KNN

- Hyperparameters: K, L_p-norm
- $K \uparrow$ Overfitting $\downarrow$
- Classification and regression
- Non-linear decision boundary

Overfitting Prevention

- Less expressive models
- Regularization
- Noise in training data
- Stop optimization early

Train/Dev/Test

- Train + Dev > Test
- Train > Test
- Never train on Test
- Use N-fold cross validation to tune

Curse of Dimensionality

- Points in high-dim. spaces are far from O
- Preprocess data to have $\mu = 0$ and unit σ

Linear Models

- Infinite hypothesis space
- Limited to linearly-separable data

Decision Trees

- Non-linear decision boundary
- Learned by ID3
- Use Entropy-Information Gain to split attr.
- Classification and regression
- Sub-optimal
- If-then-else statements

Perceptron

- Only updates on mistake
- Only linearly separable data
- All separating hyperplanes are equal
- Converges in less than $\left(\frac{R}{\gamma}\right)^2$ mistakes
- Works in adversarial distributions
- Hyperparameters: epoch (T) and learning rate (η)

Perceptron Algorithm

Given a training set $\mathcal{D} = \{(\mathbf{x}, y)\}$
Initialize $\mathbf{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$
For epoch $1 \dots T$:
  For $(\mathbf{x}, y)$ in $\mathcal{D}$:
    If $y(\mathbf{w}^T\mathbf{x}) \leq 0$:
      $\mathbf{w} \leftarrow \mathbf{w} + \eta y \mathbf{x}$
Return $\mathbf{w}$

Perceptron Update

- $\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$
- $b \leftarrow b + y$
- $\mathbf{w}_{t+1}^T\mathbf{x} = (\mathbf{w}_t + \mathbf{x})^T\mathbf{x} = \mathbf{w}^T\mathbf{x} + \mathbf{x}^T\mathbf{x}$

Sigmoid Function

- $\sigma(z) = \frac{\exp(z)}{1+\exp(z)}$
- Fits z to the range of probability

Logistic Regression

- Generates a probability, not a value
- Linear decision boundary around $P(y = 1 \mid \mathbf{x})$

Gradient Descent

- Computes the gradient over each point in the dataset
- Finds the global optimum if the function is convex
- Hyperparameter: step size (η)
- Small step sizes are slow, large ones are unstable
- SGD computes the gradient for randomly sampled points

Logistic Regression MLE

$$\begin{aligned}
\arg\max_{\mathbf{w},b} P(S; \mathbf{w}, b) &= \arg\max_{\mathbf{w},b} \prod_{i=1}^{m} P(y_i \mid x_i; \mathbf{w}, b) \\
&= \arg\max_{\mathbf{w},b} \sum_{i=1}^{m} \log P(y_i \mid x_i; \mathbf{w}, b) \\
&= \arg\max_{\mathbf{w},b} \sum_{i=1}^{m} \log \sigma(y_i(\mathbf{w}^T\mathbf{x}_i + b)) \\
&= \arg\max_{\mathbf{w},b} -\sum_{i=1}^{m} \log(1 + \exp(y_i(\mathbf{w}^T\mathbf{x}_i + b)))
\end{aligned}$$

Confusion Matrix

- Predicted + and predicted - on the rows
- True + and true - on the columns
- Accuracy, precision, recall
- F1 score used for unbalanced datasets

Neural Networks

- First uses a weight matrix Θ to process inputs
- Then uses an activation function on the result of the input function
- Extra bias term added to each weight matrix
- Reduces to linear model if activation function is linear
- If current layer has x units and y in next, Θ has dim. $y \times (x + 1)$

Softmax

- Solves multiclass with a single classifier
- Trained with MLE
- $\arg\min_{\mathbf{w}} \sum_i \left[\log \sum_{k \in \{1,2,\dots,K\}} \exp(\mathbf{w}_k^T\mathbf{x}_i) - \mathbf{w}_{y_i}^T\mathbf{x}_i\right]$

One Against All

- Uses entire training set
- Winner-takes-all selection
- Linear decision boundary may not always exist
- $O(K)$ weight vectors
- Unbalanced training set poses an issue

One Against One

- Smaller training set (only 2 labels included)
- Selection based on votes (each label gets $k-1$ votes)
- May be linearly separable when OAA isn't
- $O(K^2)$ weight vectors, $\binom{K}{2}$ models
- Vulnerable to data starvation $\Rightarrow$ overfitting

SVM

- Distance to margin is $\frac{1}{\|\mathbf{w}\|^2}$, is maximized
- Hard SVM requires training error = 0
- Soft SVM introduces a slack variable ξ
- $\frac{1}{2}\mathbf{w}^T\mathbf{w}$ acts as a regularization term to maximize the margin
- $\sum_i \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))$ acts as the loss to be minimized (hinge loss)
- Sub-gradients used to differentiate at the hinge
- Support vectors are data points on/in the margin

MAP Estimation

- The posterior is proportional to the likelihood times the prior
- If the prior is uniform, MAP reduces to maximum likelihood
- Solve by plugging in likelihood/prior, deriving, and finding optimal p
- MLE is same, but with additional term for the prior introduced

Learning Probabilistic Models

- $k - 1$ parameters needed for the prior
- $k(|X_1| \times |X_2| \times \cdots \times |X_d| - 1)$ parameters needed for likelihood
- $k(d - 1)$ parameters needed for likelihood if the Naive Bayes assumption is used

K-means algorithm a.k.a Llyod's algorithm

- Step 0: randomly assign the cluster centers $\{\mu_k\}$
- Step 1: Minimize J over $\{r_{nk}\}$ -- Assign every point to the closest cluster center

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\boldsymbol{x}_n - \boldsymbol{\mu}_j\|_2^2 \\ 0 & \text{otherwise} \end{cases}$$

- Step 2: Minimize J over $\{\mu_k\}$ -- update the cluster centers

$$\mu_k = \frac{\sum_n r_{nk}\boldsymbol{x}_n}{\sum_n r_{nk}}$$

- Loop until it converges

Properties of the K-means algorithm

- Does the K-means algorithm converge
  - Yes
- How long does it take to converge?
  - In the worst case, exponential in the number of data points
  - In practice, usually quick
- How good is its solution?
  - Local minimum (depends on the initialization)

Naive Bayes

- Assumes conditional independence of features
- $h_{NB} = P(y)P(x_1, x_2, \dots, x_d \mid y$
- Linear decision boundary
- $P(y_i) = p^{[y_i=1]}(1-p)^{[y_i=0]}$
- $p = \frac{\text{Count}(y_i=1)}{\text{Count}(y_i=1)+\text{Count}(y_i=0)}$
- $P(x_j = 1 \mid y = 1) = \frac{\text{Count}(y_i=1, x_{ij}=1)}{\text{Count}(y_i=1)}$
- $P(x_j = 1 \mid y = 0) = \frac{\text{Count}(y_i=0, x_{ij}=1)}{\text{Count}(y_i=0)}$

Naïve Bayes: Learning and Prediction

Learning

- Count how often features occur with each label. Normalize to get likelihoods
- Priors from fraction of examples with each label
- Generalizes to multiclass

Prediction

- Use learned probabilities to find highest scoring label

Choosing K

- Increasing K will always decrease the optimal value of the K-means objective

K-medoids algorithm

Step 0: randomly selecting K points as the cluster centers $\{\mu_k\}$
Step 1: Minimize J over $\{r_{nk}\}$ -- Assign every point to the closest cluster center

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\boldsymbol{x}_n - \boldsymbol{\mu}_j\|_2^2 \\ 0 & \text{otherwise} \end{cases}$$

Step 2: Update the cluster centers– the porotype for a cluster is the data that is closest to all other data points in the cluster

$$k = \arg\min_{n:z_n=k} \sum_n r_{nk}\|\boldsymbol{x}_n - \boldsymbol{x}_n\|_2^2$$

$$\boldsymbol{\mu}_k = \boldsymbol{x}_k$$

Loop until it converges

GMM algorithm

Step 0: randomly assign the cluster centers $\{\mu_k\}$ and covariance matrices $\{\Sigma_k\}$

Step 1: Assign every point to the cluster based on posterior distribution $P(z_n = k|x_n)$

$$p(z_n = k|\boldsymbol{x}_n) = \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{p(\boldsymbol{x}_n)} = \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{\sum_{k'=1}^{K} p(\boldsymbol{x}_n|z_n = k')p(z_n = k')}$$

Step 2: update the cluster centers $\{\mu_k\}$, and covariance matrices $\{\Sigma_k\}$ based on soft assignment $\gamma_{nk} = P(z_n = k|x_n)$

$$\omega_k = \frac{\sum_n \gamma_{nk}}{\sum_k \sum_n \gamma_{nk}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk}\boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top$$

Loop until it converges

γ_{nk} is similar to r_{nk} in K-means, but $\gamma_{nk} \in (0,1), r_{nk} \in \{0,1\}$

Lec 18: EM / GMM

$$\mu_l^* = \arg\max_{\mu_l} \mathcal{LL}(\theta) = \arg\max_{\mu_l} \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma_{nl} \cdot \log\{\mathbb{N}(x_n; \mu_l, \sigma_l^2) \cdot w_l\}$$

$$= \arg\max_{\mu_l} \sum_{n=1}^{N} \gamma_{nl} \cdot \log\ \mathbb{N}(x_n; \mu_l, \sigma_l^2)$$

$$= \arg\max_{\mu_l} \sum_{n=1}^{N} \gamma_{nl} \cdot \left\{-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log\sigma_l^2 - \frac{1}{2}\frac{(x_n - \mu_l)^2}{\sigma_l^2}\right\}$$

$$= \arg\min_{\mu_l} \sum_{n=1}^{N} \gamma_{nl} \cdot (x_n - \mu_l)^2$$

$$\mathcal{LL}(\theta) = \sum_{n=1}^{N} \log P(x_n, z_n|\theta)$$

$$= \sum_{n=1}^{N} \log \prod_{k=1}^{K} P(x_n, z_n|\theta)^{\gamma_{nk}}$$

$$= \sum_{n=1}^{N}\sum_{k=1}^{K} \underline{\quad\text{blank\#1}\quad} \text{ Solution: } \gamma_{nk} \log P(x_n, z_n|\theta)$$

$$= \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma_{nk}\{\underline{\quad\text{blank\#2}\quad} \text{ Solution: } \log P(x_n|z_n, \theta) + \log P(z_n|\theta)\}$$

$$= \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma_{nk} \log_k \mathcal{N}(x_n; \mu_k, \sigma_k^2) + \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma_{nk} \log w_k$$

Proof: We denote $\theta = (w_1, w_2, b)$ as the weights, $\theta^{(k)}$ as the weights after making k mistakes, particularly, $\theta^{(0)} = \mathbf{0}$, and $\theta^* = (w_1^*, w_2^*, b^*)$.

Consider the inner product of $\theta^{(k)}$ and θ^*. Let j be the data point of the $(k+1)$-th mistake, and we have

$$\theta^{(k+1)} \cdot \theta^* = \theta^{(k)} \cdot \theta^* + \underline{\quad\text{blank\#1}\quad} > \theta^{(k)} \cdot \theta^* + \underline{\quad\text{blank\#2}\quad}$$

Since $\theta^{(0)} \cdot \theta^* = 0$,

$$\theta^{(k)} \cdot \theta^* \geq k\gamma.$$

Consider the l2-norm of $\theta^{(k)}$.

$$\|\theta^{(k+1)}\|^2 = \|\theta^{(k)}\|^2 + \underline{\quad\text{blank\#3}\quad} \leq \|\theta^{(k)}\|^2 + 0 + 3.$$

Thus,

$$\|\theta^{(k)}\|^2 \leq 3k,$$

$$k\gamma \leq \theta^{(k)} \cdot \theta^* \leq \|\theta^{(k)}\|\|\theta^*\| \leq \sqrt{3k},$$

$$k \leq \frac{3}{\gamma^2}.$$

Solution:
Blank1: $y^{(j)}(w_1^* x_1^{(j)} + w_2^* x_2^{(j)} + b^*)$, or $\theta^* \cdot y^{(j)}\mathbf{x}^{(j)}$. Here $\mathbf{x}^{(j)} = (x_1^{(j)}, x_2^{(j)}, 1)$
Blank2: γ. Some equivalent replacements as Blank1 are not regarded as correct answers.
Blank3: $2y^{(j)}(w_1^{(k)} x_1^{(j)} + w_2^{(k)} x_2^{(j)} + b^{(k)}) + (x_1^{(j)2} + x_2^{(j)2} + 1)$, or $2\theta^{(k)} \cdot y^{(j)}\mathbf{x}^{(j)} + \|\mathbf{x}^{(j)}\|^2$. It's also correct to add a coefficient $y^{(j)2}$ for the second term because it is 1.

Solution: Since

$$l_a = \min_{x \in S_p} \|x\|_2, \quad r_a = \max_{x \in S_p} \|x\|_2,$$

we know all positive examples are in the ring of $h_{(l_a, r_a)}$ and they are correctly predicted. Next, from the definition of $h_{(l_a, r_a)}$, we know all the negative examples outside the ring of $h_{(l_a, r_a)}$ are correctly predicted. Finally, since the label of data point is generated by the target function $h^*_{(l^*, r^*)}$, we have $l^* \leq \|x\|_2 \leq r^*$ for all $x \in S_p$. That means $l^* \leq l_a \leq r_a \leq r^*$. Therefore, there is no negative examples in the ring of $h_{(l_a, r_a)}$. Combine all of above, we know all the training examples are correctly predicted and thus the training error is zero.

Solution: 0.20

$$P(Y|D, p_A^{(0)}, p_B^{(0)}) = \frac{P(D|Y, p_A^{(0)}, p_B^{(0)})P(Y)}{P(D|p_A^{(0)}, p_B^{(0)})}$$

Since Y has uniform prior, and denominator is independent from Y, we only consider $P(D|Y, p_A^{(0)}, p_B^{(0)})$.

$$P(D|Y^{(1)} = AA, p_A^{(0)} = 0.4, p_B^{(0)} = 0.6) = 0.4^3 \times (1 - 0.4)^2,$$

$$P(D|Y^{(1)} = AB, p_A^{(0)} = 0.4, p_B^{(0)} = 0.6) = 0.4^2 \times (1 - 0.4) \times 0.6 \times (1 - 0.6).$$

$$P(D|Y^{(1)} = BA, p_A^{(0)} = 0.4, p_B^{(0)} = 0.6) = 0.6^2 \times (1 - 0.6) \times 0.4 \times (1 - 0.4).$$

$$P(D|Y^{(1)} = BB, p_A^{(0)} = 0.4, p_B^{(0)} = 0.6) = 0.6^3 \times (1 - 0.6)^2.$$

$$P(Y^{(1)} = AA|D, p_A^{(0)}, p_B^{(0)}) = \frac{P(D|Y^{(1)} = AA, p_A^{(0)}, p_B^{(0)})}{\sum_{Y' \in \{AA, AB, BA, BB\}} P(D|Y^{(1)} = Y', p_A^{(0)}, p_B^{(0)})} = 0.20.$$

Similarly,

$$P(Y^{(1)} = AB|D, p_A^{(0)}, p_B^{(0)}) = 0.20$$

$$P(Y^{(1)} = BA|D, p_A^{(0)}, p_B^{(0)}) = 0.30$$

$$P(Y^{(1)} = BB|D, p_A^{(0)}, p_B^{(0)}) = 0.30$$

❖ Now, compute the most likely value of the parameters. [recall the scenario I]

0.826 HHHHT
0.174 THHHT
0.703 HHTHT
0.297 THTHT
0.826 HHHHT
0.174 THHHT
0.703 HHTTH
0.297 THTTH

$$\alpha_2 = \frac{0.826\times2 + 0.703\times2}{4} = 0.765$$

$$p_2 = \frac{0.826\times6 + 0.703\times4}{0.826\times8 + 0.703\times8} = 0.635$$

$$q_2 = \frac{0.174\times6 + 0.297\times4}{0.174\times8 + 0.297\times8} = 0.592$$