EECS 545 → Machine Learning

Lecture 2 → Linear Regression (Part 1)

- Supervised Learning → given data X in feature space and corresponding labels Y, learn to predict Y from X
- Classification → discrete-valued labels
- Regression → continuous-valued labels

- Notation → $x \in \mathbb{R}^D$ = data (scalar or vector)

  $\phi_j(x) \in \mathbb{R}$ = j-th feature for x (scalar), $j = 0, \dots, M-1$

  $\phi(x) \in \mathbb{R}^M$: features for x (vector)

  $y \in \mathbb{R}$ = continuous-valued label

  $x^{(n)}$ = n-th training example

  $y^{(n)}$ = n-th training label

- We want to learn a function $h(x,w) \approx y$ to predict future values
- 0-th order polynomial → $h(x,w) = w_0$
- 1-st order polynomial → $h(x,w) = w_0 + w_1 x$
- 3-rd order polynomial → $h(x,w) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$

} 1-Dimensional features

- General case → $h(x,w) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(x)$
- $h(x,w)$ is linear in parameters w
- For simplicity, we convert w_0 to a bias term

$$h(x,w) = \sum_{j=0}^{M-1} w_j \phi_j(x) = w^T \phi(x)$$

$$w = (w_0, \dots, w_{M-1})^T$$
$$\phi(x) = (\phi_0(x), \dots, \phi_{M-1}(x))^T$$
$$\phi_0(x) = 1$$

Error Functions

- Sum of squares → $E(w) = \frac{1}{2}\sum_{n=1}^{N} \{ \underbrace{h(x^{(n)}, w)}_{\text{prediction}} - \underbrace{y^{(n)}}_{\text{target}} \}^2$

- Want to find w that minimizes $E(w)$ over the training data

- Gradient of SoS → $\frac{\partial E(w)}{\partial w_k} = \frac{\partial}{\partial w_k} \frac{1}{2} \sum_{n=1}^{N} \left(\sum_{j=0}^{M-1} w_j \phi_j(x^{(n)}) - y^{(n)} \right)^2$

$$= \sum_{n=1}^{N} \left[\left(\sum_{j=0}^{M-1} w_j \phi_j(x^{(n)}) - y^{(n)} \right) \frac{\partial}{\partial w_k} \left(\sum_{j=0}^{M-1} w_j \phi_j(x^{(n)}) - y^{(n)} \right) \right]$$

$$= \sum_{n=1}^{N} \left(\underbrace{\sum_{j=0}^{M-1} w_j \phi_j(x^{(n)}) - y^{(n)}}_{\text{error}} \right) \phi_k(x^{(n)})$$

- Batch Gradient Descent → Repeat until convergence,

$$w := w - \alpha \nabla_w E(w)$$

$$\nabla_w E(w) = \sum_{n=1}^{N} \left(\sum_{j=0}^{M-1} w_j \phi_j(x^{(n)}) - y^{(n)} \right) \phi(x^{(n)})$$
$$= \sum_{n=1}^{N} \left(w^T \phi(x^{(n)}) - y^{(n)} \right) \phi(x^{(n)})$$

- Stochastic Gradient Descent → instead of computing batch gradient descent which is over the entire dataset, compute gradient for individual examples and update
- Repeat until convergence,

$$\omega := \omega - \alpha \nabla_{\omega} E(\omega | x^{(n)}) \qquad \nabla E(\omega | x^{(n)}) = \sum_{j=0}^{M-1} \omega_j \phi_j (x^{(n)}) - y^{(n)}) \phi(x^{(n)})$$
$$= \omega^T \phi(x^{(n)}) - y^{(n)}) \phi(x^{(n)})$$

- In SGD, iterate over entire dataset with multiple epochs until convergence

<u>Closed form solution</u>

- Compute gradient, then set equal to 0
- Solve the equation in a closed form

$$E(\omega) = \frac{1}{2} \sum_{n=1}^{N} \left(\sum_{j=0}^{M-1} \omega_j \phi_j (x^{(n)}) - y^{(n)} \right)^2$$

$$= \frac{1}{2} \sum_{n=1}^{N} (\omega^T \phi(x^{(n)}) - y^{(n)})^2$$

$$= \frac{1}{2} \sum_{n=1}^{N} (\omega^T \phi(x^{(n)}))^2 - \sum_{n=1}^{N} y^{(n)} \omega^T \phi(x^{(n)}) + \frac{1}{2} \sum_{n=1}^{N} (y^{(n)})^2$$

$$= \frac{1}{2} \omega^T \Phi^T \Phi \omega - \omega^T \Phi^T y + \frac{1}{2} y^T y$$

where we define Φ to be the design matrix, $N \times M$ containing the M basis functions (columns) and N data points (rows)

$$\Phi = \begin{pmatrix} \phi_0(x^{(1)}) & \phi_1(x^{(1)}) & \dots & \phi_{M-1}(x^{(1)}) \\ \phi_0(x^{(2)}) & \phi_1(x^{(2)}) & \dots & \phi_{M-1}(x^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x^{(N)}) & \phi_1(x^{(N)}) & \dots & \phi_{M-1}(x^{(N)}) \end{pmatrix} \qquad \Phi \omega \approx y$$

- We now need to compute the derivative in matrix form,

$$\nabla_{\omega} E(\omega) = \nabla_{\omega} \left(\frac{1}{2} \omega^T \Phi^T \Phi \omega - \omega^T \Phi^T y + \frac{1}{2} y^T y \right)$$

$$= \Phi^T \Phi \omega - \Phi^T y \qquad \text{(set equal to 0)}$$

- Solving this equation gives us,

$$\Phi^T \Phi \omega = \Phi y$$

$$\omega_{ML} = (\Phi^T \Phi)^{-1} \Phi^T y$$

- We call this the Moore-Penrose pseudo inverse : $\Phi^T = (\Phi^T \Phi)^{-1} \Phi^T$ applied to $\Phi \omega \approx y$

Aside → Calculating Matrix Gradients

- Suppose $f: \mathbb{R}^{m \times n} \to \mathbb{R}$ is a function that takes an input matrix A of size $m \times n$ and returns a real value
- The gradient of f w.r.t. $A \in \mathbb{R}^{m \times n}$ is,

$$\nabla_A f(A) \in \mathbb{R}^{m \times n} = \begin{bmatrix} \frac{\partial f(A)}{\partial A_{11}} & \frac{\partial f(A)}{\partial A_{12}} & \cdots & \frac{\partial f(A)}{\partial A_{1n}} \\ \frac{\partial f(A)}{\partial A_{21}} & \frac{\partial f(A)}{\partial A_{22}} & \cdot & \frac{\partial f(A)}{\partial A_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f(A)}{\partial A_{m1}} & \frac{\partial f(A)}{\partial A_{m2}} & \cdots & \frac{\partial f(A)}{\partial A_{mn}} \end{bmatrix} \qquad (\nabla_A f(A))_{ij} = \frac{\partial f(A)}{\partial A_{ij}}$$

- E.g. if A is just a vector, $x \in \mathbb{R}^n$, then $\nabla_x f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix}$

- $\nabla_x (f(x) + g(x)) = \nabla_x f(x) + \nabla_x g(x)$
- For $t \in \mathbb{R}$, $\nabla_x (t\, f(x)) = t\, \nabla_x f(x)$

- Linear function → $f(x) = \sum_{i=1}^{n} b_i x_i = b^T x$
- Gradient → $\frac{\partial f(x)}{\partial x_k} = \frac{\partial}{\partial x_k} \sum_{i=1}^{n} b_i x_i = b_k$
- Compact form → $\nabla_x f(x) = b$

- Quadratic function → $f(x) = \sum_{i,j=1}^{n} x_i A_{ij} x_j = x^T A x$
- Gradient → $\frac{\partial f(x)}{\partial x_k} = 2 \sum_{j=1}^{n} A_{kj} x_j = 2(Ax)_k$
- Compact form → $\nabla_x f(x) = 2Ax$