Hands-on Machine Learning - Chapters 1-9 Essential Concepts

This is a basic list of essential conceptual information from each of the first nine chapters. It's not exhaustive by any means and isn't meant to be. Consult the text for additional details.

Chapter 1: The Machine Learning Landscape

(The majority of the ideas introduced in this chapter are covered in more detail in subsequent chapters and will be mentioned in those respective chapters instead)

Machine learning involves making machines better at a task by learning from data rather than explicit code.

Main Challenges of Machine Learning

- Insufficient quantity of data
- Nonrepresentative training data (sampling noise and bias)
- Poor quality data
- Irrelevant features
- Overfitting/Underfitting the data

Chapter 2: End-to-End Machine Learning Project

Steps in a machine learning project

1. Look at the big picture

- Frame the problem

2. Get the data.

- Create a workspace
- Multiple data sources, from universities, Kaggle, Wikipedia, government databases, Amazon, Google, etc.

3. Discover and visualize the data to gain insights.

- Frame the problem
- Select a performance measure (accuracy, rmse, etc.)
- Check the assumptions

4. Prepare the data for Machine Learning algorithms.
- Look at the data structure
- Create a test set before data cleaning
- Visualize data
- Look for correlations
- Create attribute combinations / feature engineering
- Data cleaning
- Encode, transform, and scale the data as needed
- Create transformation pipelines

5. Select and train a model.
- Evaluate on a training set
- Use cross-validation

6. Fine-tune your model.
- GridSearchCV and Randomized search CV
- Ensemble methods
- Analyze best models and their errors
- Evaluate on test set

7. Present your solution.
- Highlight what worked and didn't work, what assumptions were made, and state the system's limitations.

8. Launch, monitor, and maintain your system.
- Monitor live performance
- Evaluate input data quality
- Retrain models regularly using fresh data

Chapter 3: Classification

Classification - supervised learning approach that predicts the class of given data points.

Binary vs Multiclass

- Predicting two classes versus predicting multiple classes.
- Binary classifiers include logistic regression and SVM. Multiclass classifiers include random forest and naive Bayes.

MNIST - classic dataset consisting of 70,000 images of handwritten digits (0-9).
Confusion matrix - a 4x4 grid where each row represents an actual class and each column represents a predicted class. Consists of true positives, false positives, true negatives, and false negatives.

Performance measures

- Precision = TP / (TP + FP)   accuracy of positive predictions
- Recall (aka sensitivity or true positive rate)= TP / (TP + FN)   ratio of positive instances correctly detected
- F1 = (precision x recall) / (precision + recall)
- TP = true positive, FP = false positive, TN = true negative, FN = false negative

Precision/Recall tradeoff

- Inverse relationship - increasing the decision threshold increases precision and decreases recall and vice versa.

ROC (receiver operating characteristic) curve - plots the true positive rate or recall (TPR) against the false positive rate (FPR).

- A perfect classifier will have an area under the curve (AUC) of 1, whereas a random classifier will have 0.5.
- FPR = 1 - TNR (true negative rate or specificity).
- Thus, ROC measures recall versus 1 - specificity.
- In Sci-Kit learn, ROC can be accessed using the roc_curve() function.

Multilabel classification - a system that outputs multiple binary tags.

Multioutput classification - generalization of multilabel classification where each label can have more than two labels.

Chapter 4: Training Models

Linear regression works well for linear datasets, but nonlinear datasets require polynomial regression models. However, the latter are prone to overfitting

RMSE (root mean squared error) and MSE (mean squared error) are common performance measures for regression models. Good models minimize these values.

Gradient descent adjusts parameters in order to minimize a cost function.

- Number of steps are determined by the learning rate hyperparameter. Too many steps may lead to a slow learning time while too few steps may cause a model to fail in finding the ideal minimum.
- GD is guaranteed to approach arbitrarily close to a global minimum for linear regression models because they are convex functions.
- Stochastic gradient descent is faster than batch gradient descent since it computes the gradient at every step using only a random instance rather than the whole dataset.

Bias/Variance tradeoff

- High bias models are likely to underfit data because they lack sensitivity.
- High variance models are prone to overfitting the data due to excessive sensitivity.
- Reducing bias increases variance and vice versa.

Lasso regression (L1) - regularization method that eliminates the weights of the least important features. Automatically performs feature selection and outputs a sparse model.

Ridge regression (L2) - minimizes model weights to fit data using a regularization term. Should only be added to the cost function during training.

Logistic regression - binary classification model that outputs a logistic between 0 and 1.

Softmax regression - supports multiple classes directly without having to combine multiple binary classifiers.

Chapter 5: Support Vector Machines

SVM - versatile ML models that can be used for linear and nonlinear regression, classification, regression and outlier detection.

Linear SVM - works best with data that is linearly separable. Attempts to draw as large as margin as possible which is determined by the support vectors located at the edge of each collection of instances.

- In Scikit_Learn's SVM classes, the C hyperparameter can be used to control the margins. Smaller C values lead to a wider margin and more margin violations. A larger C gives smaller margin but can lead to less generalizable models.

Nonlinear SVM - adding features to make a dataset linearly separable.

- Adding too few features doesn't allow a model to handle complex datasets, but adding to many can cause a model to be too slow.
- The kernel trick allows for the addition of multiple polynomial features without actually adding them.

Gaussian RBF Kernel - adding features computed using a similarity function that measures how much each instance resembles a particular landmark.

SVM Regression - rather than trying to fit the largest possible margin between two classes (as with SVM classification), SVM regression attempts to fit as many instances as possible on the margin while limiting margin violations.

Chapter 6: Decision Trees

Decision trees can be used for both regression and classification models. They start with a root node that branches into further nodes based on questions asked at each stage. Leaf nodes form when no further questions can be asked and the model converges to a conclusion on a certain path.

In Sci-Kit Learn, DecisionTreeClassifier and DecisionTreeRegressor are the respective classification and regression models.

- predict_proba() returns the probability of a particular instance being part of a certain class. predict() returns the class prediction.

White box vs black box models - white box models are easy to interpret because their decisions are clearly laid out, as with decision tree models. Black box models, such as random forests, are often more difficult to interpret because it is often difficult to understand how the models reach their decisions.

CART (classification and regression tree) algorithm - splits a training set into two subsets using a single feature and a threshold. Ideal feature and threshold are chosen by finding the pair that produces the purest subsets (weighted by size). Labeled a greedy algorithm - greedily searches for an optimum split at the top level and repeats the process at subsequent level.

Regularization hyperparameters

- Decision trees are prone to overfitting so regularization is important.
- min_samples_split (minimum nodes before split), min_samples_leaf, max_leaf_nodes, and max_features, are useful in decreasing overfitting.

Decision trees are very sensitive to small variations in the training data. Random forests are useful in controlling this instability by averaging predictions over many trees.

Chapter 7: Ensemble Learning and Random Forests

Voting classifiers - aggregation of multiple classifiers that predict a class based on the most votes.

- Hard voting - prediction based on majority vote
- Soft voting - prediction based on highest class probability averaged across all classifiers

Bagging (bootstrap aggregating) - sampling performed with replacement.
Pasting - sampling performed without replacement.

- BaggingClassifier uses bagging by default. To use pasting, set hyperparameter bootstrap=False.

Random forest - an ensemble of decision trees, generally trained via bagging and with max_samples set to size of the training set.

- RandomForestClassifier = BaggingClassifier + DecisionTreeClassifier (RandomForestRegression works similarly).

- Random forests make the measurement of feature importances relatively easy with the feature_importances_ variable.
- In comparison to decision trees, random forests have greater tree diversity which leads to higher bias but lower variance.

Boosting - ensemble method combining several weak learners into a strong learner.

- AdaBoost - each subsequent classifier is trained on the updated weights of the previous classifier.
- Gradient boosting - fits each new predictor to the residual errors made by the previous predictor.
- XGBoost- optimized gradient boosting, often used in ML competitions.

Stacking - training a model to perform predictor aggregation.

Chapter 8: Dimensionality Reduction

Curse of dimensionality - while having more features may seem like a beneficial, higher dimension datasets are at greater risk of being sparse. As such, the more dimensions that a training set has, the more likely it will be overfitted.

Main approaches to dimensionality reduction

- Projection - taking higher dimensional data and expressing it to a lower dimensional subspace.
- Manifold learning - based on manifold hypothesis that holds that most real-world datasets lie close to a lower-dimensional manifold.

PCA (Principal Component Analysis) - identifies hyperplane closest to data(accounts for the greatest amount of variance) and projects data onto it. Reduces the number of components and can compress the size of a dataset.

- In Scikit-Learn, components can be controlled using n_components hyperparameter, either inserting a whole number for actual components or a number between 0 and 1 for ratio of variance.
- Explained_variance_ratio_ variable lists proportion of variance along the axis of each principal component.

Kernel PCA - allows for complex nonlinear projections for dimensionality reduction. Useful for preserving clusters after projection and unrolling datasets lying close to a twisted manifold.

LLE (Local Linear Embedding) - manifold learning technique that works by first measuring how each training instance linearly relates to its closest neighbors and then searching for low dimensional representation that best allows for preservation of the local dimensions of the dataset.

Chapter 9: Unsupervised Learning Techniques

Clustering - identifying instances that are similar to each other and assigning them to groups of similar instances (i.e., clusters).

- Centroid - center of a cluster.
- Practical application of clustering includes fraud detection, initial data analysis, dimensionality reduction, outlier/anomaly detection, search engines, image segmentation, and semi-supervised learning.

K-Means - popular clustering algorithm that works by placing centroids (either randomly or in specific locations), labeling instances in relation to each centroid, updating centroid locations, relabeling instances, etc., until centroids stop moving.

- Finding the optimal number of clusters can involve either using the elbow method (measuring where inertia drop rate slows down) or searching for the highest silhouette score. The latter is the preferred and more reliable method.

Limits of K-Means

- Algorithm often needs to be run several times to avoid suboptimal solutions.
- Doesn't work well when clusters have varying sizes/densities or non-spherical shapes.

DBSCAN - algorithm that defines clusters as continuous regions of high density.

- ε-neighborhood - for each instance, the number of other instances located ε distance away from it.
- Core instance - an instance's designation if it has at least min_samples instances in its ε-neighborhood.

Other clustering algorithms

- Agglomerative clustering
- Birch
- Mean-shift
- Affinity propagation
- Spectral clustering

Gaussian mixtures - probabilistic model that assumes that instances are generated from mixtures of several Gaussian distributions with unknown parameters. All instances associated with a particular Gaussian distribution form a cluster.

- Gaussian mixture models work well with ellipsoid-shaped clusters, but not as well with dataset with different cluster shapes.
- Finding the ideal number of clusters involves finding the model that minimizes either the Bayesian information criterion (BIC) or Akaike information criterion (AIC). BIC tends to have fewer parameters but doesn't fit data quite as well, especially for larger datasets. Call using either the bic() or aic() methods for the GaussianMixture algorithm.

Likelihood vs Probability - Probability refers to how plausible a future outcome is knowing some given parameter values. Likelihood describes how plausible a set of parameter values after an outcome is known.

Bayesian Gaussian mixture models - capable of assigning weights close to, or equal to, zero for unnecessary clusters.

- Use by setting n_components In the BayesianGaussianMixture class to a number higher than the assumed optimal number of clusters.

Anomaly and novelty detection algorithms

- Fast-MCD
- Isolation forest
- Local outlier factor
- One-class SVM