CSC320 week 3 tutorial note

Zhaowei Liu

January 25, 2017

Recall: solution to a (probably overdetermined) linear system: for a system of linear functions $H\boldsymbol{v} = \boldsymbol{y}, H \in \mathbb{R}^{m\times n}, \boldsymbol{v} \in \mathbb{R}^n, \boldsymbol{y} \in \mathbb{R}^m$, we want to find a solution for $\boldsymbol{v}$ such that $||H\boldsymbol{v} - \boldsymbol{y}||^2$ is minimized. The solution is

$$\boldsymbol{v} = (H^T H)^{-1} H^T \boldsymbol{y} = V\Sigma^{-1}U^T\boldsymbol{y}, \text{ where } H = U\Sigma V^T \text{ by SVD}$$

Now let's take a different view at the problem. To make things simple, Assume $n = 2$, and $\boldsymbol{v} = [a, b]^T$. Then the problem has some geometrical meaning, which is to fit a line onto a bunch of observed points $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$ in 2D space, where

$$H = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_m & 1 \end{bmatrix}, \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

Clearly, every row of coefficient matrix H represents parameters in one experiment (or the x-coordinate of one data point on the 2D plain), and the corresponding entry in $\boldsymbol{y}$ is an observation. Generally, observed values contain noise. So we represent the observed values as $\boldsymbol{y}+\boldsymbol{e}$, where $\boldsymbol{y}$ are the true values and $\boldsymbol{e}$ is a $m \times 1$ vector. Then the linear system becomes $H\boldsymbol{v} = \boldsymbol{y}+\boldsymbol{e}$. Let's assume that entries in $\boldsymbol{e}$ follow i.i.d. zero-centered Gaussian distributions, i.e. $\boldsymbol{e}_i \sim \mathcal{N}(0, \sigma^2)$ for every i. The probability distribution function of $\boldsymbol{e}_i$ is

$$f_i = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{\boldsymbol{e}_i^2}{\sigma^2}\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(ax_i + b - y_i)^2}{\sigma^2}\right)$$

Using maximum likelihood estimation, the optimized values (a^*, b^*) for unknowns $\boldsymbol{v}$ is

$$\begin{aligned}
(a^*, b^*) &= \underset{a,b}{\operatorname{argmax}} \prod_{i=1}^{m} f_i \\
&= \underset{a,b}{\operatorname{argmax}} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(ax_i + b - y_i)^2}{\sigma^2}\right) \\
&= \underset{a,b}{\operatorname{argmax}} \log\left(\prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(ax_i + b - y_i)^2}{\sigma^2}\right)\right) \\
&= \underset{a,b}{\operatorname{argmax}} \sum_{i=1}^{m} \left(\log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2}\frac{(ax_i + b - y_i)^2}{\sigma^2}\right) \\
&= \underset{a,b}{\operatorname{argmin}} \sum_{i=1}^{m} \frac{1}{2}\frac{(ax_i + b - y_i)^2}{\sigma^2} \\
&= \underset{a,b}{\operatorname{argmin}} \sum_{i=1}^{m} (ax_i + b - y_i)^2
\end{aligned}$$

The result points to the same solution using pseudo-inverse, since we were trying to minimize $||H\boldsymbol{v} - \boldsymbol{y}||^2$ in the derivation for pseudo-inverse solution. However, the solution might be different if we use different noise models (e.g. uniform distribution).

So far, we have built the connection between solving system of linear equations and fitting a line onto data (although we focused on fitting a straight line in 2D space, you should be able to extend the derivation and solution to fitting a hyperplane in n-dimentional space). However, the relation between all x_i's and y_i's may not be as simple as linear. For example, the underlying relation might be a quadratic function $y_i \approx a_0 + a_1 x_i + a_2 x_i^2$. But with similar techniques, we can transform such problem to another linear system, where unknowns are $[a_0, a_1, a_2]$.

$$H \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_m & x_m^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \boldsymbol{y}$$

And the solution is

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = (H^T H)^{-1} H^T \boldsymbol{y}$$

We can simply extend such algorithm to fit data with any degree of polynomial. Assume we want to fit data with a polynomial with degree N. Then we can build the system as

$$H \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^N \\ 1 & x_2 & x_2^2 & \dots & x_2^N \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_m & x_m^2 & \dots & x_m^N \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \boldsymbol{y}$$

with solution

$$\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} = (H^T H)^{-1} H^T \boldsymbol{y}$$

There is a problem for such algorithm. If some x_i is large, x_i^N can be very large. So the difference between max value and min value in the matrix H can be huge. Such difference can result in large numerical error or instability in calculation, even if the algorithm is mathematically correct.

Consider a different way to understand polynomial fitting. Assume all data points are sampled (with noise) from a continuous function I, so that $y_i \approx I(x_i)$ for all i. We can use Taylor's expension to approximate the function I:

$$I(x) = \sum_{k=0}^{\infty} \frac{I^{(k)}(c)}{k!}(x-c)^k, c \text{ is any constant}$$

Let f be a polynomial function which is an approximation of I. We can let f just be the first $N+1$ terms of Taylor's expension.

$$f(x) = \sum_{k=0}^{N} \frac{I^{(k)}(c)}{k!}(x-c)^k$$

The only unknowns in f are the derivatives of I. Similar as before, we can construct a linear system to solve the derivatives.

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} I(x_1) \\ I(x_2) \\ \vdots \\ I(x_m) \end{bmatrix} = \begin{bmatrix} \frac{(x_1-c)^0}{0!} & \frac{(x_1-c)^1}{1!} & \cdots & \frac{(x_1-c)^N}{N!} \\ \frac{(x_2-c)^0}{0!} & \frac{(x_2-c)^1}{1!} & \cdots & \frac{(x_2-c)^N}{N!} \\ \vdots & \vdots & & \vdots \\ \frac{(x_m-c)^0}{0!} & \frac{(x_m-c)^1}{1!} & \cdots & \frac{(x_m-c)^N}{N!} \end{bmatrix} \begin{bmatrix} I^{(0)}(c) \\ I^{(1)}(c) \\ \vdots \\ I^{(N)}(c) \end{bmatrix} = H \begin{bmatrix} I^{(0)}(c) \\ I^{(1)}(c) \\ \vdots \\ I^{(N)}(c) \end{bmatrix}$$

Then the solution will be

$$\begin{bmatrix} I^{(0)}(c) \\ I^{(1)}(c) \\ \vdots \\ I^{(N)}(c) \end{bmatrix} = (H^T H)^{-1} H^T \boldsymbol{y}$$

After solving the derivatives, we can put them back to f to get an estimate of I. Note that in matrix H, higher order terms will be "normalized" by larger factorials. So the algorithm is more stable numerically.

However, we don't use high order polynomial to fit data in general. Because high order polynomial tends to "overfit" the data, and hence cannot reflect general rules between the data points.

Sometimes we believe some data points are more important than others. Thus we want to fit the important data better (i.e. to let those data points have less error than others) And we can use numbers to represent the importance where larger numbers mean greater importance. Assume point (x_i, y_i) has weight (or importance) $w_i \geq 0$. Then, instead of minimizing $\sum_{i=1}^{m}(y_i - f(x_i))^2$, we will minimize $\sum_{i=1}^{m}(w_i(y_i - f(x_i)))^2$, or in matrix form $||W(\boldsymbol{y} - H\boldsymbol{v})||^2$, where W is a diagonal matrix, $W = diag(w_1, w_2, \ldots, w_m)$. The solution for the unknowns is

$$d = (H^T W^2 H)^{-1} H^T W^2 \boldsymbol{y}$$