Spring 2022 Data 100/200 Final Reference Sheet

Principal Component Analysis (PCA)

The i-th Principal Component of the matrix X is defined as the i-th column of $U\Sigma$ defined by Singular Value Decomposition (SVD).

$X = U\Sigma V^T$ is the SVD of X if U and V^T are orthonormal matrices and Σ is a diagonal matrix. The diagonal entries of Σ, $[s_1, \ldots, s_r, 0, \ldots, 0]$, are known as singular values of X, where $s_i > s_j$ for $i > j$ and $r = \text{rank}(X)$.

Define the design matrix $X \in \mathbb{R}^{n \times p}$. Define the total variance of X as the sum of individual variances of the p features. The amount of variance captured by the i-th principal component is equivalent to s_i^2/n, where n is the number of datapoints.

Logistic Regression and Classification

Logistic Regression Model: For input feature vector x, $\hat{P}_\theta(Y = 1|x) = \sigma(x^T\theta)$. The estimate $\hat{\theta}$ is the parameter θ that minimizes the average cross-entropy loss on training data. For a single datapoint, define cross-entropy loss as $-[y\log(p) + (1-y)\log(1-p)]$, where p is the probability that the response is 1.

Logistic Regression Classifier: For a given input x and trained logistic regression model with parameter θ, compute $p = \hat{P}(Y = 1|x) = \sigma(x^T\theta)$. predict response $\hat{y}$ with classification threshold T as follows:

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

Confusion Matrix

Columns are the predicted values $\hat{y}$ and rows are the actual classes y.

	0	1
0	True negative (TN)	False Positive (FP)
1	False negative (FN)	True Positive (TP)

Classification Performance

Suppose you predict n datapoints.

Metric	Formula	Other Names	Visualization	Plot
Accuracy	$\frac{TP+TN}{n}$		Precision-Recall Curve	Precision vs. Recall for different thresholds T
Precision	$\frac{TP}{TP+FP}$		ROC Curve	TPR vs. FPR for different thresholds T
Recall/TPR	$\frac{TP}{TP+FN}$	True Positive Rate, Sensitivity		
FPR	$\frac{FP}{FP+TN}$	False Positive Rate, Specificity		

Scikit-Learn

Suppose `linear_model` is an imported `sklearn` package.

Class/Attribute	Description	Function	Description
`linear_model.LogisticRegression( fit_intercept=True, penalty='l2', C=1.0)`	Returns an ordinary least squares Linear Regression model. Hyperparameter C is inverse of regularization parameter, C = 1/λ.	`model.fit(X, y)`	Fits the scikit-learn `model` to the provided `X` and `y`.
`model.coef_`	Estimated coefficients for the model, not including the intercept term.	`model.predict_proba(X)`	Returns predicted probabilities for the `X` passed in according to the fitted `model`. If binary classes, will return probabilities for both class 0 and 1.
`model.intercept_`	Bias/intercept term of the model. Set to 0.0 if `fit_intercept=False`.	`model.predict(X)`	Returns predictions for the `X` passed in according to the fitted `model`.
		`model.score(X, y)`	Returns the average `model` accuracy on the given test data `X` and labels `y`.

Suppose `tree` and `ensemble` are imported `sklearn` packages.

Class/Function	Description
`tree.DecisionTreeClassifier(criterion='entropy', max_depth=None)`	Returns a decision tree model which uses `criterion` to measure the quality of a split. `max_depth` is the maximum depth of the tree; if `None`, then nodes are expanded until all leaves are pure.
`ensemble.RandomForestClassifier(n_estimators=100, criterion='entropy', max_depth=None)`	Fit `n_estimators` decision tree classifiers on sub-samples of the dataset.
`model.fit(X, y)`	Decision tree: Fit a decision tree `model` to the provided `X` and `y`. Random forest classifier: Build a forest `model` of decision trees fit to the provided `X` and `y`.
`model.predict(X)`	Decision tree: Returns predicted response for the `X` passed in according to the fitted `model`. Random forest classifier: Returns the predicted class by highest mean probability estimate according to the trees in the forest `model`.

Clustering

K-Means Clustering: Pick an arbitrary k, and randomly place k "centers", each a different color. Then repeat until convergence:

1. Color points according to the closest center (defined as squared distance).
2. Move center for each color to center of points with that color.

K-Means minimizes inertia, defined as the sum of squared distances from each datapoint to its center.

Agglomerative Clustering: Assign each datapoint to its own cluster. Then, recursively merge pairs of clusters together until there are k clusters remaining.

A datapoint's **silhouette score** S is defined as $S = (B - A)/\max(A, B)$, where A is the mean distance to other points in its cluster, and B is the mean distance to points in its closest cluster.

Decision Trees and Random Forests

Suppose you have a **decision tree classifier** for k classes. For each node, define the probability for class $C \in \{1, \ldots, k\}$ as $p_C = d_C/d$, where d_C is the number of datapoints in class C (of the d total in the node). Then the entropy of the node (in bits) is defined as $S = -\sum_C p_C \log_2 p_C$, and the weighted entropy of the node is its entropy scaled by the fraction of datapoints in that node.

Decision tree generation algorithm: All of the data starts in the root node. Repeat until every node is either pure or unsplittable:

- Pick the best feature x and best split value β, where β is picked to maximize the change in weighted entropy between the parent node and the child nodes.
- Split data into two nodes, one where x < β, and one where x ≥ β.

A node that has only one samples from one class is called a "pure" node. A node that has overlapping data points from different classes and thus that cannot be split is called "unsplittable".

A **random forest** is a collection of many decision trees fit to variations of the same training data (e.g., bootstrapped samples, also called bagging; or random subsets of features). It is an ensemble method.