Ryan Yee - CS 760 Midterm Note Sheet

Overview

$\mathcal{X} \rightarrow$ input space
$\mathcal{Y} \rightarrow$ output space
$\mathcal{H} \rightarrow$ hypothesis class
Goal:
model $h \in \mathcal{H}$ that best approximates $f : \mathcal{X} \rightarrow \mathcal{Y}$

Empirical Risk Minimization

$\hat{f} = \arg\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell(h(x^{(i)}), y^{(i)})$

k Nearest Neighbors

$\hat{y} = \sum_{i=1}^{k} y^{(i)}$ for k closest points
Distances:
Hamming: $d_H(x^{(i), x^{(j)}}) = \sum_{a=1}^{d} \mathbb{1}(x_a^{(i)} \neq x_a^{(j)})$
Euclidean: $d(x^{(i)}, x^{(j)}) = \left(\sum_{a=1}^{d} (x_a^{(i)} - x_a^{(j)})^2\right)^{-\frac{1}{2}}$
Manhattan: $d(x^{(i)}, x^{(j)}) = \sum_{a=1}^{d} |x_a^{(i)} - x_a^{(j)})^2|$

Information Theory

Entropy: $H(Y) = -\sum_{y \in Y} \mathbb{P}(y) \log_2(\mathbb{P}(y))$
$H(Y|X) = \sum_{x \in X} \mathbb{P}(X = x) H(Y|X = x)$
Information Gain: $H(Y) - H(Y|X)$
Gain Ratio: $\frac{H(Y) - H(Y|S)}{H(S)}$

Evaluation

		Observed Class	
		True	False
Predicted Class	True	TP	FP
	False	FN	TN

Accuracy: $\frac{TP+TN}{TP+FP+TN+FN}$
Error: 1 - Accuracy
Precision: $\frac{TP}{TP+FP}$
Recall: $\frac{TP}{TP+FN}$
False Positive Rate: $\frac{FP}{TN+FP}$
ROC Curve: recall (TPR) vs. false positive rate
Precision/Recall Curve: downward sloping with asymptote

Decision Trees

Algorithm:
MakeSubtree:
Determine Candidate Splits
if stopping criteria: make leaf
else: *MakeSubtree* with best split
return subtree rooted at N

Linear Regression

Loss: $\ell(f_\theta) = \frac{1}{n}||\mathbf{X}\theta - y||_2^2$
Gradient: $\nabla_\theta = \frac{1}{n}(2\mathbf{X}^T\mathbf{X}\theta - 2\mathbf{X}^T y)$
Solution: $\theta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y$
Ridge Loss: $\ell(f_\theta) = \frac{1}{n}||\mathbf{X}\theta - y||_2^2 + \lambda||\theta||_2^2$
Ridge Sol: $\theta = (\mathbf{X}^T\mathbf{X} + \lambda n I)^{-1}\mathbf{X}^T y$

Logistic Regression

Sigmoid: $\sigma(z) = \frac{1}{1+\exp(-z)} = \frac{\exp(z)}{1+\exp(z)} \in (0, 1)$
Properties: $1 - \sigma(z) = \sigma(-z)$
$\sigma'(z) = \sigma(z)(1 - \sigma(z))$
Cross Entropy Loss: $-[y \log \hat{y} + (1-y)\log(1-\hat{y})]$

Estimation

Maximum Likelihood:
$\hat{\theta}_{MLE} = \arg\max_\theta \mathcal{L}(\theta; X)$
$\mathcal{L}(\theta; X) = \prod_{i=1}^{n} \mathbb{P}_\theta(x_i)$
Conditioned on X:
$\hat{\theta}_{MLE} = \arg\max_\theta \mathcal{L}(\theta; Y, X)$
$\mathcal{L}(\theta; Y, X) = \prod_{i=1}^{n} \mathbb{P}_\theta(y_i|x_i)$
Maximum a posteriori Probability:
$\hat{\theta}_{MAP} = \arg\max_\theta \prod_{i=1}^{n} p(x^{(i)}|\theta)p(\theta)$

Gradient Descent

Convergence Criteria:
1. Convex:
$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$
2. Differentiable
3. Lipschitz-continuous: $\nabla^2 f(x) \preccurlyeq LI$
recall: ($B - A$ is positive semidefinite if $A \preccurlyeq B$)
recall: if C is pos. semidefinite then $x^T C x \geq 0 \ \forall x$

Naive Bayes

Assumes conditional independence of features:

$$P(X_1, \ldots, X_k, Y) \propto P(X_1, \ldots, X_k \mid Y) P(Y) = \left(\prod_{k=1}^{K} P(X_k \mid Y)\right) P(Y)$$

Perceptrons

$$\hat{y}(x) = \begin{cases} 1, & w^T x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

Algorithm:
for index i:
if $y^{(i)} w^T x^{(i)} < 1$ (i.e prediction is wrong):
then $w_{t+1} = w_t + y^{(i)} x^{(i)}$
else $w_{t+1} = w_t$
Mistake Bound: $(2 + D(S)^2)\gamma(S)^{-2}$
where $D(S)$ is the max diameter and $\gamma(S)$ is the largest margin we can have with dataset S

Neural Networks

For a single internal node:
Input: x, Weights: w, Bias: b, Activation: s
Output: $s(w^T x + b)$ which feeds into the next layer
Gradient Components:
$\frac{\partial l}{\partial w} = (\hat{y} - y)x$, $\frac{\partial l}{\partial x} = (\hat{y} - y)w$
2-Layer:
$\frac{\partial l}{\partial w_{11}^{(1)}} = \frac{\partial l}{\partial a_{11}} \frac{\partial a_{11}}{\partial w_{11}^{(1)}} = (\hat{y} - y) w_{11}^{(2)} a_{11}(1 - a_{11}) x_1$
$\frac{\partial l}{\partial x_1} = \frac{\partial l}{\partial a_{11}} \frac{\partial a_{11}}{\partial x_1} + \frac{\partial l}{\partial a_{12}} \frac{\partial a_{12}}{\partial x_1}$
$= (\hat{y} - y) w_{11}^{(2)} a_{11}(1 - a_{11}) w_{11}^{(1)} + (\hat{y} - y) w_{21}^{(2)} a_{12}(1 - a_{12}) w_{12}^{(1)}$

L2 Regularization

Effect on GD:
Loss: $\hat{|}_R(\theta) = \hat{L}(\theta) + \frac{\lambda}{2}||\theta||_2^2$
Gradient: $\nabla \hat{L}_R(\theta) = \nabla \hat{L}(\theta) + \lambda\theta$
GD Update: $\theta_{t+1} = (1 - \nu\lambda)\theta_t - \nu\nabla\hat{L}(\theta_t)$
Effect: decays weights by $(1 - \nu\lambda)$
Effect on Optimal Solution:
$\theta_R^* \approx (H + \lambda I)^{-1} H\theta^* = Q(\Lambda + \lambda I)^{-1}\Lambda Q^T\theta^*$
Effect: shrinks along eigenvectors of H

Other Regularization

1. Data augmentation (based on domain)
e.g. crop, rotate, thesaurus/back-translate
2. Adding noise (equivalent to weight decay)
3. Early stopping
4. Dropout (randomly select weights to update)

2D-Convolution

Idea: use an $k_h \times k_w$ kernel matrix which takes the sum product of the pixels in the image
Padding: adds p_h rows and p_w columns cushion on the edge of the image to preserve information
Stride: rows (s_h) and columns (s_w) per slide
Given $n_h \times n_w$ input, output will be $[(n_h - k_h + p_h + s_h)/s_h] \times [(n_w - k_w + p_w + s_w)/s_w]$
Pooling: Similar to using a kernel but can do non-linear transformations such as max pooling

3D-Convolution

Let c_i and c_o be the # of input and output channels
X: $c_i \times n_h \times n_w$
W: $c_i \times c_o \times k_h \times k_w$
Y: $c_o \times m_h \times m_w$

Activation Functions

$\sigma(x) = \frac{e^x}{1+e^x}$
$\text{ReLU}(x) = \max\{0, x\}$
$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 2\sigma(2x) - 1$
$\frac{d}{dx}\tanh(x) = 1 - \tanh^2(x)$