Lecture 7

March 10, 2016

1 Expectation Maximization

Maximize:

$$\int p\ln(p) + \lambda_1(\int px - \mu) + \lambda_2(\int p(x-\mu)^2 - \sigma^2) \tag{1}$$

$$\ln(p) + p\frac{1}{p} + \lambda_1 x + \lambda_2(x-\mu)^2 = 0 \tag{2}$$

$$p = e^{Q(x)} \tag{3}$$

2 Perception Learning

Let there be two classes: $P = \vec{p}_1, \vec{p}_2, \ldots, \vec{p}_n$, $N = \vec{n}_1, \vec{n}_2, \ldots, \vec{n}_n$Compute a linear regression and add everything together and make a decision on the class. The goal is to find the vector that will produce a positive dot product when multiplying one class and a negative dot product when multiplying the other class. The positive space is known as the *positive half-space* and the negative space is known as the *negative half-space*.

Perception Learning:

1. **start**: $\vec{\beta}_0 = randomvector$
2. **test** Pick $\vec{x} \in P \cup N$ randomly
3. **Case P**: $\vec{x} \in P$, if $\vec{\beta}_i^T \vec{x} \geq 0$ then goto *test*
4. **otherwise** $\vec{\beta}_{i+1} = \vec{\beta}_i + \vec{x}, i++$ goto *test*
5. **Case N**: if $\vec{x} \in N$ if $\vec{\beta}^T \vec{x} < 0$ then goto *test*
6. **otherwise** $\vec{\beta}_{i+1} = \vec{\beta}_i - \vec{x}, i++$ goto *test*

The algorithm updates $\vec{\beta}_i$ a finite number of times. It is a self adjusting algorithm that as the β gets close to the solution it will get larger and the contribution of each addition will effect the vector less.

2.1 Pocket Algorithm

Perception leanrning, count number of errors for each $\vec{\beta}_i$. The picket contains the best $\vec{\beta}_i$ so far. You exchange the it with the next best each time. You iterate.

2.2 Exact Solution

Linear programming. Will either find a solution or tell you when there is no solution.

notes: rojas "neural networks" "perception", "why the gauussian dsitribution?", "bias variance dilemma", "Logistic regression".

3 Lgistic Regression

Logistic regression fins a compromise between the perfect solution and what that deals with errors. This is an improvement upon *perception learning*.

$$P(\vec{x}, \vec{\beta}) = \frac{e^{\vec{\beta}^T \vec{x}}}{1 + e^{\vec{\beta}^T \vec{x}}} \tag{4}$$

Regression:

$$\vec{\beta}^T \vec{x} = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n \tag{5}$$

Positive:

$$P(\vec{x}, \vec{beta}) = \frac{e^{\vec{\beta}^T \vec{x}}}{1 + e^{\vec{\beta}^T \vec{x}}} \tag{6}$$

Negative:

$$P(\vec{x}, \vec{beta}) = \frac{1}{1 + e^{\vec{\beta}^T \vec{x}}} \tag{7}$$

Likelihood:

$$L(\vec{\beta}) = \prod^{N_1} P(\vec{x_i}, \vec{\beta}) \prod^{N_2} (1 - P(\vec{x_i}, \vec{\beta})) \tag{8}$$

Maximum likelihood will maximize the equation 8 by finding a β. maximizing products is hard, but if you take the *log* of the probabilities then you can deal with additions instead of products. As well *log* is a monotonically increasing function that allows you to find the maximum easier.

$$l(\vec{\beta}) = \sum^{N_1} log(P(\vec{x_i}, \vec{\beta})) + \sum^{N_2} log(1 - P(\vec{x_i}, \vec{\beta})) \tag{9}$$

$$l(\vec{\beta}) = \sum^{N_1} \vec{\beta}^T \vec{x} - log(1 + e^{\vec{\beta}^T \vec{x}}) + \sum^{N_2} log(1 + e^{\vec{\beta}^T \vec{x}})) \tag{10}$$

Maximize:

$$l(\vec{\beta}) = \sum^{N} y_i \vec{\beta}^T \vec{x} - log(1+e^{\vec{\beta}^T \vec{x}}) + \sum^{N} log(1+e^{\vec{\beta}^T \vec{x}})),\ y_i = 1 \text{ for Pos } \& \ y_i = 0 \text{ for Neg} \tag{11}$$

Use gradient ascent to maximize equation 11.

$$\frac{dl(\vec{\beta})}{d\vec{\beta}} = \sum^{N} y_i \vec{x_i} - \sum^{N} \frac{e^{\vec{\beta^T} \vec{x_i}} \vec{x_i}}{1 + e^{\vec{\beta^T} \vec{x_i}}} \tag{12}$$

$$\nabla l(\vec{\beta}) = \sum^{N} \vec{x_i}(y_i - P(\vec{x_i}, \vec{\beta})) \tag{13}$$

Gradient Ascent:

$$\vec{\beta_{new}} = \vec{\beta_{old}} + \alpha \nabla l(\vec{\beta}) \tag{14}$$

4 Homework

Recover 10 clusters using the EM-Algorithm for gaussians.