Early Stopping

* When validation score starts to increase with more fitting, just roll back to parameters where model had least error.

The Normal Equation (Linear Regression)

* Exact, just need to invert matrix
* Scales as O(features^2) and O(datasize)

Gradient Descent (Linear Regression)

* Walks toward minimum using gradient as guide.
* Learning rate hyperparameter affects speed/accuracy of convergence.
* No local minimum with linear data
* Without scaling, convergence will take longer
* Batch Gradient Descent
  + Compute gradient with entire dataset every iteration
* Stochastic Gradient Descent
  + Pick just one datapoint to compute gradient of iteration
  + Fast, less memory, but limited convergence
  + Use learning schedule to hone in on convergence
* Mini-batch Gradient Descent
  + Use a limited number of datapoints to compute gradient. Compromise between speed and convergence.

Polynomial Regression

* Added “features” from products of original.
* Easy to over fit. Features grow as factorial of polynomial degree.
* Check overfitting with learning curves
  + Train, validation scores vs training set size or training iteration
  + High, but similar errors? Underfitting
    - Improve features or use more complex model
  + Lower, but dissimilar errors? Overfitting
    - Add more data, add regularization,

Ridge Regression

* Regularized version of Linear Regression. Add L2 penalty for weights.
* Scaled data is important as this affects the weight penalty.
* Closed for Solution exists, but with the same tradeoff as Linear regression.

Lasso Regression

* Regulared version of linear regression with L1 penalty for weights.
* Due to L1 penalty, it tends to eliminate weights of least important features.
* However, correlated features can appear differently in the weights (split or remove one completely)

Elastic Net

* A combination of ridge and lasso regression.

Logistic Regression

* Computes weight sum of features (plus bias term) like linear regression, but output the logistic of the result.
* No exact solution, but still convex so any other optimization method (Gradient Descent, etc) is guaranteed to find the global minimum
* Regularization penalties can also be added.

Softmax Regression

* A generalization of the logistic regression to support multiple classes directly (no resorting to one vs one or one vs all as with binary classifiers).
* Ideal is that you fit logistic models for all the classes (a parameter matrix instead of a parameter vector).

Support Vector Machines Classification

* Very popular, good for complex, but small to medium sized datasets.
* Idea is to find largest “street” to separate classes.
* Hard margin (no class on wrong side of street) is often impossible and is especially susceptible to outliers.
  + Trade off between keeping street large and reducing datapoints on street
* A larger C hyperparameter leads to a smaller street but fewer datapoints on street.
* Reducing C will help correct overfitting as the street will be forced to be wider.
* LinearSVC is much faster than SVC(kernel=’linear’)
* For non-linearly-separable datasets
  + you can first add polynomial features and then run SVC.
    - The “kernel trick” save a lot of computation.
    - Use with SVC(kernel=’poly’)
  + Additionally, use similarity features or “landmarks” to transform the data.
    - It will increase feature dimensions, but make linear separating more possible

Support Vector Machine Regression

* Reverse the goal, try to make a skinny street that contains all the data.
* If non-linear, use kernel transformations.

Decision Trees

* Don’t require feature centering or scaling
* Single tree model is a “white box”, but ensembles or neural networks are black box.
* Understand node structure
  + Splits into two children based on a given feature condition (petal length>2cm)
  + Keeps track of total samples and class values as well as purity measure
  + Can use gini purity measure (slightly faster) or entropy definition (more balanced tree), but not a big difference.
  + When used for regression, leaf has estimate value instead of class
* Can return probabilities based on relative frequency in leaves.
* Node conditions are chosen to maximize the purity of children nodes.
* Complexity is O(feature \* datasize\*log(datasize))
* Hyperparameters to avoid overfitting:
  + Max\_depth, min\_samples\_split, min\_samples\_leaf
* Decision boundaries are orthogonal, so sensitive to training set rotation

Ensemble Learning

* An ensemble can give better predictions than the best estimator.
* To improve results, they should be as independent as possible (ie, use a variety of algorithms or different subsets of training data)
* Types of voting
  + Hard voting: count up the ensembles votes and choose the winner
  + Soft voting: average the predicted probabilities and choose the winner
* Ensembling strategies:
  + Bagging: Apply algorithm to many samples of training data with replacement
  + Pasting: Apply algorithm to many samples of training data without replacement
* An ensemble will generalize better even if results on training data are similar to single classifiers/regressors. Cross-validation just takes a lot of extra time.