EECS 545 → Machine Learning

Lecture 4 → Classification

- Supervised learning → learn to predict Y from X given data X in feature space and labels Y
- In classification, these labels are discrete-valued
- Classification → given an input vector x, assign it to one of K distinct classes C_k where $k=1,2,\dots,K$

- If $K=2$,
  - $y=1$ means x is in C_1
  - $y=0$ means x is in C_2
- If $K>2$,
  - use 1-of-K coding
  - $y=(0,1,0,0)^T$ means x is in C_2

- 0-1 loss → Classification error $= \frac{1}{N}\sum_{j=1}^{N} 1[h(x_{test}^{(j)} \neq y_{test}^{(j)}]$

Logistic Regression

- Decision boundary is modeled as a function of the input x
- Learn $P(C_k|x)$ over the data
- Directly predict class labels from inputs
- Logistic regression models the class probability (posterior) using a sigmoid function applied to a linear function of the feature vectors

$$p(C_1|\phi)=h(\phi)=\sigma(w^T\phi(x))$$

- Logistic sigmoid function → $\sigma(a) = \dfrac{1}{1+\exp(-a)}$
- Inverse of logistic sigmoid is logit function → $a = \ln\left(\dfrac{\sigma}{1-\sigma}\right)$ a.k.a. odds ratio
- Generalization is softmax → $p_i = \dfrac{\exp(q_i)}{\sum_j \exp(q_j)}$
- We can define the likelihood of x as,

$$P(y=1|x,w)=\sigma(w^T\phi(x))$$
$$P(y=0|x,w)=1-\sigma(w^T\phi(x))$$

$$P(y|x|w)=\sigma(w^T\phi(x))^y(1-\sigma(w^T\phi(x)))^{1-y}$$

- The likelihood function is $p(y|w)=\prod_{n=1}^{N}(h^{(n)})^{y^{(n)}}(1-h^{(n)})^{1-y^{(n)}}$

$$\text{where } h^{(n)}=p(C_1|\phi(x^{(n)}))=\sigma(w^T\phi(x^{(n)}))$$

- Define our loss function as $E[w]=-\log p(y|w)$, minimizing $E[w]$ will maximize the likelihood

MLS Derivation

$$\nabla_w \log P(y|x^{(1)},\dots,x^{(N)},w)$$

$$= \sum_{n=1}^{N} \nabla_w \left(y^{(n)} \log h(x^{(n)},w) + (1-y^{(n)}) \log(1-h(x^{(n)},w))\right)$$

$$= \sum^{N} \nabla_w\, y^{(n)} \left(\frac{\sigma^{(n)}(1-\sigma^{(n)})}{\sigma^{(n)}} \right) - (1-y^{(n)}) \frac{\sigma^{(n)}(1-\sigma^{(n)})}{1-\sigma^{(n)}} \Bigg) \nabla_w (w^T \phi(x^{(n)}))$$

$$= \sum_{n=1}^{N} (y^{(n)}(1-\sigma^{(n)}) - (1-y^{(n)})\sigma^{(n)}) \nabla_w (w^T \phi(x^{(n)}))$$

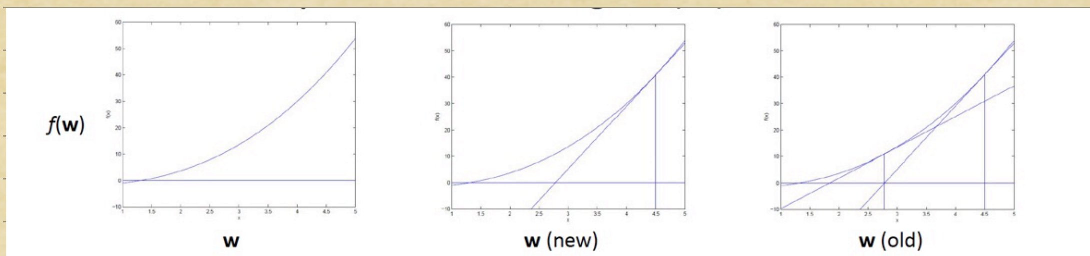
$$= \sum_{n=1}^{N} (y^{(n)} - \sigma^{(n)}) \phi(x^{(n)}))$$

- So we have $\nabla E[w] = \sum_{n=1}^{N} (h^{(n)} - y^{(n)}) \phi(x^{(n)})$, where $h^{(n)} = p(C_1|\phi(x^{(n)}) = \sigma(w^T \phi(x^{(n)}))$
- This is very similar to the gradient descent expression from linear regression, with the addition of the sigmoid function $\sigma(a)$

Newton's Method

- Goal is to minimize a general function $E[w]$
- Solve for $f(w) = \frac{\partial E[w]}{\partial w} = 0$
- We do so by repeating the following until convergence,

$$w := w - \frac{f(w) \rightarrow \text{current value}}{f'(w) \rightarrow \text{slope}}$$



- In the multivariate case, we have $w := w - H^{-1} \nabla_w E$, where H is the Hessian matrix evaluated as $H_{ij}(w) = \frac{\partial^2 E[w]}{\partial w_i \partial w_j}$

- LR closed form solution $\rightarrow w_{ML} = (\Phi^T \Phi)^{-1} \Phi^T y$
- WLR closed form solution $\rightarrow w_{WLS} = (\Phi^T R \Phi)^{-1} \Phi^T R y$ $\quad$ R is an $N \times N$ diagonal weight matrix
- In logistic regression, $h(x,w)$ is non linear and there is no closed form solution
  - we can only estimate by repeatedly applying Newton steps

- Iteratively applying NR involves least-squares with weights matrix R, $R_{nn} = h^{(n)}(1-h^{(n)})$
- R depends on w, w depends on R, so we have iterative reweighted least squares (IRLS)

$$w^{(new)} = (\Phi^T R \Phi)^{-1} \Phi^T R z, \quad z = \Phi w^{(old)} - R^{-1}(h-y)$$

K-Nearest Neighbor Classification

- Given a test example x, find the k training examples that are closest to x
- Predict as a label for the testing sample the most frequent class among all y's from KNN(x)

$$KNN(x) = \{(x^{(1)'}, y^{(1)'}), (x^{(2)'}, y^{(2)'}), \dots, (x^{(k)'}, y^{(k)'})\}$$

$$h(x) = \arg\max_{y} \sum_{(x', y') \in KNN(x)} 1[y' = y] \qquad \text{majority vote}$$

- Larger k leads to smoother decision boundary
- Classification generally improves as N increases
- As $N \to \infty$, error rate of 1-NN is never more than twice the optimal error, which is obtained from true conditional class dist^s

- Hyperparameters → ① Distance metric → $D(x, x')$
  ② Value of k

- Can turn KNN into a regression problem by assigning the label of the testing example as the average of the the nearest target labels,

$$h(x) = \frac{1}{k} \sum_{(x', y') \in KNN(x)} y'$$

- KNN advantages → simple + flexible
  effective for low dimensional outputs
- KNN disadvantages → expensive as we have to compute + store distances for the whole training data for every prediction
  all points are far away in high dimensions
  not robust to irrelevant features/outliers