CSE 5523: Lecture Notes 10
Linear regression

Contents

10.1 Linear regression

Fit weights $\mathbf{w}$ for numerical y (e.g. purchases) on continuous variables $\mathbf{x}$ (e.g. dimensions of fruits).

Use a **negative log Gaussian loss** function, so deviation from prediction $\mathbf{w}^\top\mathbf{x}$ is Gaussian 'noise':

$$\begin{aligned}
\mathsf{L}_{\text{NLG},\sigma}(y, \mathbf{w}^\top\mathbf{x}) &= -\ln \mathcal{N}_{\mathbf{w}^\top\mathbf{x},\sigma}(y) \\
&= -\ln\left[\left(\frac{1}{2\pi\sigma^2}\right)^{\frac{1}{2}} \exp\left(-\frac{1}{2\sigma^2}(y - \mathbf{w}^\top\mathbf{x})^2\right)\right]
\end{aligned}$$

Find parameters $\mathbf{w}$ that minize negative log Gaussian likelihood (squared) error over all data:

$$\begin{aligned}
0 &= \frac{\partial}{\partial \mathbf{w}} \sum_{\langle y,\mathbf{x}\rangle\in\mathcal{D}} \mathsf{L}_{\text{NLG},\sigma}(y, \mathbf{w}^\top\mathbf{x}) \\
&= \frac{\partial}{\partial \mathbf{w}} \sum_{\langle y,\mathbf{x}\rangle\in\mathcal{D}} -\ln\left[\left(\frac{1}{2\pi\sigma^2}\right)^{\frac{1}{2}} \exp\left(-\frac{1}{2\sigma^2}(y - \mathbf{w}^\top\mathbf{x})^2\right)\right] && \text{substitution} \\
&= \sum_{\langle y,\mathbf{x}\rangle\in\mathcal{D}} \frac{\partial}{\partial \mathbf{w}} -\ln\left[\left(\frac{1}{2\pi\sigma^2}\right)^{\frac{1}{2}} \exp\left(-\frac{1}{2\sigma^2}(y - \mathbf{w}^\top\mathbf{x})^2\right)\right] && \text{sum rule} \\
&= \sum_{\langle y,\mathbf{x}\rangle\in\mathcal{D}} \frac{\partial}{\partial \mathbf{w}} -\ln\left(\frac{1}{2\pi\sigma^2}\right)^{\frac{1}{2}} - \frac{1}{2\sigma^2}(y - \mathbf{w}^\top\mathbf{x})^2 && \text{log of product is sum of logs} \\
&= \sum_{\langle y,\mathbf{x}\rangle\in\mathcal{D}} \frac{\partial}{\partial \mathbf{w}} - \frac{1}{2\sigma^2}(y - \mathbf{w}^\top\mathbf{x})^2 && \text{derivative of constant} \\
&= \sum_{\langle y,\mathbf{x}\rangle\in\mathcal{D}} -\frac{1}{2\sigma^2} \frac{\partial}{\partial \mathbf{w}}(y - \mathbf{w}^\top\mathbf{x})^2 && \text{product rule}
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{\langle y,\mathbf{x}\rangle\in\mathcal{D}} -\frac{1}{2\sigma^2}2(y-\mathbf{w}^\top\mathbf{x})\frac{\partial}{\partial\mathbf{w}}(y-\mathbf{w}^\top\mathbf{x}) && \text{power rule}\\
&= \sum_{\langle y,\mathbf{x}\rangle\in\mathcal{D}} -\frac{1}{2\sigma^2}2(y-\mathbf{w}^\top\mathbf{x})(-\mathbf{x}) && \text{power rule}\\
&= \sum_{\langle y,\mathbf{x}\rangle\in\mathcal{D}} \mathbf{x}(y-\mathbf{w}^\top\mathbf{x}) && \text{multiply by } \sigma^2\\
&= \mathbf{X}^\top(\mathbf{y}-\mathbf{X}\mathbf{w}) && \text{definition of inner product}\\
&= \mathbf{X}^\top\mathbf{y}-\mathbf{X}^\top\mathbf{X}\mathbf{w} && \text{distributive axiom}
\end{aligned}
$$

This has root where $\mathbf{X}^\top\mathbf{X}\mathbf{w} = \mathbf{X}^\top\mathbf{y}$, or $\mathbf{w} = \underbrace{(\mathbf{X}^\top\mathbf{X})^{-1}}_{\text{concentration/precision matrix}}\mathbf{X}^\top\mathbf{y}$.

10.2 Sample code

Sample code in pandas:

```
import sys
import numpy
import pandas

X = pandas.read_csv( sys.argv[1] )
y = pandas.read_csv( sys.argv[2] )

w = numpy.linalg.inv(X.T @ X) @ X.T @ y

print( w )
```

Sample input file ‘X.csv’:

```
x1,x2
1,1
2,1
3,1
4,1
```

Sample input file ‘y.csv’:

```
y
3
5
7
9
```

Sample output:

```
     y
0  2.0
1  1.0
```

In other words: $y = 2x + 1$.

10.3 Fitting polynomial lines

You can use any **basis function** $\phi(\mathbf{x}')$ on input $\mathbf{x}'$ for $\mathbf{x}$, as long as weights $\mathbf{w}$ are linear.

For example: $\mathbf{x} = \phi(\mathbf{x}') = [\mathbf{x}'_{[1]}\ (\mathbf{x}'_{[1]})^2\ (\mathbf{x}'_{[1]})^3\ \mathbf{x}'_{[2]}\ (\mathbf{x}'_{[2]})^2\ (\mathbf{x}'_{[2]})^3]^\top$

In this case the weights will be coefficients for terms in a polynomial.

10.4 Convexity

Note that since the loss function is convex, there is only one minimum.

We will see other functions that don't have this property, so we'll get local optima.

10.5 Regularization

Again, we may add prior information to avoid overfitting (not zero counts, but wiggly lines).

A good choice is to add a term to the loss to penalize large weight parameters:

$$\mathsf{L}_{\text{NLG},\sigma+\text{L2},\tau}(y, \mathbf{w}^\top\mathbf{x}) = -\ln \mathcal{N}_{\mathbf{w}^\top\mathbf{x},\sigma}(y) - \ln \mathcal{N}_{0,\tau}(\mathbf{w})$$

Optimizing this (at slope zero) gives:

$$\begin{aligned}
0 &= \frac{\partial}{\partial \mathbf{w}} - \sum_{\langle y,\mathbf{x}\rangle \in \mathcal{D}} \ln \mathcal{N}_{\mathbf{w}^\top\mathbf{x},\sigma}(y) - \sum_{v=1}^{V} \ln \mathcal{N}_{0,\tau}(\mathbf{w}_{[v]}) \\
&\vdots \\
&= - \sum_{\langle y,\mathbf{x}\rangle \in \mathcal{D}} \frac{1}{2\sigma^2} \frac{\partial}{\partial \mathbf{w}} (y - \mathbf{w}^\top\mathbf{x})^2 - \sum_{v=1}^{V} \frac{1}{2\tau^2} \frac{\partial}{\partial \mathbf{w}} (\mathbf{w}_{[v]})^2 && \text{product rule} \\
&= - \sum_{\langle y,\mathbf{x}\rangle \in \mathcal{D}} \frac{1}{2\sigma^2} 2(y - \mathbf{w}^\top\mathbf{x}) \frac{\partial}{\partial \mathbf{w}} (y - \mathbf{w}^\top\mathbf{x}) - \sum_{v=1}^{V} \frac{1}{2\tau^2} 2\mathbf{w}_{[v]} \frac{\partial}{\partial \mathbf{w}} \mathbf{w}_{[v]} && \text{power rule} \\
&= - \sum_{\langle y,\mathbf{x}\rangle \in \mathcal{D}} \frac{1}{2\sigma^2} 2(y - \mathbf{w}^\top\mathbf{x})(-\mathbf{x}) - \sum_{v=1}^{V} \frac{1}{2\tau^2} 2\mathbf{w}_{[v]} && \text{power rule} \\
&= - \sum_{\langle y,\mathbf{x}\rangle \in \mathcal{D}} (y - \mathbf{w}^\top\mathbf{x})(-\mathbf{x}) - \sum_{v=1}^{V} \frac{\sigma^2}{\tau^2} \mathbf{w}_{[v]} && \text{multiply by } \sigma^2 \\
&= \mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) - \frac{\sigma^2}{\tau^2}\mathbf{I}\mathbf{w} && \text{definition of inner product} \\
&= \mathbf{X}^\top\mathbf{y} - \mathbf{X}^\top\mathbf{X}\mathbf{w} - \frac{\sigma^2}{\tau^2}\mathbf{I}\mathbf{w} && \text{distributive axiom} \\
&= \mathbf{X}^\top\mathbf{y} - (\mathbf{X}^\top\mathbf{X} + \frac{\sigma^2}{\tau^2}\mathbf{I})\mathbf{w} && \text{distributive axiom}
\end{aligned}$$

which has a single root: $\mathbf{w} = (\mathbf{X}^\top\mathbf{X} + \frac{\sigma^2}{\tau^2}\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}$, so it is still convex.

10.6 Gradient descent

Log Gaussian (quadratic) loss can be sensitive to outliers.

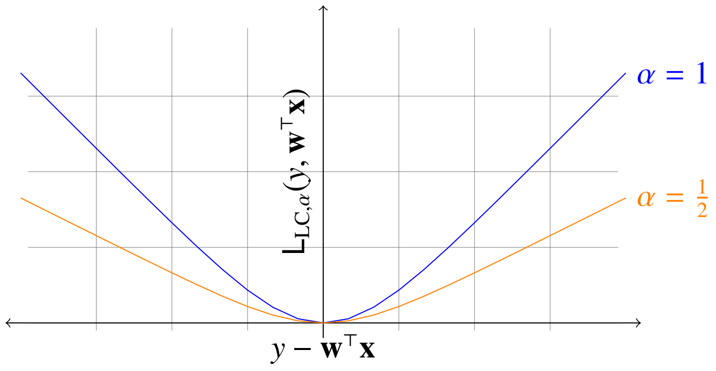
We could use absolute value loss, but it is indeterminate (there are multiple optimal solutions).

People often combine the two into log-cosh loss:

$$\mathsf{L}_{\mathrm{LC},\alpha}(y, \mathbf{w}^\top\mathbf{x}) = \alpha \ln\cosh(y - \mathbf{w}^\top\mathbf{x})$$

(where α is the distance at which you want to start ignoring outliers).

It's quadratic in the middle (so it has a single optimum) and linear at the edges (resists outliers):



The gradient (derivative for multivariable functions) is:

$$\begin{aligned}
\frac{\partial}{\partial\mathbf{w}} \sum_{\langle y,\mathbf{x}\rangle\in\mathcal{D}} \mathsf{L}_{\mathrm{LC},\alpha}(y, \mathbf{w}^\top\mathbf{x}) &= \frac{\partial}{\partial\mathbf{w}} \sum_{\langle y,\mathbf{x}\rangle\in\mathcal{D}} \alpha \ln\cosh(y - \mathbf{w}^\top\mathbf{x}) && \text{substitution} \\
&= \frac{\partial}{\partial\mathbf{w}} \alpha \sum_{\langle y,\mathbf{x}\rangle\in\mathcal{D}} \ln\cosh(y - \mathbf{w}^\top\mathbf{x}) && \text{distributive axiom} \\
&= \alpha \frac{\partial}{\partial\mathbf{w}} \sum_{\langle y,\mathbf{x}\rangle\in\mathcal{D}} \ln\cosh(y - \mathbf{w}^\top\mathbf{x}) && \text{product rule} \\
&= \alpha \sum_{\langle y,\mathbf{x}\rangle\in\mathcal{D}} \frac{\partial}{\partial\mathbf{w}} \ln\cosh(y - \mathbf{w}^\top\mathbf{x}) && \text{sum rule} \\
&= \alpha \sum_{\langle y,\mathbf{x}\rangle\in\mathcal{D}} \frac{1}{\cosh(y - \mathbf{w}^\top\mathbf{x})} \frac{\partial}{\partial\mathbf{w}} \cosh(y - \mathbf{w}^\top\mathbf{x}) && \text{derivative of log} \\
&= \alpha \sum_{\langle y,\mathbf{x}\rangle\in\mathcal{D}} \frac{\sinh(y - \mathbf{w}^\top\mathbf{x})}{\cosh(y - \mathbf{w}^\top\mathbf{x})} \frac{\partial}{\partial\mathbf{w}} y - \mathbf{w}^\top\mathbf{x} && \text{derivative of cos} \\
&= \alpha \sum_{\langle y,\mathbf{x}\rangle\in\mathcal{D}} \frac{\sinh(y - \mathbf{w}^\top\mathbf{x})}{\cosh(y - \mathbf{w}^\top\mathbf{x})} (-\mathbf{x}) && \text{product rule}
\end{aligned}$$

$$= \alpha \sum_{\langle y, \mathbf{x}\rangle \in \mathcal{D}} \tanh(y - \mathbf{w}^\top \mathbf{x})(-\mathbf{x}) \qquad \text{definition of tan}$$

$$= -\alpha\, \mathbf{X}^\top \tanh(\mathbf{y} - \mathbf{X}\mathbf{w}) \qquad \text{definition of inner product}$$

Bah, we can't find roots here (the optimization surface is non-linear), but we can use this as update!

Start with any $\mathbf{w}^{(0)}$; move in opposite direction of error times active inputs $\mathbf{x}$, to reduce loss:

$$\begin{aligned}\mathbf{w}^{(i)} &= \mathbf{w}^{(i-1)} - \frac{1}{N} - \alpha\, \mathbf{X}^\top \tanh(\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &= \mathbf{w}^{(i-1)} + \frac{1}{N}\, \alpha\, \mathbf{X}^\top \tanh(\mathbf{y} - \mathbf{X}\mathbf{w})\end{aligned}$$

This algorithm is called **gradient descent**.

10.7 Sample code

Sample code in pandas:

```
import sys
import numpy
import pandas

a = .1

X = pandas.read_csv( sys.argv[1] )
y = pandas.read_csv( sys.argv[2] )
N = len(X)

## Randomly initialize...
w = pandas.DataFrame( numpy.random.rand(len(X.columns),1), X.columns, ['y'] )

## Run 100 epochs...
for i in range(100):
  w = w + 1/N * a * X.T @ numpy.tanh( y - X @ w )

  # Print sum of loss of training data: should decrease at each epoch...
  print( numpy.sum( numpy.log( numpy.cosh( y - X @ w ) ) ) )

print( w )
```

Sample input file 'X.csv':

```
x1,x2
1,1
2,1
3,1
4,1
```

Sample input file 'y.csv' (now with an outlier at the third point):

```
y
3
5
3
9
```

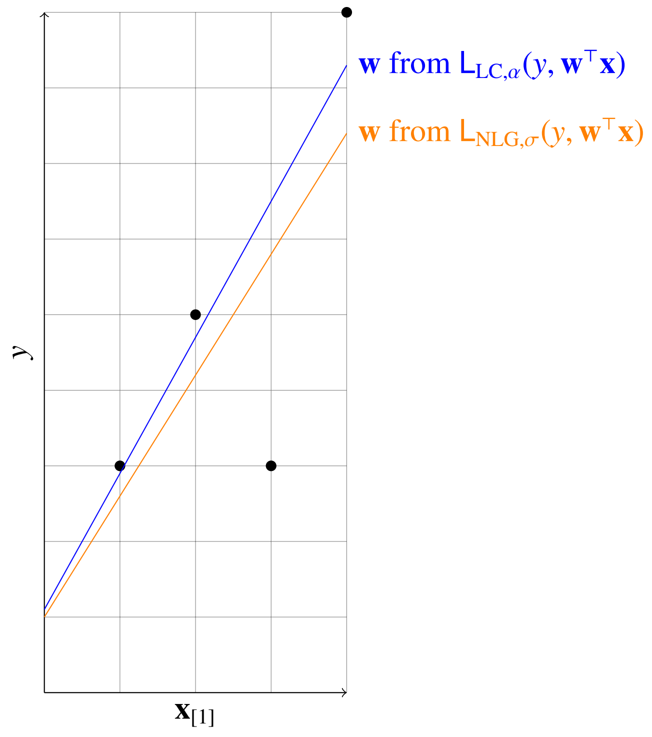
Sample output:

```
           y
x1  1.801261
x2  1.113004
```

whereas quadratic loss gives us:

```
     y
0  1.6
1  1.0
```

Graphically, we can see log cosh was less sensitive to the outlier:


$\mathbf{w}$ from $\mathsf{L}_{\mathrm{LC},\alpha}(y, \mathbf{w}^\top \mathbf{x})$
$\mathbf{w}$ from $\mathsf{L}_{\mathrm{NLG},\sigma}(y, \mathbf{w}^\top \mathbf{x})$
y
$\mathbf{x}_{[1]}$