Chris Piech
CS109

Section #8
Dec 1, 2017

Section #8: Machine Learning Soln

1. Vision Test:

We are going to solve this problem by finding the MLE estimate of θ. To find the MLE estimate, we are going to find the argmax of the log likelihood function. To calculate argmax we are going to use gradient ascent, which requires that we know the partial derivative of the log likelihood function with respect to theta.

First write the log likelihood

$$L(\theta) = \prod_{i=1}^{20} p^{y^{(i)}}(1-p)^{[1-y^{(i)}]}$$

$$LL(\theta) = \sum_{i=1}^{20} y^{(i)} \log(p) + (1-y^{(i)}) \log(1-p)$$

Then, find the derivative of log likelihood with respect to θ. We first do this for one data point:

$$\frac{\partial LL}{\partial \theta} = \frac{\partial LL}{\partial p} \cdot \frac{\partial p}{\partial \theta}$$

We can calculate both the smaller partial derivatives independently:

$$\frac{\partial LL}{\partial p} = \frac{y^{(i)}}{p} - \frac{1-y^{(i)}}{1-p}$$

$$\frac{\partial p}{\partial \theta} = p[1-p]$$

Putting it all together for one letter:

$$\begin{aligned}
\frac{\partial LL}{\partial \theta} &= \frac{\partial LL}{\partial p} \cdot \frac{\partial p}{\partial \theta} \\
&= \Big[\frac{y^{(i)}}{p} - \frac{1-y^{(i)}}{1-p}\Big] p[1-p] \\
&= y^{(i)}(1-p) - p(1-y^{(i)}) \\
&= y^{(i)} - p \\
&= y^{(i)} - \sigma(\theta - f)
\end{aligned}$$

For all twenty examples:

$$\frac{\partial LL}{\partial \theta} = \sum_{i=1}^{20} y^{(i)} - \sigma(\theta + f^{(i)})$$

2. Run a Tensor Flow Algorithm

See the FLoydHub solutions in section!

3. Deep Dream

a. Loop over all images, run them through your deep learning network, select the one for which the activation of the final neuron is the largest.

b. Note that: $h_2 = \sigma(\sum_{i=0}^{m_x} \theta_{i,2}\mathbf{x}_i)$. We are going to use gradient ascent to chose pixels ($\mathbf{x}$) that maximize the activation of the neuron (h_2)!

$$\arg\max_{\mathbf{x}} h_2 = \arg\max_{\mathbf{x}} \sigma\Big(\sum_{i=0}^{m_x} \theta_{i,2}\mathbf{x}_i\Big)$$

Which requires us to solve for the derivative of h_2 with respect to each pixel $\mathbf{x}_i$.

$$\begin{aligned}
\frac{\partial h_2}{\partial x_i} &= \frac{\partial \sigma(z)}{\partial z} \cdot \frac{\partial z}{\partial x_i} && \text{chain rule where } z = \sum_{k=0}^{m_x} \theta_{k,2}\mathbf{x}_k \\
&= \sigma(z)[1-\sigma(z)] \cdot \frac{\partial z}{\partial x_i} && \text{the derivative of the sigmoid function} \\
&= h_2[1-h_2] \cdot \frac{\partial z}{\partial x_i} && \text{since } h_2 = \sigma(z) \\
&= h_2[1-h_2] \cdot \theta_i && \text{That's all folks!}
\end{aligned}$$

c. Same as the previous part, but start with pixels set to the input picture. Optimize for the output of $Y = 1$, the cat neuron, and only run a few iterations of gradient descent.

4. Multiclass Bayes: First make the naive bayes assumption and simplify argmax of $P(Y|\mathbf{X})$

$$\begin{aligned}
\hat{Y} &= \arg\max_y \frac{P(X_1 = 1, X_2 = 1, X_3 = 0|Y = y)P(Y = y)}{P(X_1 = 1, X_2 = 1, X_3 = 0)} \\
&= \arg\max_y P(X_1 = 1, X_2 = 1, X_3 = 0|Y = y)P(Y = y) \\
&= \arg\max_y P(X_1 = 1|Y = y)P(X_2 = 1|Y = y)P(X_3 = 0|Y = y)P(Y = y)
\end{aligned}$$

$$\begin{aligned}
P(X_1 = 1, Y = y) &= [\mathbf{count}(X_1 = 1, Y = y) + 1]/\mathbf{count}(Y = y) + 2 \\
P(X_2 = 1, Y = y) &= [\mathbf{count}(X_2 = 1, Y = y) + 1]/\mathbf{count}(Y = y) + 2 \\
P(X_3 = 1, Y = y) &= [\mathbf{count}(X_3 = 1, Y = y) + 1]/\mathbf{count}(Y = y) + 2 \\
P(X_1 = 0, Y = y) &= [\mathbf{count}(X_1 = 0, Y = y) + 1]/\mathbf{count}(Y = y) + 2 \\
P(X_2 = 0, Y = y) &= [\mathbf{count}(X_2 = 0, Y = y) + 1]/\mathbf{count}(Y = y) + 2 \\
P(X_3 = 0, Y = y) &= [\mathbf{count}(X_3 = 0, Y = y) + 1]/\mathbf{count}(Y = y) + 2
\end{aligned}$$

You don't need to use MAP to estimate Y:

$$P(Y = y) = \mathbf{count}(Y = y)/10,000$$