**Week 1:**

**Machine learning:** To train a machine (create a function) with algorithms based on data.

**Supervised learning:** Labeled data used to train a machine. Divided into “regression” which predicts continuous values (house prices), and “classification” which predicts discrete values (malignant or benign).

**Unsupervised learning:** Train a machine without labeled data. It can find structures/clusters (photos, bacteria, news), anomalies (computer center) and reduce dimensionality (usually with PCA) (centimeter, millimeter).

**Cost function:** Fancy mean of predicted and actual value. Represents how distant we are from a perfect function.

**Gradient descent:** Function that tries to minimize the cost function (tries to find the global optimum) by continuously adapting the parameters. Higher α (step size/learning rate) finds global optimum faster but may diverge. If cost function ever increases by plotting gradient descent number of iterations by cost function, probably you should decrease α.week 2

**Week 2:**

**Linear regression:** Linear supervised algorithm to predict continuous values.

**Feature normalization:** Equation that transforms the features into the same range (terrain size, bathrooms). Making training faster. Main techniques are feature scaling and mean normalization.

**Polynomial regression:** Same as linear regression but have polynomial features.

**Normal equation:** Similar to gradient descent but doesn’t need feature scaling, to choose alpha, nor to iterate, but is slower when n is very large (> 10,000).

**Octave tutorial**

**Week 3:**

**Logistic regression:** Linear supervised learning algorithm to predict discrete values. It outputs a probability.

**Sigmoid function:** Function that squeezes a value to a 0 to 1 range.

**Decision boundary:** Line that tries to separate the 0 or 1 ground truth values. Created by the hypothesis function.

**Advanced optimization:** Similar to gradient descent but more sophisticated.

**Multiclass classification:** Method used to predict multiple classes instead of binary classes.

**Regularization:** Algorithm to create a smoother hypothesis function. Higher λ (regularization parameter) creates a smoother function and helps to solve high variance (overfitting) and vice-versa.

**Week 4:**

**Neural network:** Non-linear supervised learning algorithm that is usually used to predict discrete values. It is slower compared with logistic regression, can create more complex functions (non-linear functions). Small neural networks may underfit, big neural networks may overfit.

**Week 5:**

**Backpropagation:** Similar to gradient descent but used in neural networks.

**Gradient checking:** Function to check if backpropagation is working correctly.

**Week 6:**

**Deciding what to try next:**

To solve high variance:

* Increase λ.
* Reduce some features.
* Get more data.

To solve high bias:

* Reduce λ.
* Add new features.
* Add polynomial features.

**Evaluating a hypothesis:** Separate your data into 60% for training set, 20% for cross-validation (cv) set and 20% for test set. Use the training set to create functions with different polynomial degrees. Evaluate the function on the cross-validation set, picking the one with least error. The final accuracy is given by using the function chosen by the cross-validation set on the test set.

**Diagnosing bias vs. variance:** Method used to select polynomial degree and λ (complete steps on week 6). Plot cost from both the training set and cv cost by the polynomial degree (and another with λ, what comes next is the same for both). High bias problem if both training set cost and cv error are high and similar. High variance problem if training set cost is much lower than cv cost. Just right if both costs are small and similar.

**Learning curves:** Method used to know if more data is going to help. Plot cost from both the training set and cv cost by number of examples. High variance if their costs are not similar (training set cost is still somewhat bigger than cv cost), so more data is likely to help.

**Error analysis:** Recommended approach to solve machine learning problems:

1. Start with a simple algorithm, implement it quickly, and test it early.
2. Plot learning curves to decide if more data, more features… may help.
3. Error analysis: manually examine the errors in the cv set to try to find a trend.

**Error metrics for skewed classes:** Precision, recall, F score and accuracy. Those metrics evaluate the truth table.

**Trading off precision and recall:** Increasing TP confidence (higher precision) reduces TN confidence. Whereas increasing TN confidence (higher recall) reduces TP confidence. Use cv for those measures.

**Week 7:**

**Support vector machine (SVM):** Non-linear supervised learning (if using Kernels) algorithm which usually predicts discrete values. It doesn’t output probability, it outputs 1 or 0 (discriminant function). .

**Large margin intuition:** SVM creates the largest margin possible between the decision boundary and predicted values.

**Kernels:** Allows SVM to create non-linear functions.

**Using a SVM:** In SVM you must choose C and the kernel (similarity function). Gaussian must also choose the σ2. Higher σ2 helps to solve high variance and vice-versa. Perform feature scaling before using the Gaussian kernel. Examples for when to use:

* n = 10,000, m = 10 to 1,000, use logistic regression or SVM without kernel.
* n = 1 to 1000, m = 10 to 10,000, use SVM with Gaussian kernel.
* n = 1 to 1000, m = 50,000+, create more features then use logistic regression or SVM without kernel.
* Neural network would likely work well for those settings but may be slower to train.

**Week 8:**

**Clustering:** Type of unsupervised learning (photos, bacteria, news).

**K-Means:** Unsupervised learning algorithm used in clustering. K = number of clusters.

**Random initialization:** K-means can get stuck on local optima. So, run K-means a few times using random initialization for its centroids.

**Choosing the number of clusters:** Plotting cost by K may help (elbow method). Use hierarchical clustering to estimate K. Analyze your problem and estimate K (bacteria).

**Dimensionality reduction:** Method to reduce your data set dimensions (k), to better visualize it, or speed up and use less memory by removing the most insignificant features or remove redundant features (centimeter, millimeter).

**Principal component analysis (PCA):** Algorithm to apply dimensionality reduction. Apply feature normalization before using PCA.

**Choosing the number of principal components:** If you want to visualize your data, reduce it to 2 or 3 dimensions. If you want to speed up and use less memory, try different k values and choose k to be the smallest value so that at least 99% of the variance is retained.

**Week 9:**

**Anomaly detection:** Some supervised learning algorithm used to detect anomalies (computer center). Higher ϵ predicts more anomalous detection.

**Developing and evaluating an anomaly detection system:** Separate your data into 60% for training set, 20% for cv set and 20% for test set. Use the training set to create the functions with different ϵ and don’t use the anomalous labeled examples in the training set. Evaluate the function on the cross-validation set, picking the one with least error. The final accuracy is given by using the function chosen by the cross-validation set on the test set. Use precision, recall and F score to evaluate the model.

**Anomaly detection vs. supervised learning:** Use anomaly detection when your data is skewed, e.g., 99% of your ground truth is positive and 1% is negative. And when future anomalies may look nothing like any of the anomalous examples in the data set.

**Choosing what features to use:** Try some transformations to transform your data into a Gaussian curve. If model is wrongly predicting anomalies, you could look at those examples and try to create some feature (CPU load/network traffic) that would best describe the anomaly.

**Multivariate gaussian distribution:** Function that can be applied in anomaly detection algorithm making the algorithm capable of choosing features automatically (simple Gaussian distribution can’t). Use multivariate Gaussian distribution (sometimes better) instead of the Gaussian distribution (cheaper) when m ≥ 10n.

**Recommender system:** Algorithm to produce user recommendations (movies, products).

**Collaborative filtering:** A type of recommender system thatuses other users input to do a recommendation to another user. Improves user classification and then movie classification in a loop.

**Implementation detail: mean normalization:** Use a mean normalization term so that users that have never rated anything can have a recommendation for them as well.

**Week 10:**

**Learning with large datasets:** Traditional methods are too inefficient to work with big data (e.g. m > 100,000,000); some algorithms adjustments must be done to speed up the process.

**Other gradient descent algorithms:** The gradient descent that we have being using (batch gradient descent) which uses every training example (m) in each of its gradient descent step can be adapted to mini-batch (smaller batch, typically m from 2 to 100, for every gradient descent step) or stochastic (just one m for every gradient descent step) gradient descent. One way to get better performance out of stochastic gradient descent is to gradually decreases α or you may have to run it 1 to 10 times to find a good model.

**Online learning:** Method to be used on websites with massive data flow. We use each user’s input data to slightly improve our method and then discard his input data.

**Map reduce and data parallelism:** Method to compute big data by connecting computers and dividing the task among them. For example, pick 10 computers and make each of them compute 10% of the data (through, for example, batch gradient descent) and then return the output to a final computer which will glue those computations together.

**Week 11:**

**Ways to get more data:**

* Artificially produce more data by applying distortions to existing data.
* Collect/label the data yourself.
* Using human resources, “crowd source”, like Amazon Mechanical Turk.

**Pipeline:** The output of one element is the input of the next one. Considering the photo optical character recognition problem, we may have as our pipeline: image → text detection → character segmentation → character recognition.

**Ceiling analysis:** Method to explore the best way to improve your algorithm by improving the part of the pipeline that would more significantly improve the overall model. It gives us information about which components, if improved, are most likely to have a significant impact on the performance of the final system. For example, if our overall system has 72% accuracy, increasing to 89% on the text detection step, to 90% on the character segmentation step and finally to 100% (obviously) on the character recognition step. That means that even when you give the algorithm perfect character segmentation, your performance goes up by only 1%, whereas when you give the algorithm perfect text detection it goes up by 17%. Therefore, in this case, if you try to have a perfect character segmentation part, your overall system will only increase by 1% of its accuracy, whereas if you try to have a perfect text detection, the accuracy would increase by 17%, making text detection a worth option to work on.