CS 357 - 17 Least Square Fitting

Boyang Li (boyangl3)

Linear regression (all-in-one): Given m data points in the form (t_i, y_i), we want to find a straight line that best fits the data. We are finding x_0 and x_1 such that $y_i = x_0 + x_1 t_i$. In matrix form, the linear system in the form $\mathbf{Ax} = \mathbf{b}$ is:

$$\begin{bmatrix} 1 & t_1 \\ 1 & t_2 \\ \vdots & \vdots \\ 1 & t_m \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

Note: $\mathbf{b}$ is a linear combination of columns of $\mathbf{A}$. $\mathbf{b} \in \text{range}(\mathbf{A})$.

This is an overdetermined system, since the number of equations is greater then the number of unknowns. So we can write the system in the form $\mathbf{A}x \cong \mathbf{b}$.

We will find $\mathbf{x}$ that minimizes the Euclidean norm of the residual vector $\mathbf{r} = \mathbf{b} - \mathbf{Ax}$, which is $\min_{\mathbf{x}} \|\mathbf{r}\|_2^2 = \min_{\mathbf{x}} \|\mathbf{b} - \mathbf{Ax}\|_2^2$. As long as $m \geq n$, (n is the number of prediction variables), the system always has a solution, the solution is unique if $\text{rank}(\mathbf{A}) = n$.

Suppose we have a quadratic model, $y = x_0 + x_1 t + x_2 t^2$, now the system would be:

$$\begin{bmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ \vdots & \vdots & \vdots \\ 1 & t_m & t_m^2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

```python
import numpy as np
import numpy.linalg as la
A = np.vstack([np.ones(m), t, t_sq]).T
x = la.lstsq(A, b)[0]
x_0, x_1, x_2 = x
```

Normal Equation: Given the origional system $\mathbf{Ax} = \mathbf{b}$, we would change it to the normal equation system $\mathbf{Mx} = \mathbf{n}$, where $\mathbf{M} = \mathbf{A}^T\mathbf{A}$ and $\mathbf{n} = \mathbf{A}^T\mathbf{b}$. The time complexity of the Normal Equations is $\mathcal{O}(mn^2)$.

Solving least square using SVD: We can use SVD decomposition $\mathbf{A} = \mathbf{U\Sigma V}^T$, and applied it to the system $\mathbf{Ax} = \mathbf{b}$, then $\mathbf{x} = \mathbf{V\Sigma}^+\mathbf{U}^T\mathbf{b}$. The cost to compute given SVD is $\mathcal{O}(mn)$

```python
import numpy as np
import numpy.linalg as la
U, S, VT = la.svd(A)
x = VT.T @ (U.T @ b / S)
```

Link to course textbook for more detailed information.