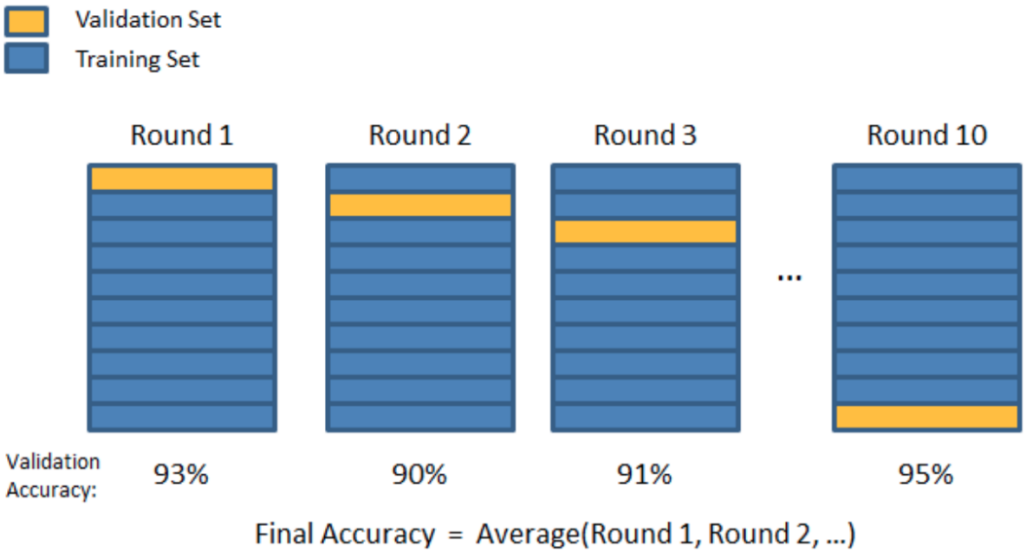
Statistics and ML In General

Cross Validation

Cross-validation is a technique to evaluate predictive models by partitioning the original sample into a training set to train the model, and a validation set to evaluate it. For example, a k-fold cross validation divides the data into k folds (or partitions), trains on each k-1 fold, and evaluate on the remaining 1 fold. This results to k models/evaluations, which can be averaged to get a overall model performance.



Feature Importance

- In linear models, feature importance can be calculated by the scale of the coefficients
- In tree-based methods (such as random forest), important features are likely to appear closer to the root of the tree. We can get a feature's importance for random forest by computing the averaging depth at which it appears across all trees in the forest.

back to top

Mean Squared Error vs. Mean Absolute Error

- **Similarity**: both measure the average model prediction error; range from 0 to infinity; the lower the better
- Mean Squared Error (MSE) gives higher weights to large error (e.g., being off by 10 just MORE THAN TWICE as bad as being off by 5), whereas Mean Absolute Error (MAE) assign equal weights (being off by 10 is just twice as bad as being off by 5)
- MSE is continuously differentiable, MAE is not (where y_pred == y_true)

back to top

L1 vs L2 regularization

- **Similarity**: both L1 and L2 regularization **prevent overfitting** by shrinking (imposing a penalty) on the coefficients
- **Difference**: L2 (Ridge) shrinks all the coefficient by the same proportions but eliminates none, while L1 (Lasso) can shrink some coefficients to zero, performing variable selection.
- **Which to choose**: If all the features are correlated with the label, ridge outperforms lasso, as the coefficients are never zero in ridge. If only a subset of features are correlated with the label, lasso outperforms ridge as in lasso model some coefficient can be shrunken to zero.
- In Graph (a), the black square represents the feasible region of the L1 regularization while graph (b) represents the feasible region for L2 regularization. The contours in the plots represent different loss values (for the unconstrained regression model). The feasible point that minimizes the loss is more likely to happen on the coordinates on graph (a) than on graph (b) since graph (a) is more **angular**. This effect amplifies when your number of coefficients increases, i.e. from 2 to 200. The implication of this is that the L1 regularization gives you sparse estimates. Namely, in a high dimensional space, you got mostly zeros and a small number of non-zero coefficients.

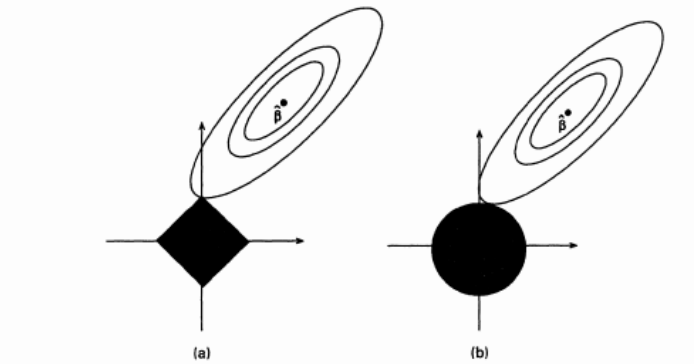


Fig. 2. Estimation picture for (a) the lasso and (b) ridge regression

back to top

Correlation vs Covariance

- Both determine the relationship and measure the dependency between two random variables
- Correlation is when the change in one item may result in the change in the another item, while covariance is when two items vary together (joint variability)
- Covariance is nothing but a measure of correlation. On the contrary, correlation refers to the scaled form of covariance
- Range: correlation is between -1 and +1, while covariance lies between negative infinity and infinity.

back to top

Would adding more data address underfitting

Underfitting happens when a model is not complex enough to learn well from the data. It is the problem of model rather than data size. So a potential way to address underfitting is to increase the model complexity (e.g., to add higher order coefficients for linear model, increase depth for tree-based methods, add more layers / number of neurons for neural networks etc.)

back to top

Activation Function

For neural networks

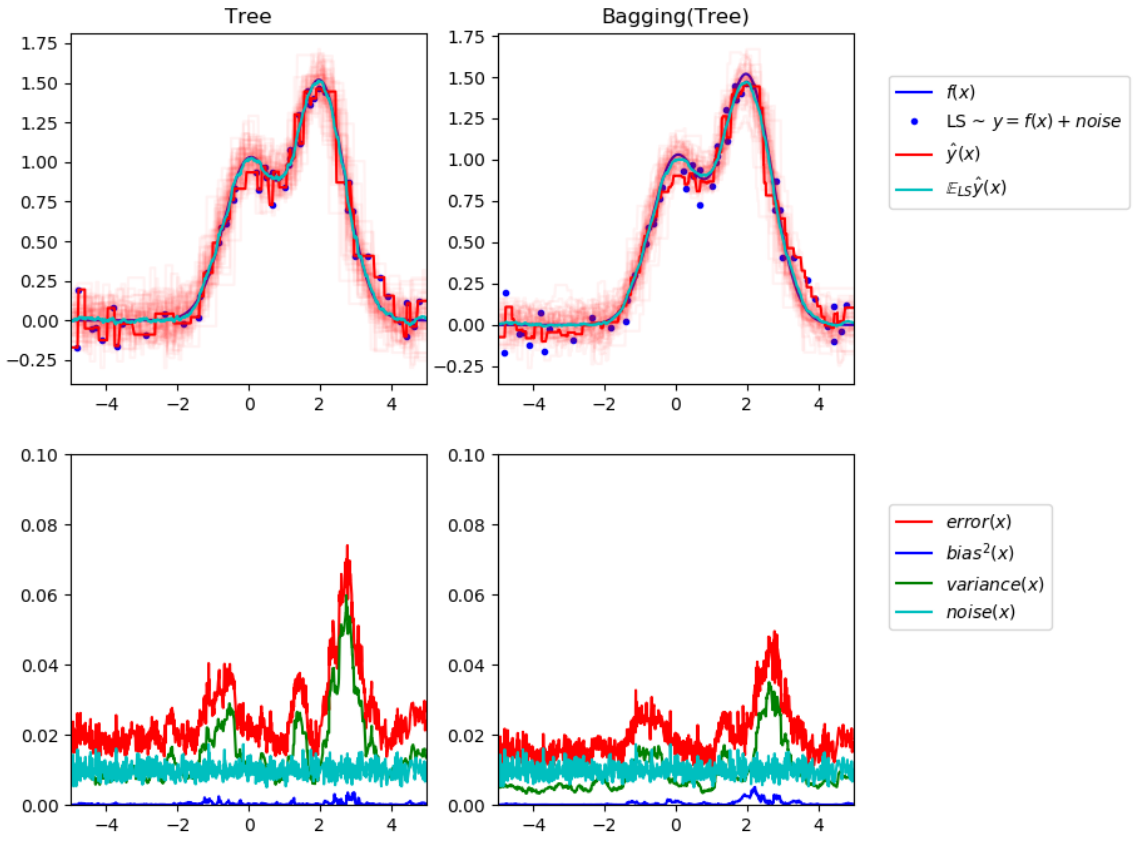
- Non-linearity: ReLU is often used. Use Leaky ReLU (a small positive gradient for negative input, say, `y = 0.01x` when x < 0) to address dead ReLU issue
- Multi-class: softmax
- Binary: sigmoid
- Regression: linear

back to top

Bagging

To address overfitting, we can use an ensemble method called bagging (bootstrap aggregating), which reduces the variance of the meta learning algorithm. Bagging can be applied to decision tree or other algorithms.

Here is a great illustration of a single estimator vs. bagging.



- Bagging is when samlping is performed *with* replacement. When sampling is performed *without* replacement, it's called pasting.
- Bagging is popular due to its boost for performance, but also due to that individual learners can be trained in parallel and scale well
- Ensemble methods work best when the learners are as independent from one another as possible
- Voting: soft voting (predict probability and average over all individual learners) often works better than hard voting
- out-of-bag instances can act validation set for bagging

back to top

Stacking

- Instead of using trivial functions (such as hard voting) to aggregate the predictions from individual learners, train a model to perform this aggregation
- First split the training set into two subsets: the first subset is used to train the learners in the first layer
- Next the first layer learners are used to make predictions (meta features) on the second subset, and those predictions are used to train another models (to obtain the weigts of different learners) in the second layer
- We can train multiple models in the second layer, but this entails subsetting the original dataset into 3 parts

back to top

Generative vs discriminative

- Discriminative algorithms model *p(y|x; w)*, that is, given the dataset and learned parameter, what is the probability of y belonging to a specific class. A discriminative algorithm doesn't care about how the data was generated, it simply

categorizes a given example

- Generative algorithms try to model *p(x|y)*, that is, the distribution of features given that it belongs to a certain class. A generative algorithm models how the data was generated.

> Given a training set, an algorithm like logistic regression or the perceptron algorithm (basically) tries to find a straight line—that is, a decision boundary—that separates the elephants and dogs. Then, to classify a new animal as either an elephant or a dog, it checks on which side of the decision boundary it falls, and makes its prediction accordingly.

> Here's a different approach. First, looking at elephants, we can build a model of what elephants look like. Then, looking at dogs, we can build a separate model of what dogs look like. Finally, to classify a new animal, we can match the new animal against the elephant model, and match it against the dog model, to see whether the new animal looks more like the elephants or more like the dogs we had seen in the training set.

back to top

Parametric vs Nonparametric

- A learning model that summarizes data with a set of parameters of fixed size (independent of the number of training examples) is called a parametric model.
- A model where the number of parameters is not determined prior to training. Nonparametric does not mean that they have no parameters. On the contrary, nonparametric models (can) become more and more complex with an increasing amount of data.

back to top

Recommender System

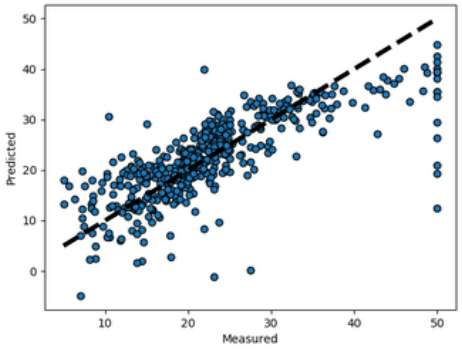
- I put recommend system here since technically it falls neither under supervised nor unsupervised learning
- A recommender system seeks to predict the 'rating' or 'preference' a user would give to items and then recommend items accordingly
- Content based recommender systems recommends items similar to those a given user has liked in the past, based on either explicit (ratings, like/dislike button) or implicit (viewed/finished an article) feedbacks. Content based recommenders work solely with the past interactions of a given user and do not take other users into consideration.
- Collaborative filtering is based on past interactions of the whole user base. There are two Collaborative filtering approaches: **item-based** or **user-based**
  - item-based: for user u, a score for an unrated item is produced by combining the ratings of users similar to u.
  - user-based: a rating (u, i) is produced by looking at the set of items similar to i (interaction similarity), then the ratings by u of similar items are combined into a predicted rating
- In recommender systems traditionally matrix factorization methods are used, although we recently there are also deep learning based methods
- Cold start and sparse matrix can be issues for recommender systems
- Widely used in movies, news, research articles, products, social tags, music, etc.

back to top

Supervised Learning

Linear regression

- How to learn the parameter: minimize the cost function
- How to minimize cost function: gradient descent
- Regularization:
  - L1 (Lasso): can shrink certain coef to zero, thus performing feature selection
  - L2 (Ridge): shrink all coef with the same proportion; almost always outperforms L1
  - Elastic Net: combined L1 and L2 priors as regularizer
- Assumes linear relationship between features and the label
- Can add polynomial and interaction features to add non-linearity



back to top

Logistic regression

- Generalized linear model (GLM) for binary classification problems
- Apply the sigmoid function to the output of linear models, squeezing the target to range [0, 1]
- Threshold to make prediction: usually if the output > .5, prediction 1; otherwise prediction 0
- A special case of softmax function, which deals with multi-class problems

back to top

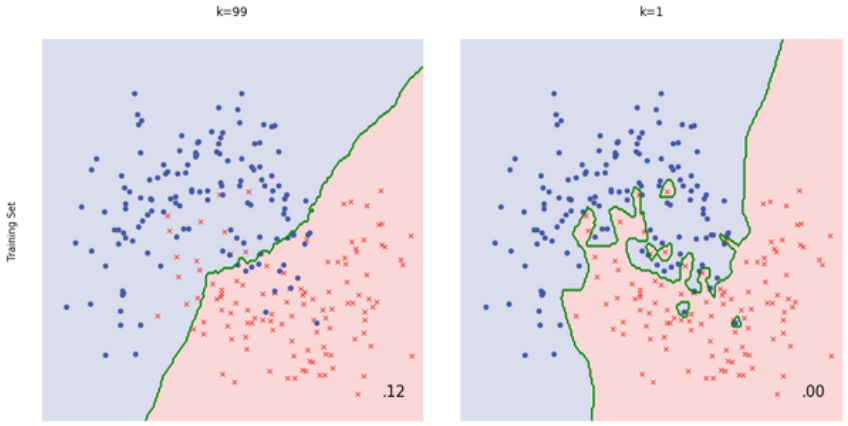
Naive Bayes

- Naive Bayes (NB) is a supervised learning algorithm based on applying Bayes' theorem
- It is called naive because it builds the naive assumption that each feature are independent of each other
- NB can make different assumptions (i.e., data distributions, such as Gaussian, Multinomial, Bernoulli)
- Despite the over-simplified assumptions, NB classifier works quite well in real-world applications, especially for text classification (e.g., spam filtering)
- NB can be extremely fast compared to more sophisticated methods

back to top

KNN

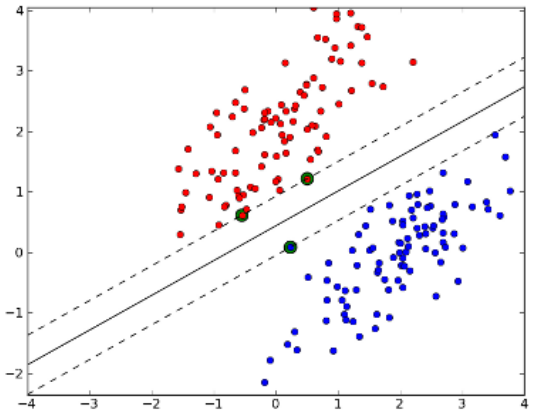
- Given a data point, we compute the K nearest data points (neighbors) using certain distance metric (e.g., Euclidean metric). For classification, we take the majority label of neighbors; for regression, we take the mean of the label values.
- Note for KNN we don't train a model; we simply compute during inference time. This can be computationally expensive since each of the test example need to be compared with every training example to see how close they are.
- There are approximation methods can have faster inference time by partitioning the training data into regions (e.g., annoy)
- When K equals 1 or other small number the model is prone to overfitting (high variance), while when K equals number of data points or other large number the model is prone to underfitting (high bias)



back to top

SVM

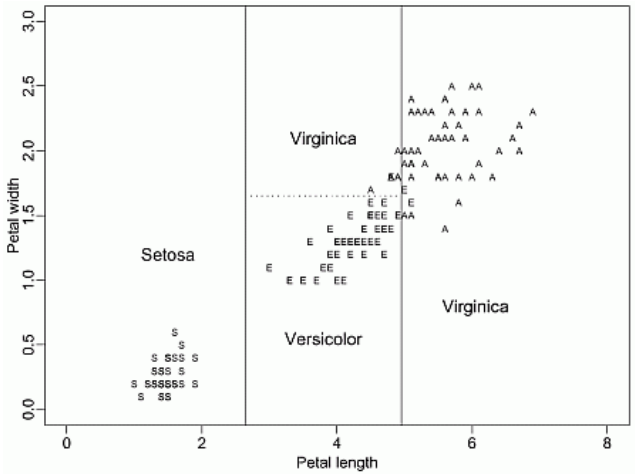
- Can perform linear, nonlinear, or outlier detection (unsupervised)
- Large margin classifier: using SVM we not only have a decision boundary, but want the boundary to be as far from the closest training point as possible
- The closest training examples are called support vectors, since they are the points based on which the decision boundary is drawn
- SVMs are sensitive to feature scaling



back to top

Decision tree

- Non-parametric, supervised learning algorithms
- Given the training data, a decision tree algorithm divides the feature space into regions. For inference, we first see which region does the test data point fall in, and take the mean label values (regression) or the majority label value (classification).
- **Construction**: top-down, chooses a variable to split the data such that the target variables within each region are as homogeneous as possible. Two common metrics: gini impurity or information gain, won't matter much in practice.
- Advantage: simply to understand & interpret, mirrors human decision making
- Disadvantage:
  - can overfit easily (and generalize poorly) if we don't limit the depth of the tree
  - can be non-robust: A small change in the training data can lead to a totally different tree
  - instability: sensitive to training set rotation due to its orthogonal decision boundaries



back to top

Random forest

Random forest improves bagging further by adding some randomness. In random forest, only a subset of features are selected at random to construct a tree (while often not subsample instances). The benefit is that random forest **decorrelates** the trees.

For example, suppose we have a dataset. There is one very predicative feature, and a couple of moderately predicative features. In bagging trees, most of the trees will use this very predicative feature in the top split, and therefore making most of the trees look similar, **and highly correlated**. Averaging many highly correlated results won't lead to a large reduction in variance compared with uncorrelated results. In random forest for each split we only consider a subset of the features and therefore reduce the variance even further by introducing more uncorrelated trees.

I wrote a notebook to illustrate this point.

In practice, tuning random forest entails having a large number of trees (the more the better, but always consider computation constraint). Also, `min_samples_leaf` (The minimum number of samples at the leaf node)to control the tree size and overfitting. Always cross validate the parameters.

back to top

Boosting Tree

How it works

Boosting builds on weak learners, and in an iterative fashion. In each iteration, a new learner is added, while all existing learners are kept unchanged. All learners are weighted based on their performance (e.g., accuracy), and after a weak learner is added, the data are re-weighted: examples that are misclassified gain more weights, while examples that are correctly classified lose weights. Thus, future weak learners focus more on examples that previous weak learners misclassified.

Difference from random forest (RF)

- RF grows trees **in parallel**, while Boosting is sequential
- RF reduces variance, while Boosting reduces errors by reducing bias

XGBoost (Extreme Gradient Boosting)

> XGBoost uses a more regularized model formalization to control overfitting, which gives it better performance

Unsupervised Learning

- Clustering
- Principal Component Analysis
- Autoencoder
- Generative Adversarial Network

Clustering

- Clustering is a unsupervised learning algorithm that groups data in such a way that data points in the same group are more similar to each other than to those from other groups
- Similarity is usually defined using a distance measure (e.g, Euclidean, Cosine, Jaccard, etc.)
- The goal is usually to discover the underlying structure within the data (usually high dimensional)
- The most common clustering algorithm is K-means, where we define K (the number of clusters) and the algorithm iteratively finds the cluster each data point belongs to

scikit-learn implements many clustering algorithms. Below is a comparison adopted from its page.

back to top

Principal Component Analysis

- Principal Component Analysis (PCA) is a dimension reduction technique that projects the data into a lower dimensional space
- PCA uses Singular Value Decomposition (SVD), which is a matrix factorization method that decomposes a matrix into three smaller matrices (more details of SVD here)
- PCA finds top N principal components, which are dimensions along which the data vary (spread out) the most. Intuitively, the more spread out the data along a specific dimension, the more information is contained, thus the more important this dimension is for the pattern recognition of the dataset
- PCA can be used as pre-step for data visualization: reducing high dimensional data into 2D or 3D. An alternative dimensionality reduction technique is t-SNE

Autoencoder

- The aim of an autoencoder is to learn a representation (encoding) for a set of data
- An autoencoder always consists of two parts, the encoder and the decoder. The encoder would find a lower dimension representation (latent variable) of the original input, while the decoder is used to reconstruct from the lower-dimension vector such that the distance between the original and reconstruction is minimized
- Can be used for data denoising and dimensionality reduction