Machine Learning Summarized (Work in Progress)

Daniel Tang: B.S. Computer Science, Data Science

Rice University, March 5, 2020

1 Linear Regression

1.1 Definitions

- **Regression:** When the output you are trying to estimate or predict is a continuous valued number
- **Classification**: When the output you are trying to estimate or predict is a categorical quantity.

1.2 Hypothesis and Cost Function

The linear regression hypothesis:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Total cost of predictions for the whole training set ($2m$ is present for the purpose of calculating the gradient):

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Goal:

$$minimize_{\theta_0, \theta_1} J(\theta_0, \theta_1)$$

1.3 Gradient Descent

- Iterative Approach
- Requires a convex cost function
- Given a function $J(\theta_0, \theta_1)$, start with some θ_0, θ_1, and update θ_0, θ_1 such that it reduces $J(\theta_0, \theta_1)$.

$$\theta_0^{[k+1]} = \theta_0^{[k]} - \alpha \frac{\partial J(\theta_0, \theta_1)}{\partial \theta_0}$$

$$\theta_1^{[k+1]} = \theta_1^{[k]} - \alpha \frac{\partial J(\theta_0, \theta_1)}{\partial \theta_1}$$

- There's a minus sign because if the gradient is negative, we want to increment θ to move towards the minimum, and if the gradient is negative we want to move backwards.
- α is the learning rate. If α is too small, we converge prohibitively slowly.
- If α is too large, we may diverge, and jump around the curve, and to the other side of the curve.

1.4 Normal Equation

- If $x^T x$ is invertible you can use the normal equation to minimize θ^*

$$\theta^* = (x^T x)^{-1} x^T y$$

2 Multiple Linear Regression

2.1 Hypothesis and Cost Function

The multiple linear regression hypothesis:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$$

This can be written as:

$$h_\theta(x) = \theta^T x^{(i)}$$

Where:

$$\theta^T = [\theta_0, \theta_1, \theta_2, \theta_3, \ldots, \theta_n],\ x^{(i)} = [1, x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}]$$

i ranges from 1 to the number of tuples in our data
Generalized cost function (same as linear regression):

$$J(\theta_0, \theta_1, \theta_2, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

2.2 Normal Equation (Same as Linear Regression)

- If $x^T x$ is invertible you can use the normal equation to minimize θ^*

$$\theta^* = (x^T x)^{-1} x^T y$$

3 Feature Normalization

3.1 Why Normalization/Scaling?

- Data-sets generally contain features highly varying in magnitudes, units and range.
- Machine learning algorithms typically use Euclidean distance to measure the distance between data points.
- This causes a problem because larger magnitude features will weigh in more heavily compared to lower magnitude features.

3.2 Min-Max Normalization

$$x_{j,normalized}^{(i)} = \frac{x_j^{(i)} - x_{j,min}}{x_{j,max} - x_{j,min}}$$

- Used most notably in re-scaling image pixel values.
- **Prone to outlier issues**.

3.3 Z-Score Normalization

$$x^{(i)}_{j,normalized} = \frac{x^{(i)}_j - \mu_j}{\sigma_j}$$

- More robust to outliers

3.4 Algorithms

- **Algorithms that require scaling:** PCA, kNN, gradient descent algorithms.
- **Algorithms that don't require scaling:** Tree based algorithms, Naive Bayes, LDA.

4 Logistic Regression

4.1 Hypothesis

- Still regression but used for classification.
- Uses a logistic function to model a binary dependent variable.

Hypothesis:

$$h_\theta(x) = g(\theta^T x^{(i)}),\ g(z) = \frac{1}{1+e^{-z}}$$

After substitution:

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x^{(i)}}},\ 0 \leq h_\theta(x) \leq 1$$

4.2 Interpreting the Hypothesis

- $g(\theta^T x^{(i)}) \leq 0.5 \implies y = 0$ happens when $\theta^T x^{(i)} \leq 0$
- $g(\theta^T x^{(i)}) > 0.5 \implies y = 0$ happens when $\theta^T x^{(i)} > 0$
- $\theta^T x^{(i)}$ is the **decision boundary**, the line that best separates the data.

4.3 Cost Function

- We can't use the linear regression cost function (substituting $h_\theta(x)$ above) for logistic regression because the cost function will not be strictly convex.
- This causes non-convergence in gradient descent.

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)})$$

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))$$

- The y terms basically say that if $y = 1$ then we use $-\log(h_\theta(x^{(i)}))$ as our $Cost(h_\theta(x^{(i)}), y^{(i)})$, and if y = 0 we use $-\log(1-h_\theta(x^{(i)}))$ as our $Cost(h_\theta(x^{(i)}), y^{(i)})$.

4.4 Gradient Descent

$$\theta_j^{[k+1]} = \theta_j^{[k]} - \alpha\frac{\partial J(\theta)}{\partial \theta_j}$$

$$\theta_j^{[k+1]} = \theta_j^{[k]} - \frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

5 Fitting and Bias/Variance

- **Training Data:** Used to train the model.
- **Test Data:** Used to evaluate performance and checking the generalization error.
- **Bias:**
  - How much averaged predicted values differ from true values.
  - $Bias[\hat{f}(x)] = \mathbb{E}[\hat{f}(x) - f(x)]$
- **Variance:**
  - How much predicted values differ from averaged predicted values.
  - $Var[\hat{f}(x)] = \mathbb{E}[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])^2]$
- **High Bias:**
  - Pays too little attention to training data.
  - Oversimplified/Under-fitted.
  - High error on training set.
- **High Variance:**
  - Pays too much attention to data.
  - Over-fitted.
  - Higher error on test set.
- **Underfitting:**
  - Training error and test error are high.
  - Too few or too simple features.
- **Overfitting:**
  - Cannot generalize well to new data.
  - Memorizes the training data.
  - Low training error, and high test error.

5.1 Fixing Overfitting (Harder than fixing Underfitting)

- More data (more rows)!
- Reduce # of features or dimension.
- Regularization:
  - Keep all the features but penalize some features/values of parameters.
  - This is particularly useful when we have a lot of features, each contributing a bit to the prediction.

5.2 Bias-Variance Decomposition

- We write the relationship between predictor variables X and the response Y as variables:
  - $Y = f(X) + \epsilon,\ \epsilon \sim \mathcal{N}(0, \sigma_\epsilon)$
- Then the expected value of the squared error can be written as:
  - $SE(x) = \mathbb{E}[(Y - \hat{f}(x))^2)]$
  - After some transformations: $SE(x) = \mathbb{E}[(\mathbb{E}[\hat{f}(x) - f(x)])^2 + \mathbb{E}[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])^2] + \sigma_e^2$
  - In words: $SE(x) = bias^2 + variance - irreducible\ error$

5.3 Occam's razor

- The simplest solution tends to be the correct one.
- When presented with competing hypotheses to solve a problem, one should select the solution with the fewest assumptions.

5.4 Best Practices!

- Suppose you are using some algorithm on a given training set but it makes unacceptably large errors in its predictions on unseen data.
  - Get more training examples which fixes high variance/overfitting.
  - Try smaller sets of features, fixes high variance/overfitting.
  - Try getting additional features, fixes high bias/underfitting.
  - Try adding polynomial features $x_1,\ x_2,\ x_1^2,\ x_2^2$, fixes highbias/underfitting.
- Learning curves ($Error_{train}$ and $Error_{test}$ vs. increasing number of tuples) help diagnose problems in terms of bias and variance.

6 Regularization

6.1 Definition

- Technique used for tuning the function by adding an additional penalty term in the error function.
- Rules out or minimizes the impact of certain features before we build our models.
- Same cost function as linear regression with an extra term.
- Reduces overfitting.

6.2 Cost Function

$$J(\theta_0, \theta_1, \theta_2, \ldots, \theta_n) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2$$

6.3 Gradient Descent

$$\theta_j^{[k+1]} = \theta_j^{[k]} - \alpha\frac{\partial J(\theta)}{\partial \theta_j}$$

$$\theta_j^{[k+1]} = \theta_j^{[k]} - \frac{\alpha}{m}[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \lambda\theta_j^{[k]}], \text{ For } \theta_0, \text{ the } \theta_j^{[k]} \text{ term is } 0$$

- λ trades off training error reduction with number of features to be included ($\theta_j \neq 0$) in the model
  - λ small $\implies$ Many features included, smaller training error, but prone to over-fitting.
  - λ large $\implies$ Few features included, larger training error, prone to under-fitting.

6.4 Normal Equation

The normal equation method to find the θ^* that minimizes the cost function for the linear regression problem WITH regularization is:

$$\theta^* = (x^T x + \lambda \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \dots & 1 \end{bmatrix})^{-1} x^T y, \ (n+1) \times (n+1) \text{ matrix}$$

This helps reduce invertibility problems as well!!!

6.5 Logistic Regression Regularization

We can use regularization in logistic regression as well.

$$J(\theta_0, \theta_1, \theta_2, \dots, \theta_n) = \frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

7 Feature Engineering

Feature engineering is the process of using domain knowledge to extract features from raw data via data mining techniques.

7.1 Curse of Dimensionality

- A name for various problems that arise when analyzing data in high dimensional space.
- Dimensions = independent features in ML.
- Occurs when d (dimensions) is large in relation to n (number of samples).
- Multiple ML algorithms use the concept of Euclidean distance to measure distance between points. High dimensional data causes all points to seem equidistant.
- High dimensionality leads to sparse data, a few points in a large space. This is because volume increases faster than data.
- More features lead to more noise making it harder to find the true signal.
- **Example:** Genomics: We have approx. 20,000 genes, but samples sizes for diseases range in the 100s and 1000s.

7.2 Combating the Curse

- Reduce dimensions: Use feature selection like regularization or PCA (transforms data to lower dimension space).
- Results in information loss but that's okay.

7.3 How to Feature Select

- **Filter method**:
  - Ranks features or feature subsets independently of the classifier.
  - Select subsets of variables as a pre-processing step.
  - Not tuned by a given learner (can be good or bad)
- **Wrapper Method:**
  - Uses a predictive model (machine learning) to score feature subsets.

- Often better than filter method. They are accurate, rarely over-fitting, slow, and model dependent.
- Requires training a model for each feature set.
- **Forward selection:** Start with an empty feature set and add features at each step.
- **Backward selection:** Start with a full feature set and discard features at each step.
- Forward and Backward selection are both Greedy.

- **Embedded method:**
  - Performs variable selection (implicitly) in the course of model training (e.g. decision tree).
  - Popular one is Lasso Regularization.

7.4 Common Scoring Functions

- **Pearson Correlation:**
  - Choose features (x) correlated with label (y)
  - Can only detect linear dependencies

$$r_{x_i y} = \frac{Cov(x_i, y)}{\sqrt{Var(x_i)Var(y)}}$$

- **Mutual Information:**
  - "How much information two variables share".
  - If two variables are independent, $MI = 0$.
  - Can detect any form of statistical dependency.

$$MI(x_i, y) = \int\int p(x_i, y) \log(\frac{p(x_i, y)}{p(x_i)p(y)}) dxdy$$

7.5 Best Practices

- **Label Association:** Choose top 10 features associated (Pearson correlation) to the label.
- **Low variation features:** Remove features with little variation.
- **Correlated features:** Keep only one of two highly correlated features.

8 K-Nearest Neighbors

8.1 What is it?

- The idea is that similar examples have similar label and that we classify new examples like similar training examples.
- Given some new example x for which we need to predict its class y, find the most similar training examples, and classify x like those training examples.
- kNN is a lazy learner because there are no pre-constructed model for classification.

8.2 Algorithm

1. To classify an unknown record, first compute distance to other training records.
2. Identify k nearest neighbors.
3. Select the class based on the majority vote in the k nearest neighbors to determine the class label of unknown record.

8.3 Measuring Distance

- **Euclidean Distance:**

$$dist(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{i=1}^{n}(a_i - p_i)^2}$$

- **Hamming Distance:**
  - bat vs cat (distance = 1).
  - toned vs roses (distance = 3).
- **Discrete Metric (Boolean Metric):**

$$dist(x, y) = 0 \text{ if } x = y; \text{ else } dist(x, y) = 1$$

8.4 Selecting Number of Neighbors

- Increasing k makes kNN less sensitive to noise.
- Decreasing k allows capturing finer structure of space.
- Typically choose that value of k which has lowest error rate in validation data.

8.5 Curse of Dimensionality

- When number of attributes increases, we lose relevance in the majority of the attributes.
- Distance metric breaks down as described previously.
- To fix, we can remove irrelevant attributes in pre-processing step.
- Weight attributes differently.
- Increase k (with reason).

9 Model Evaluation

9.1 Is Accuracy Always the Best Metric?

- The answer is maybe, depending on your data.
- Accuracy is defined as the closeness to true values.
- How often is the classifier correct overall?
- Does not tell us the false negatives, false positives, true negatives, and true positives.
- The confusion matrix allows us to present false negatives, false positives, true negatives, and true positives.
- Given an unbalanced data-set where 99% of our data are of one class, and 1% of our data is of the other class. If we label all data as the first class, we see a 99% accuracy, but the model is clearly incorrect!

9.2 More informative Metrics

- **Recall: How many of the accurate predictions did we capture?**

$$\frac{TP}{FN+TP}$$

- **Precision: What percent of our predictions are accurate?**

$$\frac{TP}{FP+TP}$$

- **F1 Score: The harmonic mean of precision and recall, ranges between 0 and 1.**

$$F1-Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

9.3 Receiver Characteristic Curve (ROC-Curve)

- Plots the True positive rate (TPR) vs. the False positive rate (FPR)
- The maximum area under the curve (AUC) is 1.
- Completely random predictions have an AUC of 0.5.

10 Support Vector Machine

10.1 What is it?

- Finding a hyperplane that separates positive points from negative points as far as possible.
- The hyperplane must maximize the distance from each sample. (Hyperplane π is margin maximizing)
- π^+ (positive hyperplane) parallel to π^- (negative hyperplane) parallel to π
- Support vectors are points that π^+ and π^- pass through.

10.2 Mathematical formulation

Hard-margin SVM

$$Margin\ d = \frac{2}{||w||}$$

$$\text{Find } (w*, b*) = argmax_{w,b} \frac{2}{||w||}\ s.t.\ \forall x_i,\ y_i(w^T x_i + b) \geq 1$$

If our data is not completely linearly separable, we can instroduce a slack variable. This is called **soft-margin SVM**. Our hinge loss will be $C\frac{1}{n}\sum_{i=1}^{n}\zeta_i$.

$$Margin\ d = \frac{2}{||w||}$$

$$argmax_{w,b}\frac{2}{||w||} = argmin_{w,b}\frac{||w||}{2}$$

$$\text{Find } (w*, b*) = argmin_{w,b}\frac{||w||}{2} + C\frac{1}{n}\sum_{i=1}^{n}\zeta_i\ s.t.\ \forall x_i,\ y_i(w^T x_i + b) \geq 1 - \zeta_i \text{ where } \zeta_i \geq 0$$

10.3 Tuning Hyper-parameter C

- If C increases, tendency to make mistakes decreases on training data $\implies$ small margin, over-fitting.
- If C decreases, high bias $\implies$ large margin, under-fitting

10.4 Multiple Class SVM

- Split the task into k binary tasks and learn k SVMs:
  - Class 1 vs. the rest (classes 2 - k)
  - Class 2 vs. the rest (classes 1, 3 - k)
  - Class k vs. the rest
- Pick the class that puts the point farthest into its positive region.

10.5 The Kernel Trick

If every data point is mapped into high-dimensional space via some transformation $\phi : x \to \phi(x)$, the dot product becomes:

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

A kernel function is some function that corresponds to an inner product in some expanded feature space.

11 K-means

11.1 What is it?

- K-means is an unsupervised algorithm, an algorithm that clusters points when a label is unavailable or not 100% reliable.

11.2 Algorithm

- K-means requires a K, the number of clusters, and a training set:

$$\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\},\ x^{(i)} \in \mathbb{R}^p$$

.

- p features for each observation.

1. **Initialize centroids** to be any point in the plane. Typically you pick two observations in your training data.
2. **Find Closest Centroids:**
   - Calculate the distance of each point to the centroids.
   - 'Assign' each point to the centroid it is closest to.
   - The points assigned to a given centroid define the 'cluster'
3. **Recompute cluster centroids:**
   - Calculate the average of all the points within each cluster.
   - This defines the new centroid locations
4. **Iteratively do steps 2 and 3 with Stopping Criterion:** No change in cluster assignment.

11.3 Cost Function

$$J(c^{(1)}, c^{(2)}, \ldots, c^{(m)}, \mu_1, \mu_2, \ldots, \mu_k) = \frac{1}{m} \sum_{i=1}^{m} ||x^{(i)} - \mu_{c^{(i)}}||_2^2$$

Where $c^{(i)}$ is the cluster for the i^{th} observation, $x^{(i)}$, μ_k is the centroid location of the k^{th} cluster, and $\mu_{c^{(i)}}$ is the centroid location of the cluster to which $x^{(i)}$ belongs.
In **Find Closest Centroids**, we find the $\{c^{(1)}, c^{(2)}, \ldots, c^{(m)}\}$ that minimizes J while keeping $\{\mu_1, \mu_2, \ldots, \mu_k\}$ fixed.
In **Recompute Cluster Centroids**, we find the $\{\mu_1, \mu_2, \ldots, \mu_k\}$ that minimizes J while keeping $\{c^{(1)}, c^{(2)}, \ldots, c^{(m)}\}$ fixed.

11.4 Initializing centroids

- In random initialization of centroids, we may get get stuck in local optima for the centroid choices.
- To solve this we run K-means several times with different initial centroids and compute J each time.

11.5 Choosing Number of Clusters, K

- **The Elbow Method:**
  1. Plot the minimized cost function value of J for different choices of K.
  2. Calculate the change in slope (angle) of the curve at every K.
  3. Set K as the number of clusters where the change in slope is highest, aka the elbow point.
- **The Silhouette method:**
  - The silhouette index $s(i)$ measures how appropriate is the cluster assignment for $x^{(i)}$
  - $a(i)$: Average distance between $x^{(i)}$ and all points in its assigned cluster.
  - $b(i)$: Minimum of average distances between $x^{(i)}$ and points in other clusters.

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))}$$

  - When $a(i) \ll b(i)$, $s(i) \approx 1 \implies x^{(i)}$ in correct cluster.
  - When $a(i) \gg b(i)$, $s(i) \approx -1 \implies x^{(i)}$ in wrong cluster.
  - When $a(i) = b(i)$, $s(i) = -1 \implies x^{(i)}$ can go in either cluster.
- **Cross-validation**
  - Split your data into 'v' parts. Use 'v-1' parts for training (running K-means with different K) and 1 part for testing (compute J using only the test points). Choose the K such that increasing k has minimal effect on J.
- **Expert Suggestion**
  - Let the expert in the field suggest the right number of clusters based on their knowledge.
    1. In tumor classification for a particular cancer, if the possible stages are I,II, III, and IV, then it makes sense to set $K = 4$.
    2. If glucose profiles can only be of six distinct types based on biology, then $K = 6$ is a good starting point.

12 Dimensionality Reduction and Principal Component Analysis (PCA)

12.1 Why Dimensionality Reduction?

- Data compression helps us store more files than we otherwise could.
- Lot of REDUNDANCY in the data which we can throw away without losing vital. information

12.2 The Theory

- Highly correlated columns can be graphed to form a linear graph. This can then be rotated so that only one column is needed to represent all the data. This corresponds to one axis being sufficient to explain the data.
- We find the principle component axis such that the sum of distances from each point to the axis is the least.
- DIFFERENT from linear regression. Linear regression finds line that minimizes the vertical distances between y and $\hat{y}$. PCA finds the line that minimizes the projection distances between features x_1 and x_2.
- We almost always normalize our data using z-scoring because without mean centering, the origin of the PC axes will be forced to be the original $(0, 0)$,thus leading to choosing not-useful axes as the PC's. Without scaling, PC's will be dominated by the features that have a large scale.
- Exceptions to this are when our features are similar in scale, in which case feature scaling may remove useful information. This is uncommon though and not a huge problem.

12.3 Singular Value Decomposition (SVD) for PCA

- Singular value decomposition is a factorization of a real or complex matrix that generalizes the eigendecomposition of a square normal matrix to any matrix via an extension of the polar decomposition.
$$A = U\Sigma V^T$$
- V^T is responsible for the first rotation.
- Σ is responsible for scaling.
- U is responsible for rotating back.
- We will focus on computing the Principle Components (PC) using the covariance matrix
$$K = \frac{X^T X}{m}$$
.
- $[U, \Sigma, V] = svd(K)$
  - The columns of U define the new axes or the Principal Components
  - The PCs of U are in decreasing order of data variation explained.
  - U is a $p \times p$ matrix where p corresponds to the number of dimensions in the compressed feature space.
  - Σ allows us to compute the variance explained.
  - Choose the smallest k such that variance explained satisfies:
$$Var_{explained} = \frac{\sum_{i=1}^{k} \Sigma_{ii}}{\sum_{i=1}^{p} \Sigma_{ii}} \geq 0.95$$
- Compressed PC representation:
  - You can typically convey most of the information in your data with far fewer dimensions $k < p$.
  - Let U_{reduce} be a $p \times k$ matrix with the first k dimensions of U.
  - Your data in the new PC space is:
$$z^{(i)} = U_{reduce}^T x^{(i)}$$
  - We can recover or reconstruct our data from the PC space back to the original space by simply doing:
$$\hat{x}^{(i)} = U_{reduce} z^{(i)}$$

13 Credits

This pdf was adapted from my notes in Machine Learning for Data Science, a class taught by Dr. Souptik Barua and Dr. Akane Sano at Rice University, Fall 2019.