mean of prices on training set
mean_price = np.sum(t_train) / X_train.shape[0]
print("a) Mean price on training set: " + str(mean_price))

# (b) RMSE function
def rmse(t, tp):
    n = X_train.shape[0]
    return np.sqrt(1/n * np.sum((t - tp)**2))

array_with_mean = np.full((X_train.shape[0], 1), mean_price)
rmse_house = rmse(array_with_mean, t_train)
print("b) RMSE when using mean as prediction: " + str(rmse_house))

# (c) visualization of results
plt.scatter(t_train, array_with_mean)
plt.xlabel("True price")
plt.ylabel("Estimated price")
plt.show()
```

Running housing_1.py gives the following and the scatter plot.

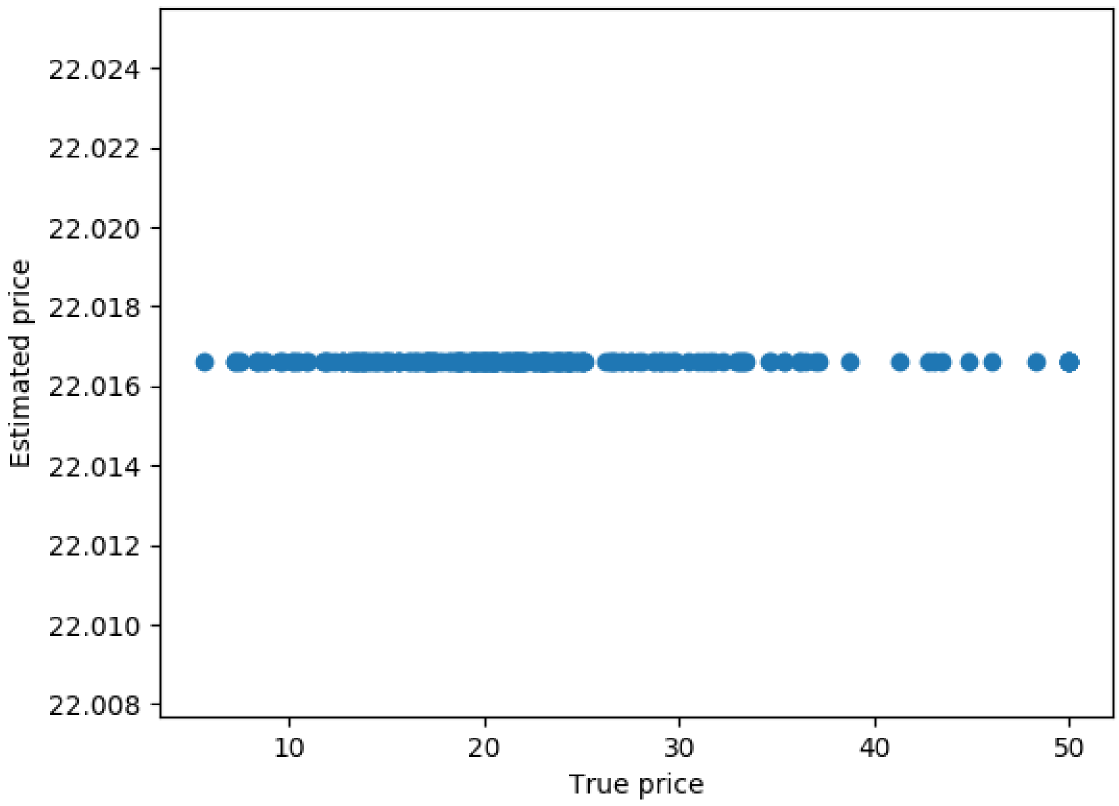
```
> python3 housing_1.py
Number of training instances: 253
Number of test instances: 253
Number of features: 13
a) Mean price on training set: 22.016600790513834
b) RMSE when using mean as prediction: 8.707193490723665
```

The mean is is approximately 22. The RMSE is approximately 8.7. The script also shows the scatter plot in Figure 1.

4 House Prices II

Using the results from RG section 1.3, a Python class for least squares linear regression is implemented with a fit and predict method.

First, fit augments the training data matrix X with a column of ones, for the offset w_0. Then it solves the equation $\mathbf{X}^T\mathbf{X} = \mathbf{X}^\mathbf{T}\mathbf{t}$ from (1.15) in RG in order to determine the optimal parameters W which are saved in the private w field.



Figur 1: Scatterplot from `housing_1.py`

Second, `predict` takes the test data matrix and computes the predictions as **Xw** and this approach follows from section 1.3.3 in RG.

Listing 2: Implementation of linreg.py

```
import numpy as np
class LinearRegression():
    def __init__(self):
        self.w = []
        pass

    def augment(self, X):
        if(len(X.shape) == 1):
            ones = np.ones(len(X))
            return np.transpose(np.stack((ones, X)))
        else:
            ones = np.ones((X.shape[0], 1))
            return np.concatenate((ones, X), axis=1)

    def fit(self, X, t):
        X = self.augment(X)
        A = np.dot(X.T,X)
        B = np.dot(X.T,t)
        W = np.linalg.solve(A,B)
        self.w = W

    def predict(self, X):
        X = self.augment(X)
        return np.matmul(X,self.w)
```

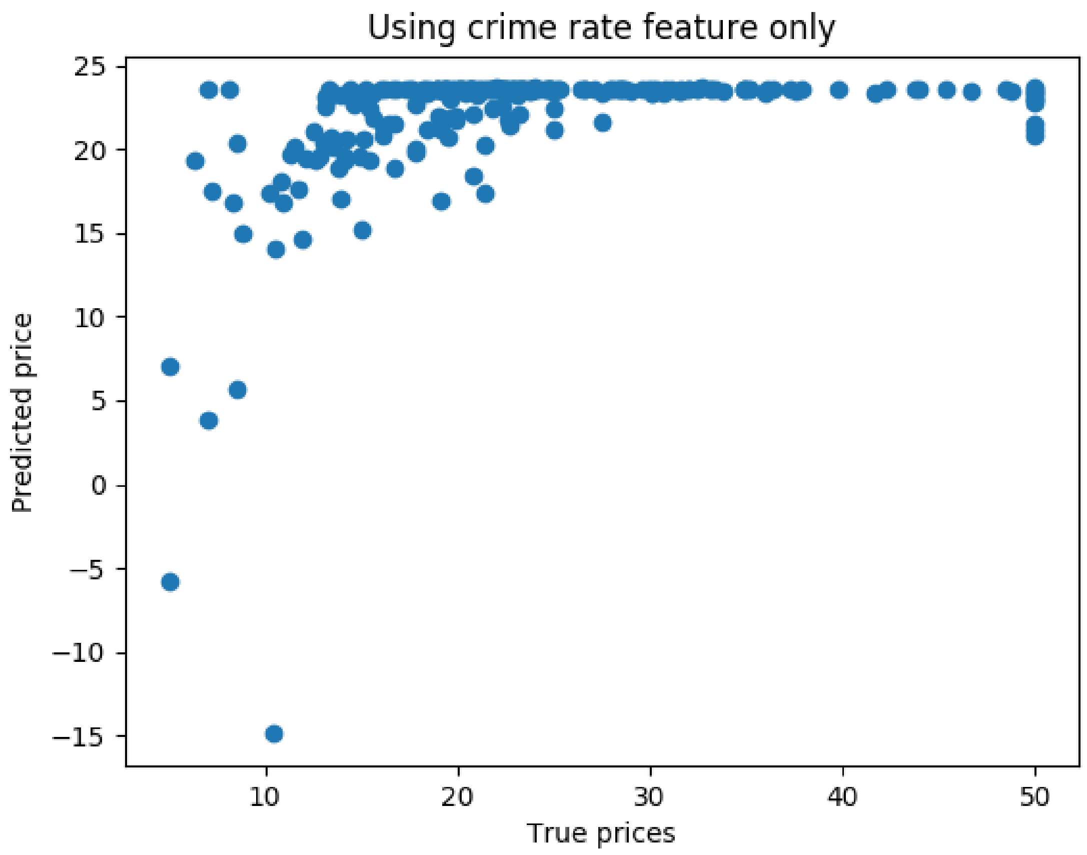
Next, we use the implementation to make linear models of the prices of houses in Bostom.

Listing 3: Extension of housing_2.py

```python
def rmse(t, tp):
    n = X_train.shape[0]
    return np.sqrt(1/n * np.sum((t - tp)**2))

# (b) fit linear regression using only the first feature
model_single = linreg.LinearRegression()
model_single.fit(X_train[:,0], t_train)
prediction_single = model_single.predict(X_test[:,0])
rmse_single = rmse(prediction_single, t_test)
print("\n(b) RMSE using single feature: " + str(rmse_single))
print("Weights:")
print("w0: " + str(model_single.w[0][0]))
print("w1: " + str(model_single.w[1][0]))

# (c) fit linear regression model using all features
model_multi = linreg.LinearRegression()
model_multi.fit(X_train, t_train)

print("\n(c) Weight using all features:")
print(model_multi.w)

# (d) evaluation of results
prediction_multi = model_multi.predict(X_test)
rmse_multi = rmse(prediction_multi, t_test)
print("\n(c) RMSE using all features: " + str(rmse_multi))

# Scatter plots
plt.scatter(t_test, prediction_single)
plt.xlabel("True prices")
plt.ylabel("Predicted price")
plt.title("Using crime rate feature only")
plt.show()

plt.scatter(t_test, prediction_multi)
plt.xlabel("True price")
plt.ylabel("Predicted price")
plt.title("Using all features")
plt.show()
```

Running housing_2.py gives the following and two scatter plots.

```
> python3 housing_2.py
Number of training instances: 253
Number of test instances: 253
Number of features: 13

(b) RMSE using single feature: 8.954859906611233
Weights:
w0: 23.63506195389148
w1: -0.43279318038902775

(c) Weight using all features:
[[ 3.13886978e+01]
 [-5.96169127e-02]
 [ 2.93672792e-02]
 [-2.90605834e-02]
 [ 2.29256181e+00]
```



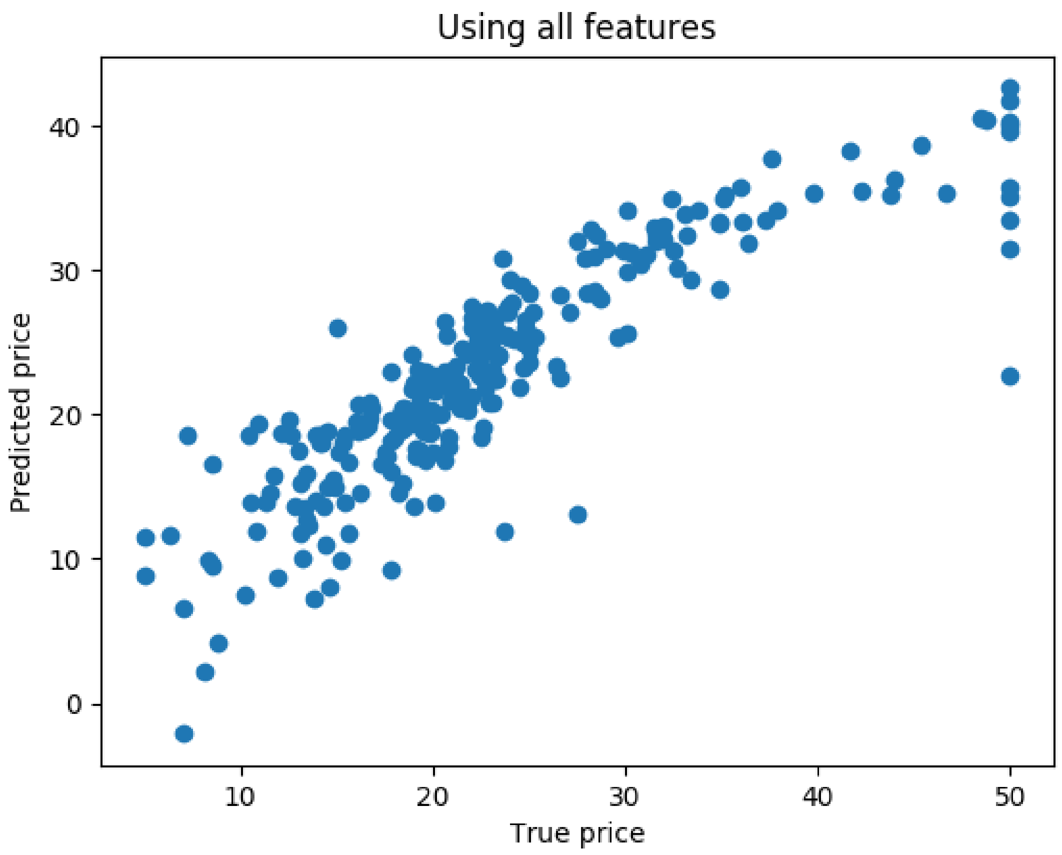
Figur 2: Scatter plot for `housing_2.py` (b)

```
[-1.73263655e+01]
[ 3.99375996e+00]
[ 3.23077761e-03]
[-1.28724508e+00]
[ 3.54780191e-01]
[-1.55819191e-02]
[-8.14647713e-01]
[ 1.17820208e-02]
[-4.64869014e-01]]
```

```
(c) RMSE using all features: 4.68833365362771
```

In (b), using the class, linear regression was made on the Boston Housing Dataset, using only the first feature *crime rate*, by letting the training data matrix be `X_train[:,0]` as shown in line 7 of Listing 3. By computing the predictions on the test data using `predict`, the RMSE was evaluated to approximately 8.95. By using one feature we get 2 parameters w_0 and w_1 which are the coefficients of the line $y = ax + b$ which is our model. We get $w_0 \approx 23.635$ and $w_1 \approx -0.43$, meaning house prices and crime rates are negatively correlated. The root mean squared error on the test data is approximately 8.95. The related scatter plot is shown in Figure 2.

In (c) we use the whole training data matrix X which has 13 features so we 14 weights, shown above, and the computed root mean squared error on the test data is approximately 4.69. The related scatter plot is show in Figure 3.



Figur 3: Scatter plot for `housing_2.py` using all features

5 Total training loss

Let L' be the loss function from definition (1.3) in RG. Then $L = NL'$. It is shown on page 22, that $L' = \frac{1}{N}(\mathbf{w}^T\mathbf{X}\mathbf{w} - 2\mathbf{w}^T\mathbf{X}^T\mathbf{t} + \mathbf{t}^T\mathbf{t})$ so $L = \mathbf{w}^T\mathbf{X}\mathbf{w} - 2\mathbf{w}^T\mathbf{X}^T\mathbf{t} + \mathbf{t}^T\mathbf{t}$

$$\nabla L = \frac{\partial L}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}}\mathbf{w}^T(\mathbf{X}^T\mathbf{X})\mathbf{w} - 2\frac{\partial}{\partial \mathbf{w}}\mathbf{w}^T(\mathbf{X}^T) + \frac{\partial}{\partial \mathbf{w}}\mathbf{t}^T\mathbf{t} = 2(\mathbf{X}^T\mathbf{X})\mathbf{w} - 2\mathbf{X}^T\mathbf{t}$$

by rule 1 and 4 on page 23. So $\nabla L = \nabla L'$ and thus the optimal solution is the same as shown on page 25

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

5.1 Derivation without L'

$$L = \sum_{n=1}^{n}(\mathbf{w^T x_n} - t_n)^2 = (\mathbf{X}\mathbf{w} - \mathbf{t})^T(\mathbf{X}\mathbf{w} - \mathbf{t}) = ((\mathbf{X}\mathbf{w})^T) = ((\mathbf{X}\mathbf{w})^T - t^T)(\mathbf{X}\mathbf{w} - \mathbf{t}) =$$

$$(\mathbf{X}\mathbf{w})^T(\mathbf{X}\mathbf{w}) - (\mathbf{X}\mathbf{w})^T\mathbf{t} - \mathbf{t}^T(\mathbf{X}\mathbf{w}) + \mathbf{t}^T\mathbf{t} = \mathbf{w}^T\mathbf{X^T}\mathbf{X}\mathbf{w} - 2\mathbf{w}^T\mathbf{X}^T\mathbf{t} + \mathbf{t}^T\mathbf{t}$$

The last step is made since $(\mathbf{X}\mathbf{w})^T\mathbf{t} = \mathbf{t}^T(\mathbf{X}\mathbf{w})$ because of symmetry $(\mathbf{X}\mathbf{w})^T\mathbf{t})^T = \mathbf{t}^T(\mathbf{X}w)^T$. They are 1×1 matrices, scalars.

Now,

$$\nabla L = \frac{\partial L}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}}\mathbf{w}^T(\mathbf{X}^T\mathbf{X})\mathbf{w} - 2\frac{\partial}{\partial \mathbf{w}}\mathbf{w}^T(\mathbf{X}^T) + \frac{\partial}{\partial \mathbf{w}}\mathbf{t}^T\mathbf{t}$$

By rule 4 and 1 in Table 1.4 in RG we get

$$\nabla L = 2(\mathbf{X}^T\mathbf{X})\mathbf{w} - 2\mathbf{X}^T\mathbf{t}$$

For optimality, ∇L must vanish. Enforcing

$$\nabla L = 0$$

$$2(\mathbf{X}^T\mathbf{X})\mathbf{w} - 2\mathbf{X}^T\mathbf{t} = 0$$

$$2(\mathbf{X}^T\mathbf{X})\mathbf{w} = 2\mathbf{X}^T\mathbf{t}$$

$$(\mathbf{X}^T\mathbf{X})\mathbf{w} = \mathbf{X}^T\mathbf{t}$$

Premultiplication by $(\mathbf{X}^T\mathbf{X})^{-1}$, assuming it gives

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