Assignment 1

Aleksandr Salo

Due Feb 3, 2015

1 Maximum likelihood for linear regression (10 points)

1. Write down the likelihood function $L(\theta)$:

$$\begin{aligned} L(\theta) &= P(y|x;\theta) \\ &= \prod_{i=1}^{m} P(y^{(i)}|x^{(i)};\theta) \\ &= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{1}{2\sigma^2}(y^{(i)} - \theta x^{(i)})^2) \end{aligned}$$

Then take the logarithm of the likelihood and simplify it.

$$\begin{aligned} \ell(\theta) &= \log(L(\theta)) \\ &= \Sigma_{i=1}^{m} \log(\frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{1}{2\sigma^2}(y^{(i)} - \theta x^{(i)})^2)) \\ &= \Sigma_{i=1}^{m} [\log(\frac{1}{\sqrt{2\pi\sigma^2}}) - \frac{1}{2\sigma^2}(y^{(i)} - \theta x^{(i)})^2)] \end{aligned}$$

2. Take the derivative of $\ell(\theta)$ with respect to θ.

$$\begin{aligned} \frac{\partial \ell(\theta)}{\partial \theta} &= -\frac{2}{2\sigma^2} \Sigma_{i=1}^{m} ((y^{(i)} - \theta x^{(i)})(-x)) \\ &= \frac{1}{\sigma^2} \Sigma_{i=1}^{m} ((y^{(i)} - \theta x^{(i)})x) \end{aligned}$$

Then use this to find the maximum likelihood value of θ. That is when $\frac{\partial \ell(\theta)}{\partial \theta} = 0$.

$$\begin{aligned} y^{(i)}x^{(i)} - \theta x^{(i)}x^{(i)} &= 0 \\ \theta x^{(i)}x^{(i)}(x^{(i)}x^{(i)})^{-1} &= y^{(i)}x^{(i)}(x^{(i)}x^{(i)})^{-1} \\ \theta &= y^{(i)}x^{(i)}(x^{(i)}x^{(i)})^{-1} \end{aligned}$$

Now, in the linear algebra form that gives:

$$\theta = y^T x (x^T x)^{-1}$$

Which is basically doesn't differ from n-dimensional case.

The figure below illustrates the result of finding θ with ML for linear regression

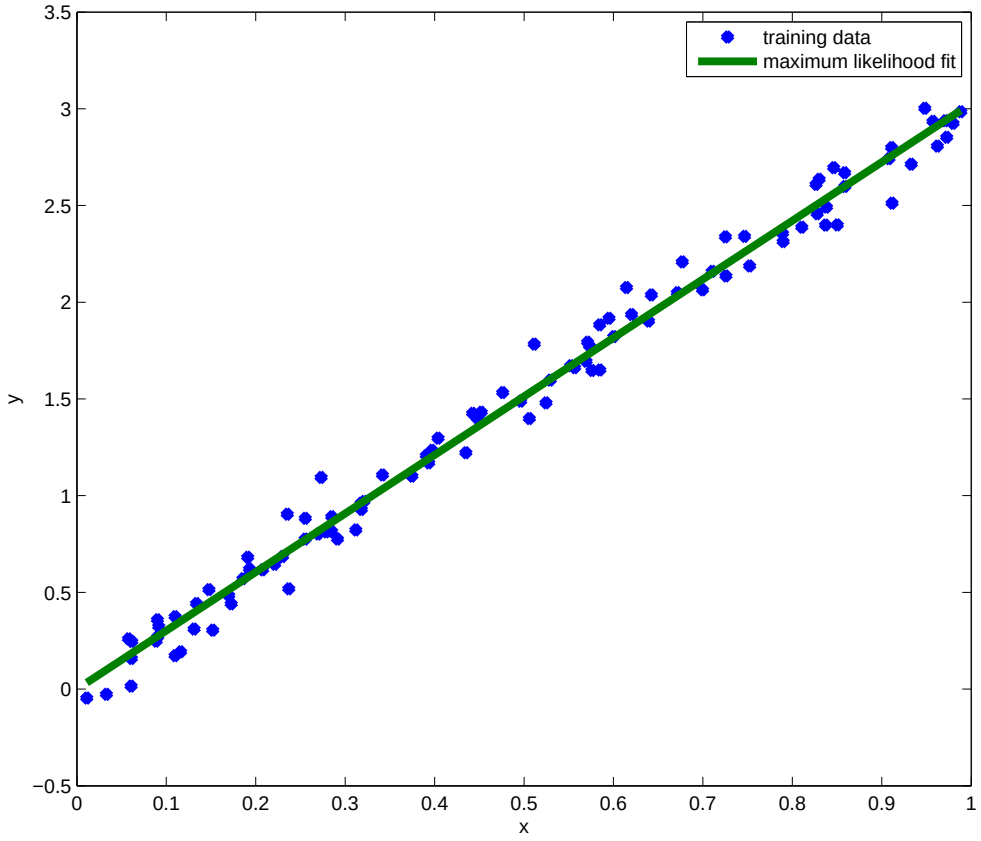


Figure 1: Estimation of θ with ML for linear regression

2 Locally-weighted linear regression (20 points)

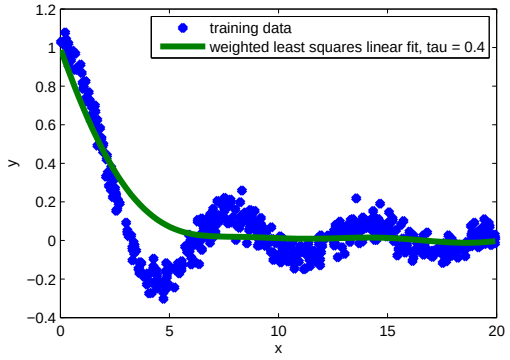
We need to fit θ to optimize cost function:

$$\Sigma_{i=1}^{m} w^{(i)} (\theta^T x^{(i)} - y^{(i)})^2$$

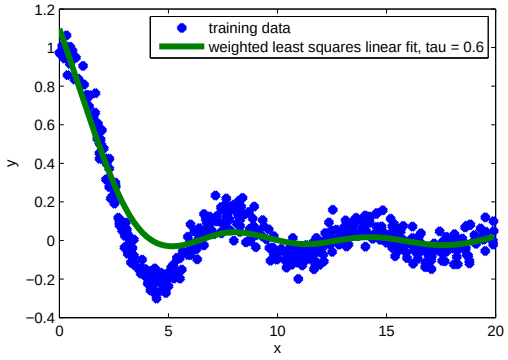
Where w are weights obtained by Gaussian kernel:

$$w^{(i)} = exp(-\frac{||x^{(i)} - x||^2}{2\tau^2})$$

In the example below training data is generated with sin function and some Gaussian noise. LWR gives the different fits with various τ:



(a) $\tau = 0.4$: poor fit, oversimplified prediction



(b) $\tau = 0.6$: nice fit, reasonable prediction

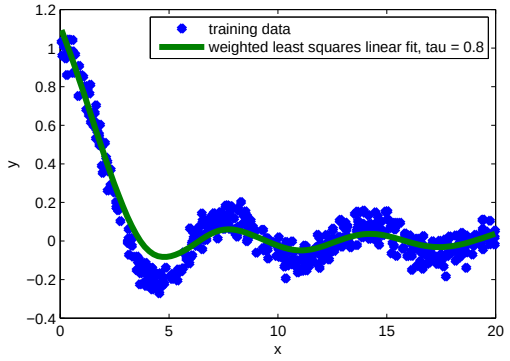


Figure 3: $\tau = 0.8$: good fit, better prediction

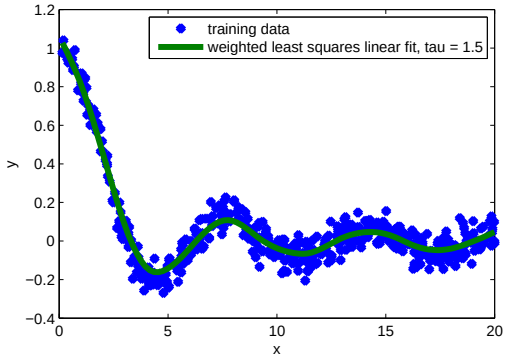


Figure 4: $\tau = 1.5$: good fit, very accurate prediction

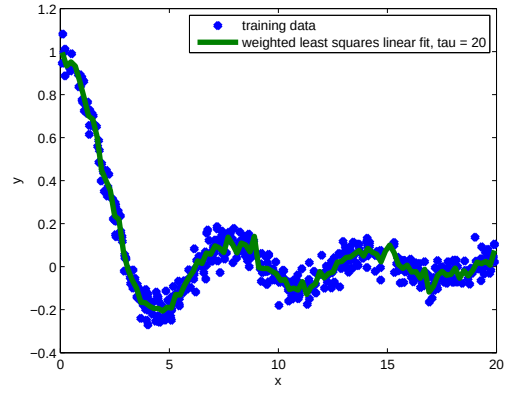


Figure 5: $\tau = 20$: overfitting, useless prediction

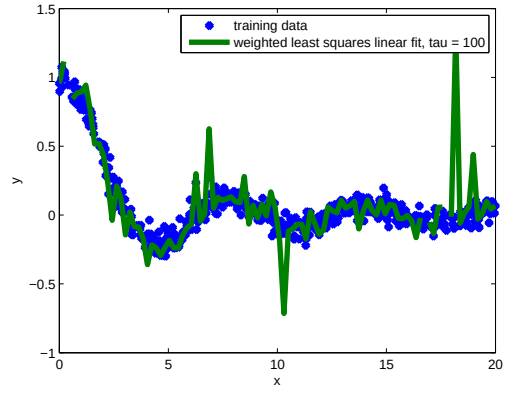


Figure 6: $\tau = 100$: Heavy overfitting, wrong prediction

3 Gradient descent, Newton's method, and logistic regression (30 points)

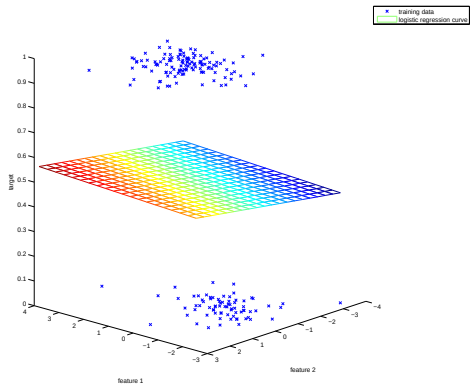
Although by implementing and using both algorithms we achieve convergence on learning for provided dataset, there are significant difference in approaches.

3.1 Gradient Descent

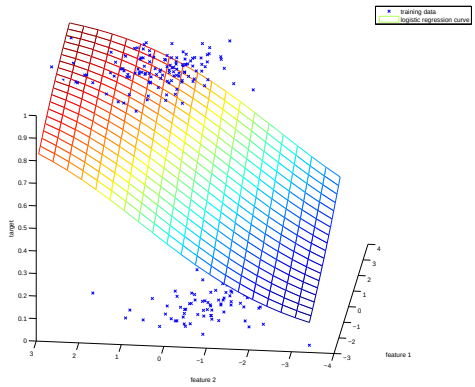
First, let us take a look at Gradient Descent for Logistic Regression. The learning step is:

$$\theta \leftarrow \theta + \alpha \nabla_\theta l(\theta)$$

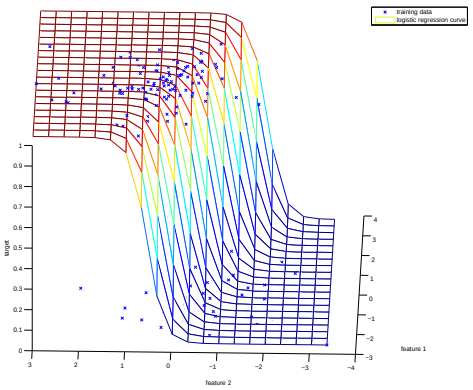
First thing to notice: we have to choose learning rate α which is often undesirable for *machine* learning. After first iteration we get different pictures depending on learning rate:



(a) $\alpha = 0.001$: Almost no learning



(b) $\alpha = 0.01$: reasonable improvement



(c) $\alpha = 0.1$: seems like almost there

Loigcally, higher the learning rate - faster the convergence (we will discuss the conditions later). However, that is not that simple.

1. Indeed, convergence happens faster for $\alpha = 0.01$ than $\alpha = 0.001$

2. Too big learning rate, i.e. 0.1 leads to "jumping" behavior and not convergence at all. However this could be solved by decrementing learning rate by some percentage after each iteration.

Conditions for convergence are:

1. accuracy = 1, where accuracy is number correct guesses divided by number of examples
2. norm(H) < 1e-10
3. norm(grad) < 1e-10

While dataset provided has overlapping values we never reach the accuracy condition. However, we can add another condition about accuracy not changing much (or at all). For instance, for given dataset, maximum accuracy is 0.835. Method described above converges to this value over 9 iterations (with $\alpha = 0.01$).

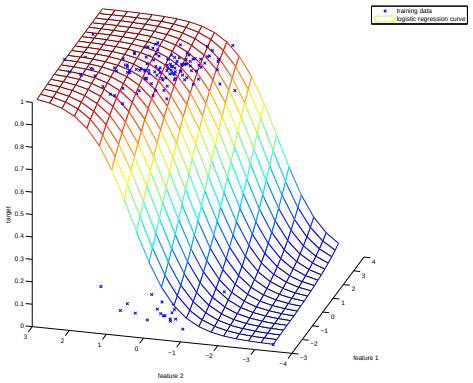
3.2 Newton's method

For Newton's method, the learning step is:

$$\theta \leftarrow \theta - H' \nabla_\theta l(\theta)$$

where H is a Hessian matrix, which basically contains second partial derivatives.

Intuitively, Newton's method converges faster. On provided dataset we have this picture after two iterations: That is almost maximum accuracy that possible on the dataset. Method converges after 7 iterations because



(a) Newton's method after 2 iteration: accuracy = 0.830

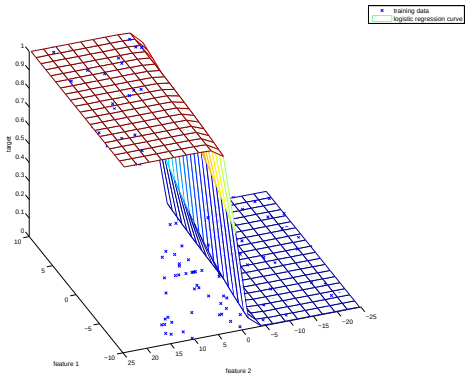
of norm(H) condition.

3.3 Convenient dataset

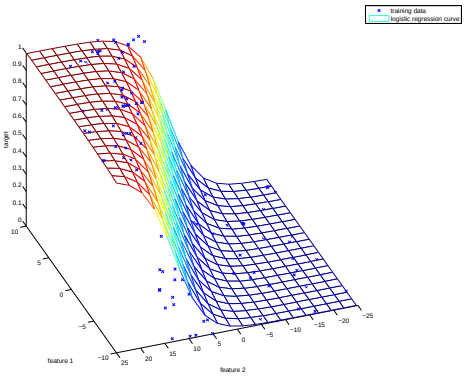
For testing purposes we create convenient dataset with clear division:

```
testx1 = rand(m, 1) * 20 - 10; %generate some rand x
testx2 = rand(m, 1) * 50 - 25;
testy = zeros(m, 1);
%make separation clear
testx1((testx2 + testx1) < 10 & (testx2 + testx1) > 7) = 0;
testx2((testx2 + testx1) < 10 & (testx2 + testx1) > 7) = 0;
testy((testx2 + testx1) > 10) = 1; % arbitrary classification
testx = [testx1, testx2]; %combine features
```

Gradient Descent doesn't converge after 100 iterations:

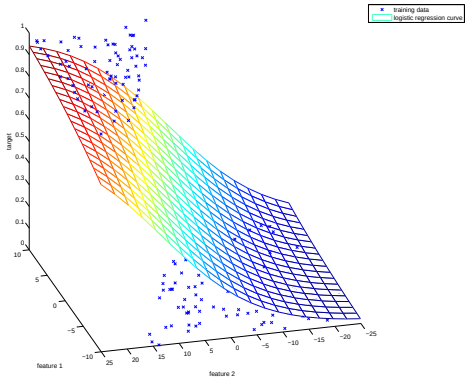


(a) Gradient Descent after 1 step

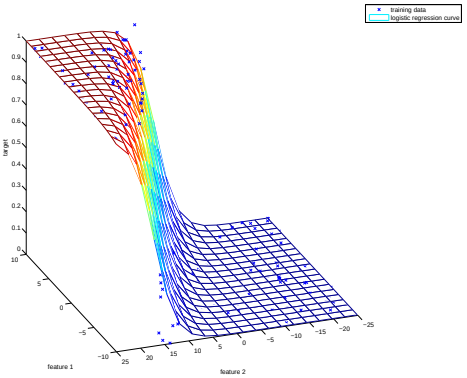


(b) Gradient Descent after 100 steps. Accuracy = 0.93

Newton's method converges after 4 iterations with perfect accuracy:



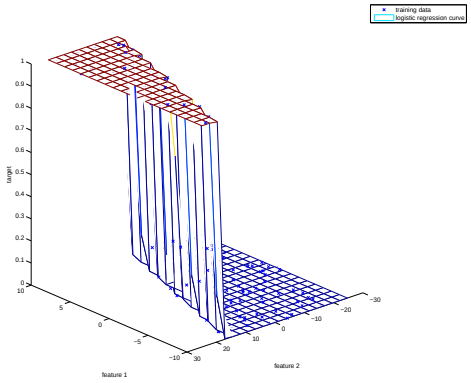
(a) Newton's method after 1 step



(b) Converges after 4 iterations with accuracy = 1

3.4 Theta behavior without accuracy condition

If we eliminate stopping condition of accuracy, both methods tend to amplify θ thus narrowing the Sigmoid's "bump".



(a) Newton's method after 13 iteration without accuracy stopping condition

That behavior continues until norma of gradient or Hessian matrix is too small. For test dataset we receive θ = (-133.0180 15.6132 15.7568) instead of appropriate (-4.9162 0.5683 0.6271).

4 Extra credit 1: deriving logistic regression (+5 points)

Show the derivations for the gradient and the Hessian of the log-likelihood for logistic regression.
We'll use the following rules of derivative:

1. $log'(x) = \frac{1}{x}$
2. $(f(g(x)))' = f'(g(x)) * g'(x)$
3. $f'(x) = f(x)(1 - f(x))$
4. ∇ is a set of partial derivatives

So if:

$$\begin{aligned}\ell(\theta) &= logL(\theta) \\ &= \sum_{i=1}^{m} y^{(1)} \log h_\theta(x^{(i)}) + (1 - y^{(1)}) \log(1 - h_\theta(x^{(i)}))\end{aligned}$$

Where:

$$h_\theta(x) = g(\theta^T x)$$

Then partial derivative for one example (x, y) with respect of θ_j:

$$\frac{\partial}{\partial \theta_j}\ell(\theta) = (y\frac{1}{g(\theta^T x)} - (1 - y)\frac{1}{g(1 - \theta^T x)})\frac{\partial}{\partial \theta_j} g(\theta^T x)$$

By (2) and (3):

$$\begin{aligned}\frac{\partial}{\partial\theta_j}g(\theta^T x) &= g(\theta^T x)(1-g(\theta^T x))\frac{\partial}{\partial\theta_j}\theta^T x \\ &= g(\theta^T x)(1-g(\theta^T x))x_j\end{aligned}$$

Plug it back and simplify:

$$\begin{aligned}\frac{\partial}{\partial\theta_j}\ell(\theta) &= (y(1-g(\theta^T x)) - (1-y)g(\theta^T x))x_j \\ &= (y-g(\theta^T x))x_j\end{aligned}$$

Then the gradient for several features for one example would be a vector with values for various j:

$$(y^{(i)} - g(\theta^T x^{(i)}))x^{(i)}$$

Thus gradient for all the examples:

$$\nabla_\theta\ell(\theta) = \sum_{i=1}^{m}[y^{(i)} - g(\theta^T x^{(i)})]x^{(i)}$$

Now, Hessian matrix is just another derivative by θ:

$$H_{jk} = \frac{\partial^2\ell(\theta)}{\partial\theta_j\partial\theta_k}$$

Using rule (3) again for one example (x, y) feature we have:

$$\begin{aligned}\frac{\partial}{\partial\theta_k}(y-g(\theta^T x))x_j &= \frac{\partial}{\partial\theta_k}(yx_j - g(\theta^T x)x_j) \\ &= -x_j(g(\theta^T x)(1-g(\theta^T x))\frac{\partial}{\partial\theta_k}\theta^T x) \\ &= -x_j x_k g(\theta^T x)(1-g(\theta^T x))\end{aligned}$$

Thus Hessian matrix:

$$H_{jk} = -\sum_{i=1}^{m} x_j^{(i)} x_k^{(i)} g(\theta^T x^{(i)})(1-g(\theta^T x^{(i)}))$$

5 Extra credit 2: weighted logistic regression (+5 points)

Consider applying the idea of locally-weighted examples to the logistic regression model. Then prediction for single example is:

$$h_\theta(x) = g(w\theta^T x)$$

Taking the derivative of corresponding log-likelihood function with respect to θ_j gives:

$$(y-g(\theta^T x))wx_j$$

Then gradient for all the examples:

$$\nabla_\theta \ell(\theta) = \sum_{i=1}^{m} [y^{(i)} - g(\theta^T x^{(i)})] w^{(i)} x^{(i)}$$

Finally Hessian matrix by applying the same rules is:

$$H_{jk} = -\sum_{i=1}^{m} (w^{(i)})^2 x_j^{(i)} x_k^{(i)} g(\theta^T x^{(i)})(1 - g(\theta^T x^{(i)}))$$