AWS Course - The Elements of Data Science - part 3

Lesson 8 of 12

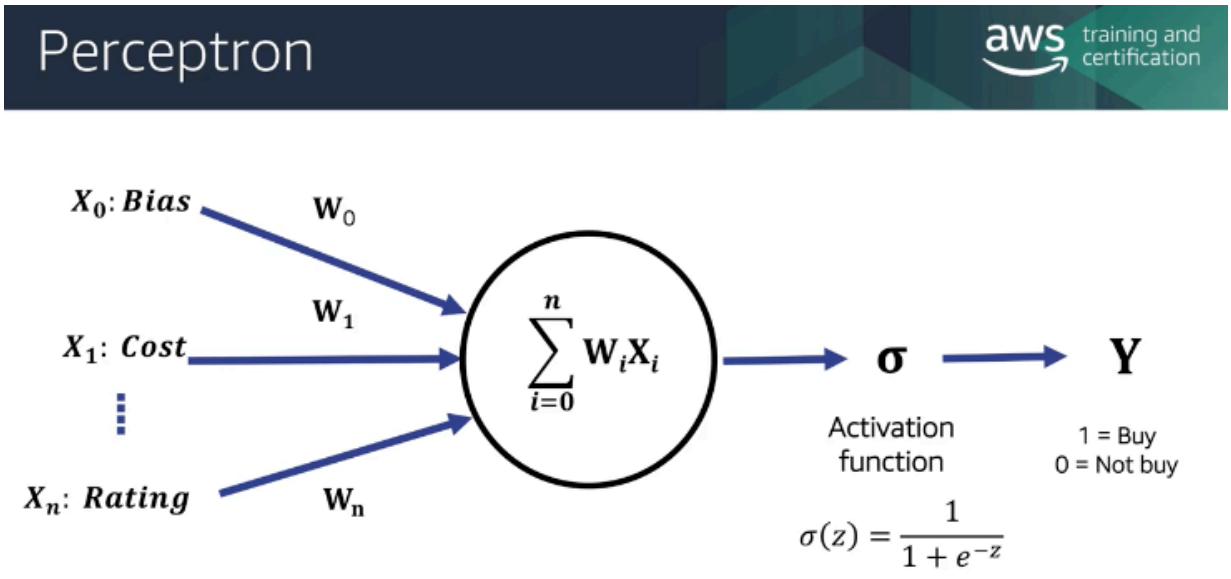
Model Training, Tuning, and Debugging

Model Training, Tuning, and Debugging

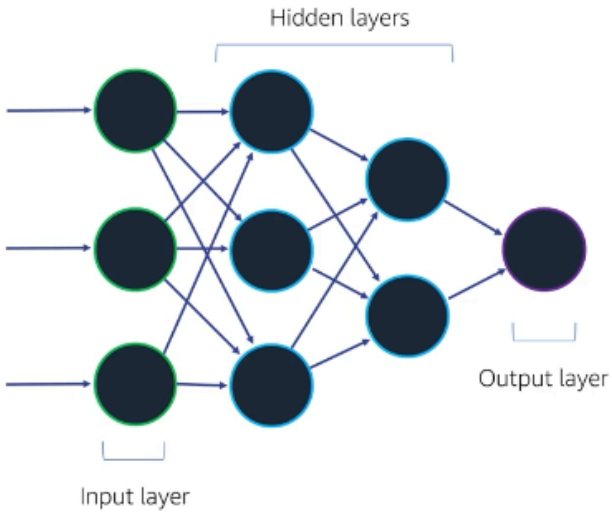
In this section of the course we discuss how you can continuously improve your machine learning models by training, tuning, and debugging them.

Supervised Learning: Neural Networks

In this exploration of neural networks we feature single-layer neural networks, convolutional neural networks, and recurrent neural networks.

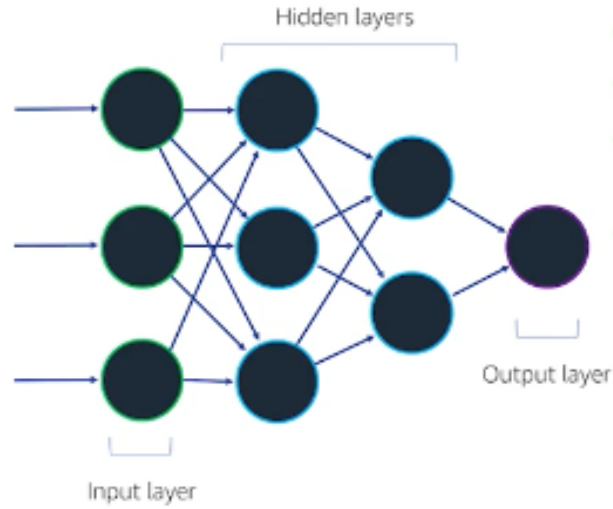


Neural networks

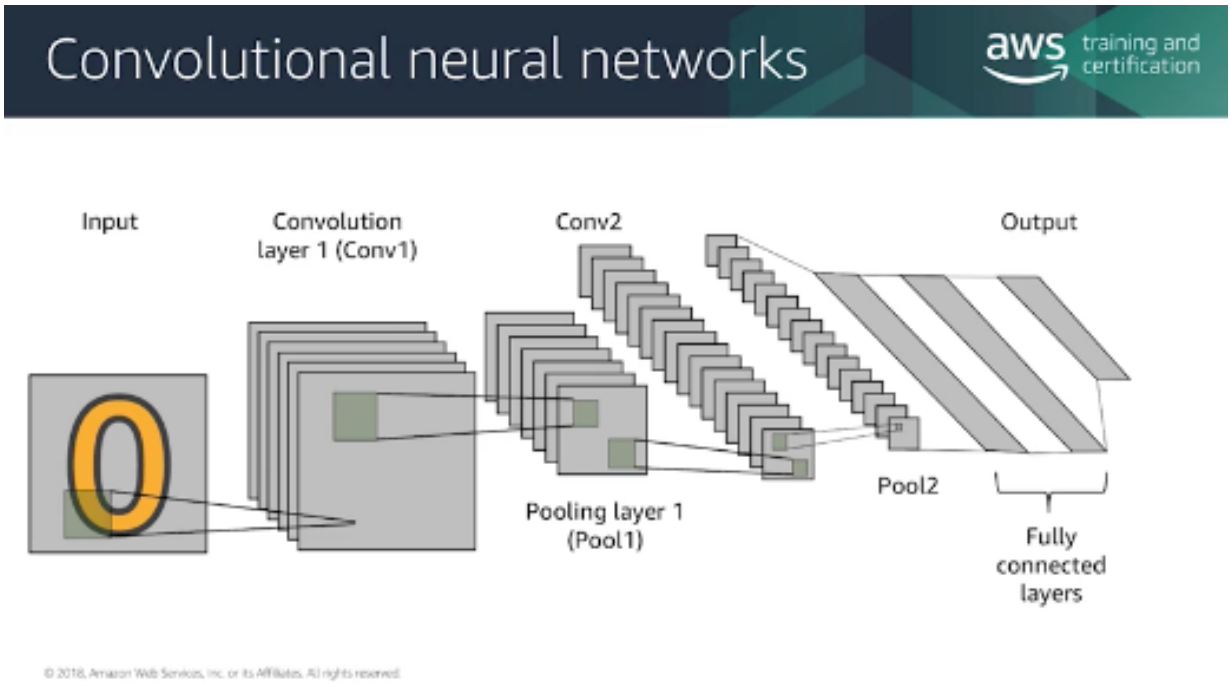


- Layers of nodes connected together
- Each node is one multivariate linear function, with an univariate nonlinear transformation
- Trained via (stochastic) gradient descent
- Can represent any non-linear function (very expressive)

Neural networks

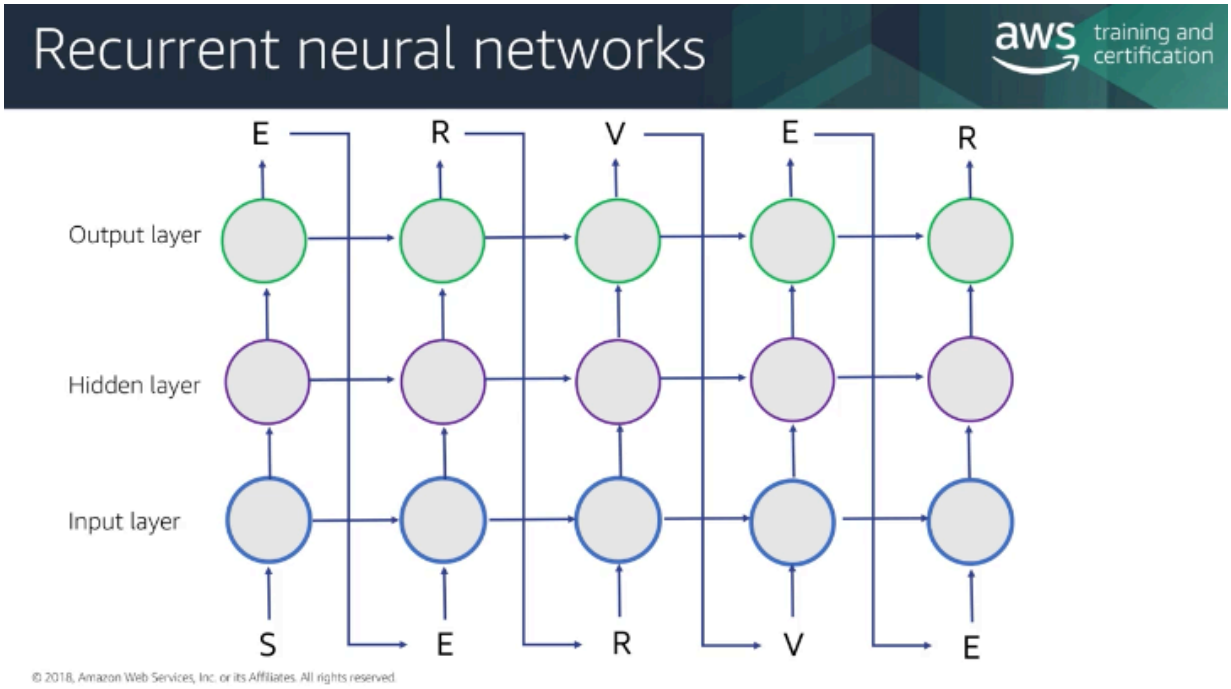


- Generally hard to interpret
- Expensive to train, fast to predict
- scikit-learn: sklearn.neural_network.MLPClassifier
- Deep learning frameworks:
  - MXNet
  - TensorFlow
  - Caffe
  - PyTorch



Input image
- use filters to build a convolution layer
- once we have an input, we may want to reduce the size => use **Pooling** process (reduce single scaler taking max or average) => **Dimension reduction process**
- in the end, we convert the Tensor into a Vector
- output is usually a category of the graph - for example a digit “0”

Recurrent neural networks for NLP, translation or time based entries:
In the below example, we just use 1 layer to represent all hidden layers
The characters have meaning as a SEQUENCE
During training process, information flow is re-used, propagated to different nodes



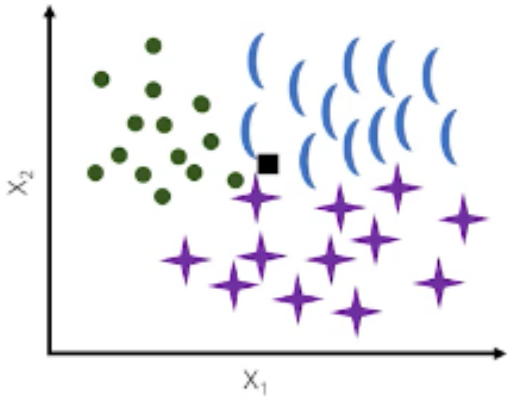
Supervised Learning: K-Nearest Neighbors

Here, we explore the K-Nearest neighbor methodology and describe how it can be applied to different use cases.

Used in classification / specify k = closest nearest neighbors
The black box is the “observation” - we want ot find the color for this new observation
K-nearest neighbors will help find it

K-Nearest Neighbors

aws training and certification

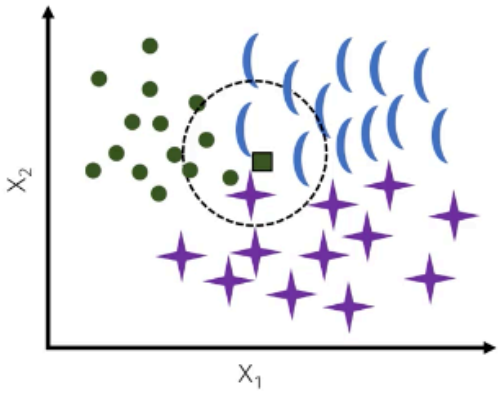


- Define a distance metric
  - Euclidean distance
  - Manhattan distance
  - Any vector norm
- Choose the number of **k** neighbors
- Find the **k** nearest neighbors of the new observation that we want to classify

Let's say k = 6 => we find the 6 nearest neighbors
In that case, the observation is green
usually we use the square root of the samples

K-Nearest Neighbors

aws training and certification



- Assign the class label by majority vote
- Important to find the right **k**
  - Commonly use $k = \frac{\sqrt{N}}{2}$

    where N = number of samples

To note we have not fiit any model => k-nearest neighbors is non-parametric (no specific function),
=> lazy loading - the model is the result of memorizing the training data => can be expensive (lots of memory usage)

K-Nearest Neighbors

aws training and certification

- Non-parametric, instance-based, lazy
  - Non-parametric: Model is not defined by fixed set of parameters
  - Instance-based or lazy learning: Model is the result of effectively memorizing training data
- Requires keeping the original data set
- Space complexity and prediction-time complexity grow with size of training data
- Suffers from curse of dimensionality: points become increasingly isolated with more dimensions, for a fixed-size training dataset
- scikit-learn: sklearn.neighbors.KNeighborsClassifier

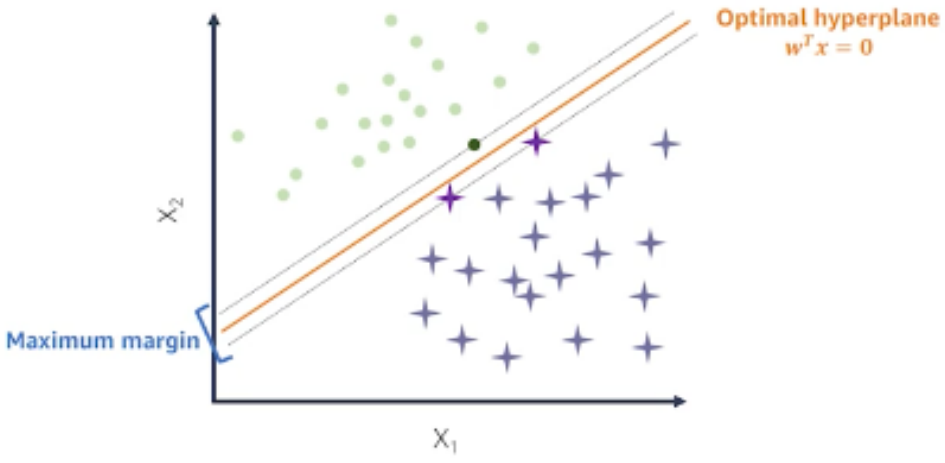
Supervised Learning: Linear and Non-Linear Support Vector Machines

In this video, we'll discuss linear and non-linear support vector machines and how they are commonly used in industry and in business.

2 classes: x1 and x2
both groups are lineray separated

Linear support vector machines

aws training and certification



The boundary data points are more important than other points
optimal hyper plane will maximize the margins to separate the 2 groups
The **support vectors are the data points at the boundary**

- Popular approach in research, but not in the industry
- Simplest case: Maximize the margin – the distance between the decision boundary (hyperplane) and the *support vectors* (training examples closest to the boundary)
- Max margin picture not applicable in non-separable case
- scikit-learn: sklearn.svm.SVC

- Also popular approach in research, but not in industry
- "Kernelize" for nonlinear problems:
  - Choose a distance function named a "kernel"
  - Map the learning task to a higher dimension space
  - Apply a linear SVM classifier in the new space
- Not memory-efficient, because it stores the support vectors, which grow with the size of the training data
- Computation is expensive
- scikit-learn: sklearn.svm.SVC

Supervised Learning: Decision Trees and Random Forests

This video covers decision trees and random forests. Deepening your understanding of decision trees and random forests can help you describe entropy and understand the concept of ensembles.

Example

aws training and certification

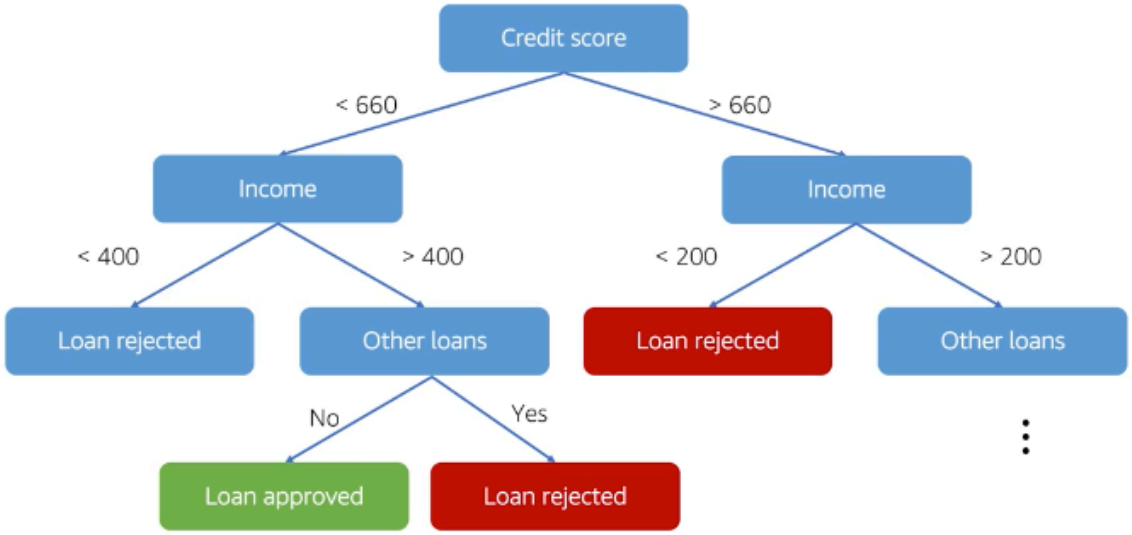
Will the loan of $500,000 be approved for the customer?

Sample	Age	Household Income (k)	Other Loans	Marital Status	Credit Score	Loan Approved
1	60	257	Yes	Married	650	No
2	36	352	Yes	Married	800	Yes
3	43	957	No	Single	720	Yes
4	38	492	No	Single	600	Yes
5	29	181	Yes	Single	585	No
6	49	324	No	Single	690	Yes
7	40	128	No	Married	750	No
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Decision tree is like a decision making process that we go through as human beings
We can make a mental model of the procedure to make the decision

Decision trees

aws training and certification

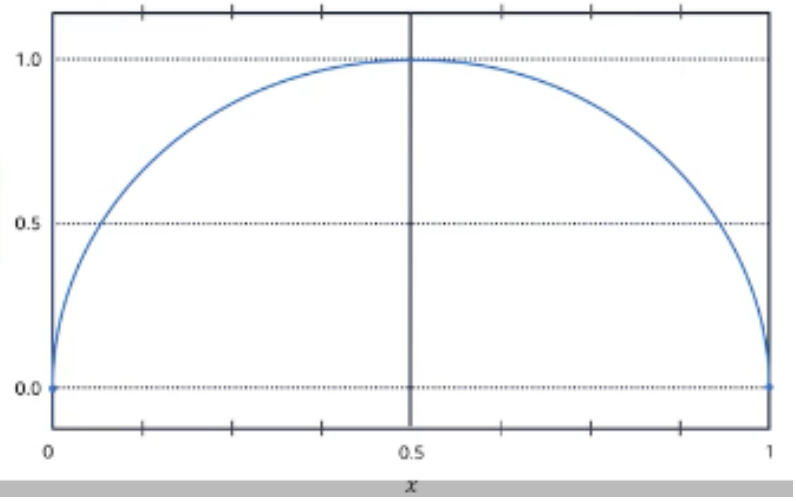


Entropy

aws training and certification

- Relative measure of disorder in the data source

$$H(X) = -\sum_{i=1}^{N} P(x_i)\log(P(x_i))$$



Disorder happens when there is no clear way to decide

Decision trees

aws training and certification

- **Nodes** are **split based** on the feature that has the **largest information gain** (IG) between parent node and its split nodes
- One metric to **quantify IG** is to **compare entropy before and after splitting**
- In a binary case:
  - **Entropy is 0** if all **samples belong to the same class** for a node (i.e., pure)
  - **Entropy is 1** if **samples contain both classes** with equal proportion (i.e., 50% for each class, chaos)
- The **splitting procedure** can go **iteratively** at each child node **until the end-nodes (or leaves) are pure** (i.e., there is only one class in in each node)
  - But the splitting procedure usually stops at certain criteria to prevent overfitting

Decision trees

aws training and certification

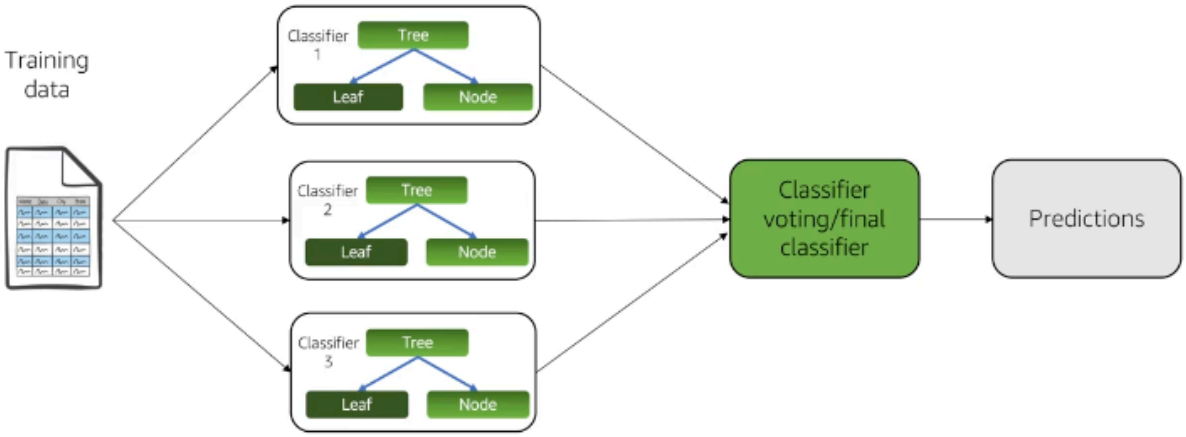
- Train (build the tree) by maximizing IG to choose splits (i.e., the impurity of split sets are lower)
- Easy to interpret (superficially)
- Expressive = flexible
- Less need for feature transformations
- Susceptible to overfitting
- Must “prune” the tree to reduce potential overfitting
- scikit-learn: sklearn.tree.DecisionTreeClassifier

Ensemble method - using many decision trees => **Random Forest**

Ensemble methods

aws training and certification

Learn multiple models and combine their results, usually via majority vote or averaging



Random forest algorithm

aws training and certification

- Set of decision trees, each learned from a different randomly sampled subset with replacement
- Features to split on for each tree, randomly selected subset from original features
- Prediction: Average output probabilities
- Increases diversity through random selection of training dataset and subset of features for each tree
- Reduces variance through averaging
- Each tree typically does not need to be pruned
- More expensive to train and run
- scikit-learn: sklearn.ensemble.RandomForestClassifier

Model Training: Validation Set

The training and tuning processes are *iterative*, and they are not separate from one another. Here, we investigate the training process and see how validation sets help tune models during this phase.

Model Training and Tuning are iterative steps
- Tuning -> tweak hyper parameters, add/remove features,...

Model Training and Tuning Motivation

aws training and certification

It's rare that the first model you train will meet your business requirements.

Model Training

Improve the model by optimizing parameters or data.

Model Tuning

Analyze the model for generalization quality and sources of underperformance such as overfitting.

Problems With Using Test Set

aws training and certification

Motivation

Model training and tuning involve comparing performance for different model or data settings.

Problem

When you use the test set for these comparisons, that effectively makes it part of a training set.

- The model may learn the patterns from the test set during tuning.

Validation Set

aws training and certification

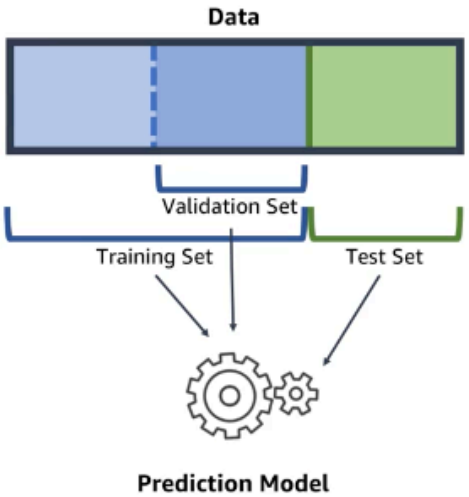
Solution

Split training data in two parts: a training set and a validation set.

- Use the training set to train candidate models, etc.
- The validation set plays the role of the test set during debugging and tuning.
- Save the test set for measuring the generalization of your final model.

Validation Set

aws training and certification



- Issue: Splitting the training data into training and validation sets may make it too small or unrepresentative.
- Solution: Use the holdout method to get the test set, then use k-fold cross-validation on the training set for debugging and tuning.

Model Training: Bias Variance Tradeoff

Machine learning models depend on input data, output data, and understanding the relationship between the two. Despite training and tuning, it's still difficult to find this mechanism.

Take a moment to explore how bias and variance affect the relationship between input and output data, before diving into this video on the bias variance tradeoff.

- Bias: an error from flawed assumptions in the algorithm. High bias can cause an algorithm to miss important relationships between features and target outputs resulting in underfitting.

- Variance: an error from sensitivity to small variations in the training data. High variance can cause an algorithm to model random noise in the training set, resulting in overfitting.

F = true model
F “hat” = estimated model

$$Total\ Error(x) = Bias^2 + Variance + Irreducible\ Error$$

Bias

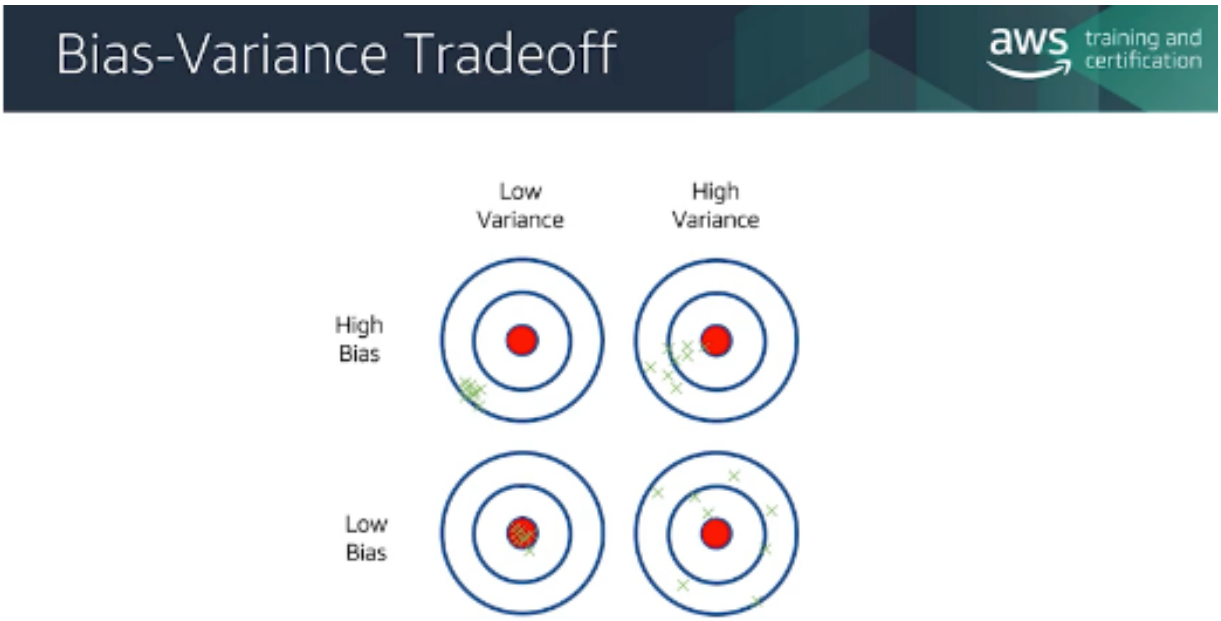
- $Bias = E[\hat{f}(x)] - f(x)$

Variance

- $Var = E[(\hat{f}(x) - E[\hat{f}(x)])^2]$

When model is highly biased - there is a systematic difference between true model and
Variance = variability

If a new input data come, with high variance, we may get a significantly different response

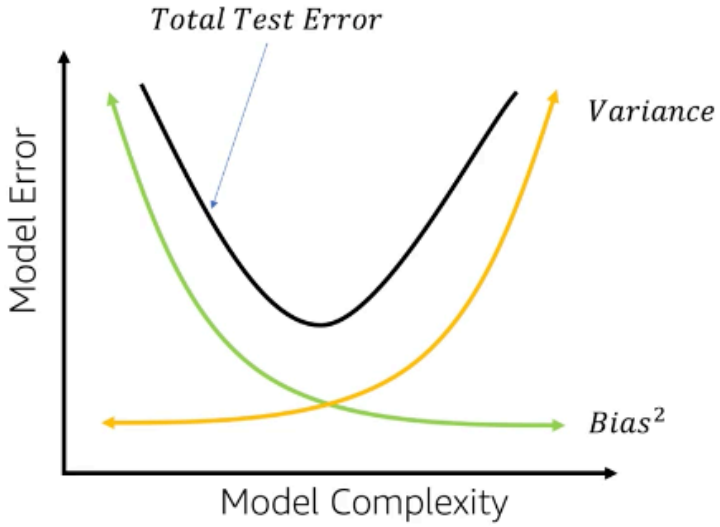


Good model = low bias and low variance
but most of the time, it's a trade-off between the 2

High bias model = indication of **underfitting** values => forecast values quite different from actual response
High variance model => indication of **overfitting** => small change in input can result in a large difference in output

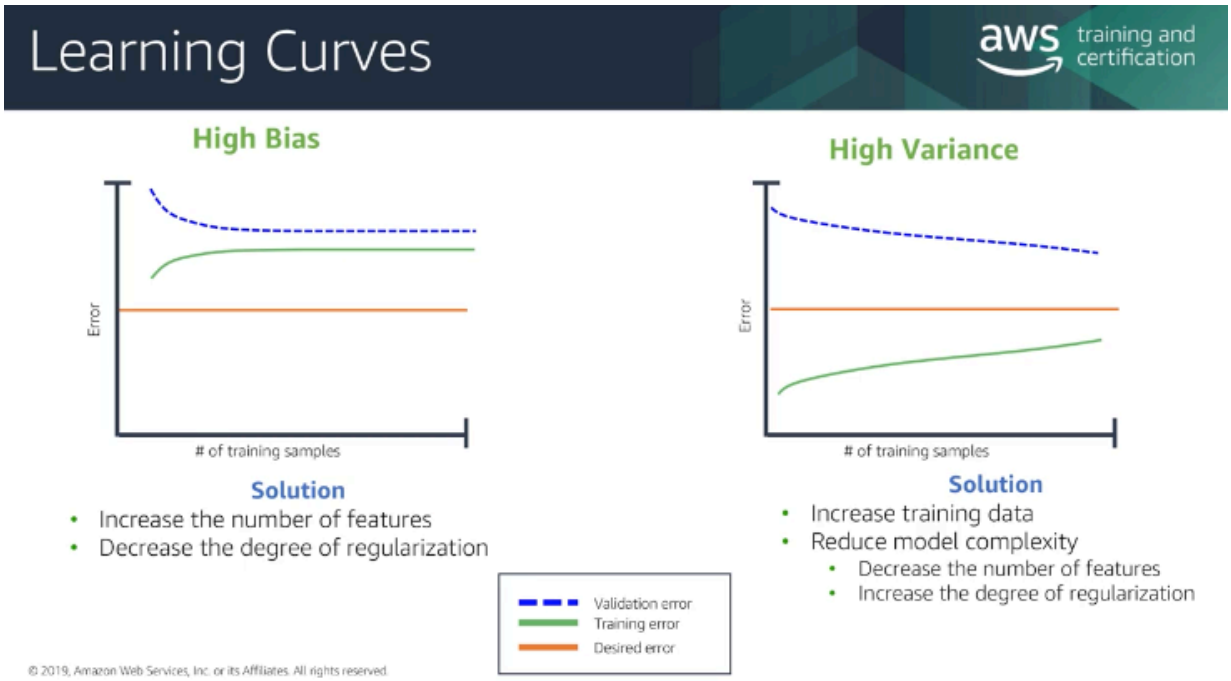
Middle point below is the "sweet spot" balancing variance and bias:

In General



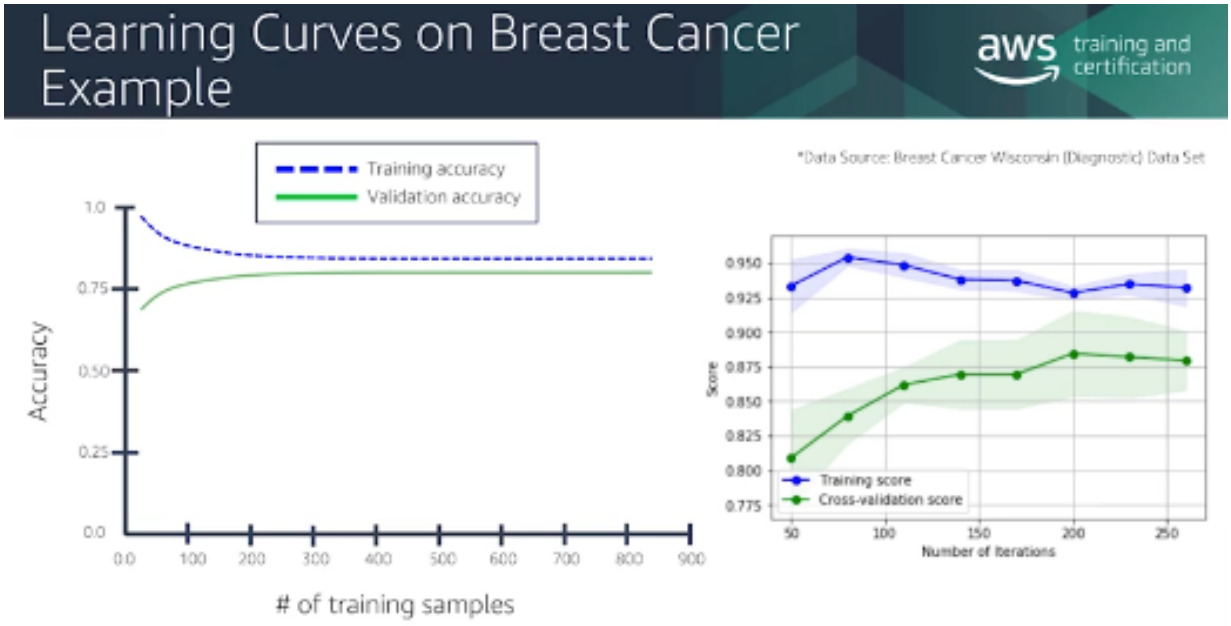
Using Learning Curves to Evaluate the Model

- Motivation: Detect if the model is underfitting or overfitting, and impact of **training data size** the error
- Learning curves: Plot training dataset and validation dataset error or accuracy against training set size
- scikit-learn: sklearn.learning_curve.learning_curve
  - Uses stratified k-fold cross-validation by default if output is binary or multiclass (preserves percentage of samples in each class)
  - Note: sklearn.*model_selection*.learning_curve in v.0.18



Example of SVM on cancer data

Training accuracy is better than validation accuracy.
=> The below model is under-fitting, with a high bias
=> we need a more complicated model with more data points



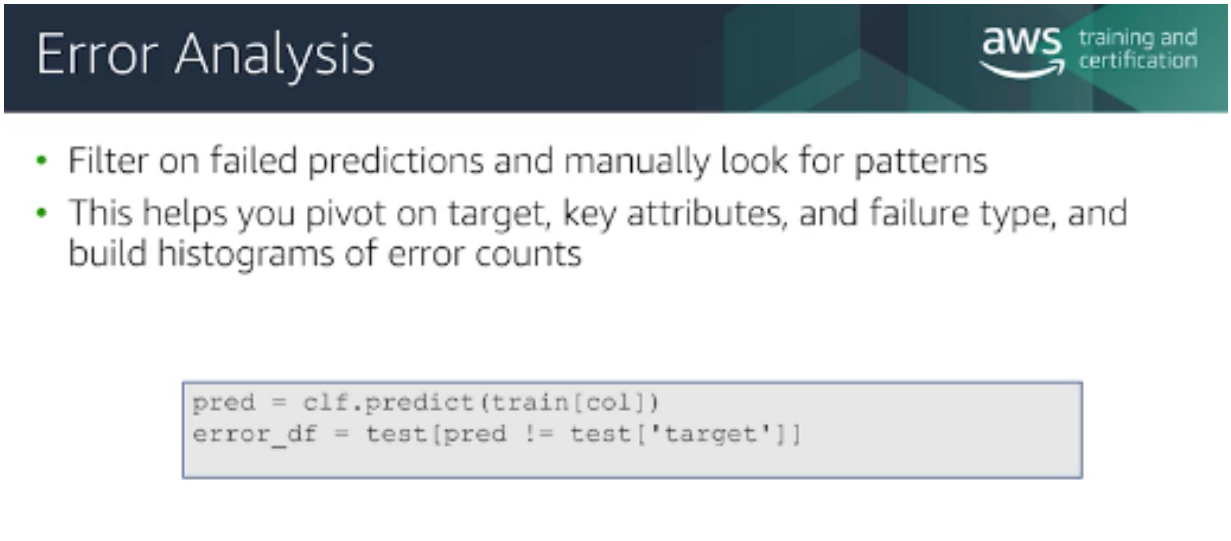
Overfitting if: training loss << validation loss
Underfitting if: training loss ?? validation loss

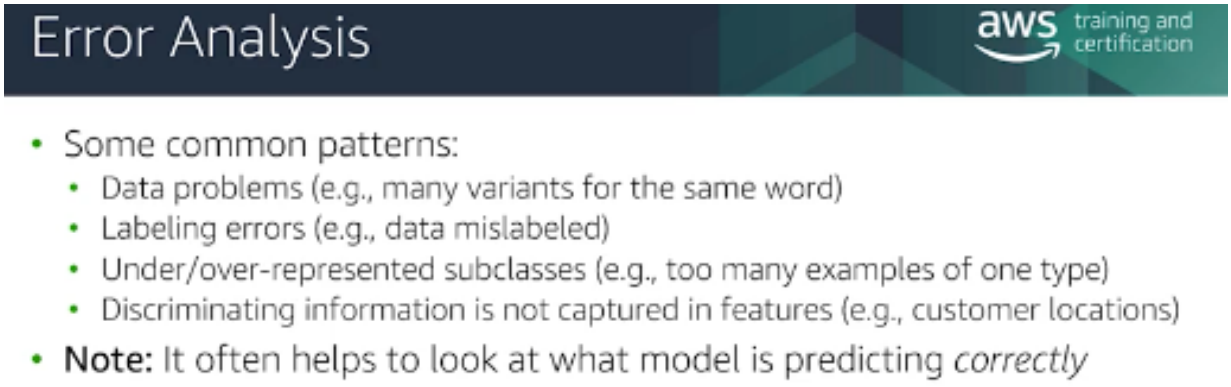
Just right is: training Loss ~ validation loss

Model Debugging: Error Analysis

We know that finding the relationship between input and output data is challenging, and models are prone to error. This section covers how to analyze those errors and use them to find potential problems with the data.

Residual = actual response vs forecast





Model Tuning: Regularization

Overfitting errors can be reduced using regularization - a technique that helps evenly distribute weights among features.

Regularization

aws training and certification

- Motivation: Overfitting often caused by overly-complex models capturing idiosyncrasies in training set
- Regularization: Adding penalty score for complexity to cost function

$$cost_{reg} = cost + \frac{\alpha}{2} penalty$$

Regularization in Linear Models

aws training and certification

- Idea: Large weights correspond to higher complexity
  ⟹Regularize by penalizing large weights

- Two standard types:
  - L1 regularization, Lasso: $penalty = \|\vec{w}\|_1 = \sum_{j=1}^{m} |w_j|$
  - L2 regularization, Ridge: $penalty = \|\vec{w}\|_2^2 = \sum_{j=1}^{m} w_j^2$

Model Tuning: Hyperparameter Tuning

Before we discuss how and when to tune the hyperparameters of a model, take a moment to distinguish between parameters and hyperparameters.

Model Tuning Choices

aws training and certification

Questions to address when choosing the optimal model

- What learning rate/nodes/layers should I use for the neural network model?
- What is the minimum number of samples I should use at the leaf node in a decision tree or random forest model?
- What is the optimum C parameter to use in a SVM model?

Hyperparameter Tuning

aws training and certification

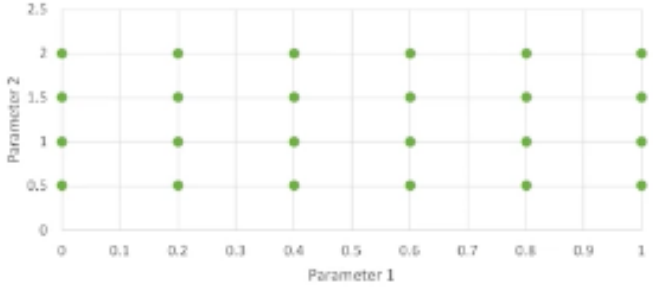
Hyperparameter: Estimator parameter that is **not** fitted to the data

- Hyperparameters must be optimized separately
- Techniques:
  - Grid search
  - Random search

Grid Search

aws training and certification

- Allows you to search for the best parameter combination over a set of parameters
- Compute intensive
- scikit-learn: sklearn.grid_search.GridSearchCV



Grid Search

aws training and certification

```
from sklearn.datasets import make_classification
from sklearn import svm
from sklearn.model_selection import GridSearchCV

plt.figure(4)
X2, Y2 = make_classification(n_features=5, n_redundant=0, n_informative=2)

parameters = {'kernel':('linear', 'rbf'), 'C':[1, 5, 10, 15], 'degree':[2,3,4,5]}
svc = svm.SVC()
clf = GridSearchCV(svc, parameters)
clf.fit(X2, Y2)

clf.best_params_
```

```
{'C': 5, 'degree': 2, 'kernel': 'rbf'}
```

Other Hyperparameter Tuning

aws training and certification

RandomizedSearchCV

- Each setting is sampled from a distribution over possible parameter values

Model Tuning

So far, you've split your data, run your model, addressed errors, and tuned some hyperparameters, but you'd still like to see an improved performance. It's time to tune your training data and your feature set.

Training Data Tuning

- Training set too small?
  - Sample and label more data if possible
- Training set biased against or missing some important scenarios?
  - Sample and label more data for those scenarios if possible
- Can't easily sample or label more?
  - Consider creating synthetic data (duplication or techniques like **SMOTE**)
- Important: Training data doesn't need to be exactly representative ("Whatever works"), but your test set does.

Feature Set Tuning

- Add features that help capture pattern for classes of errors.
- Try different transformations of the same feature.
  - Example: Square the values of a feature (as in earlier example)
- Apply dimensionality reduction to reduce impact of weak features.

Dimensionality Reduction

- Motivation: Common cause of overfitting: too many features for the amount of data
  - This is exacerbated if there are noisy/irrelevant features
  - Also a problem for curse of dimensionality
- Dimensionality reduction: Reduce the (effective) dimension of the data with minimal loss of information

Model Tuning: Feature Extraction

Not all features contribute to a model's forecasting power, and you can use feature extraction or feature selection to reduce these noisy or uninformative features. This video focuses on feature extraction.

Feature Extraction

aws training and certification

Maps data into smaller feature space that captures the bulk of the information in the data

- a.k.a. data compression

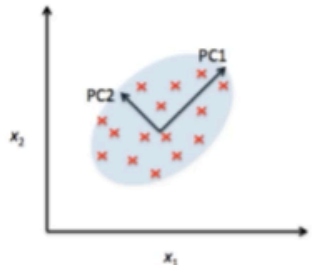
- Motivation
  - Improves computational efficiency
  - Reduces curse of dimensionality
- Techniques
  - Principal component analysis (PCA)
  - Linear discriminant analysis (LDA)
  - Kernel versions of these for fundamentally non-linear data

Principal Component Analysis (PCA)

aws training and certification

- Is an **unsupervised** linear approach to feature extraction
- Finds patterns based on correlations between features
- Constructs principal components: orthogonal axes in directions of maximum variance
- scikit-learn: sklearn.decomposition.PCA

```
>>> pca = PCA(n_components=2)
>>> X_train_pca = pca.fit_transform(X_train_std)
>>> lr = LogisticRegression()
>>> lr.fit(X_train_pca)
```

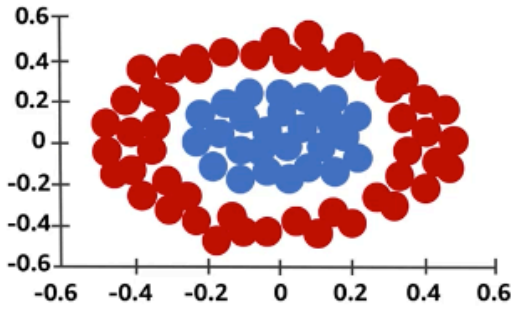


Example: linear relationship of x1 and x2 is not existent
But a relative non linear function will => distance from center => Z label variable.
We can see now how the 2 groups can be separated relatively easily => in that case we can just uze Z instead of X and Y (using just 1 feature instead of 2)

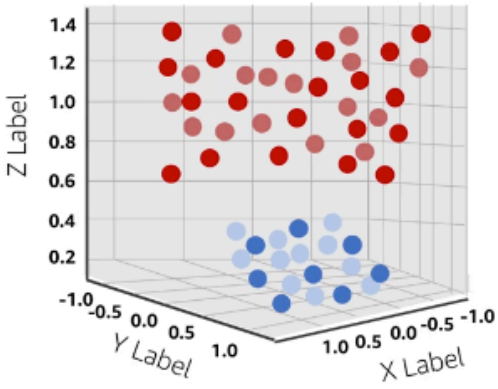
Kernel PCA

aws training and certification

Original



Kernel PCA



Linear Discriminant Analysis (LDA)

aws training and certification

- A **supervised** linear approach to feature extraction
- Transforms to subspace that maximizes class separability
- Assumes data is normally distributed
- Used for dimensionality reduction of features
- Can reduce to at most #classes-1 components
- scikit-learn: sklearn.discriminant_analysis.LinearDiscriminantAnalysis

Model Tuning: Bagging/Boosting

Feature extraction and selection are relatively manual processes. This video covers bagging and boosting, which are automated or semi-automated approaches to determining which features to include.

Bagging (Bootstrap Aggregating)

aws training and certification

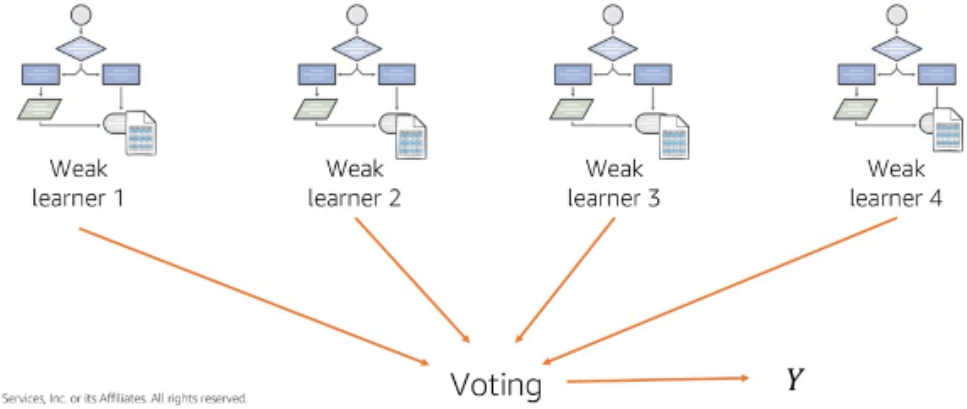
Motivation: generate a group of weak learners that when combined together generate higher accuracy

- Create a x datasets of size m by randomly sampling original dataset with replacement (duplicates allowed)

Bagging (Bootstrap Aggregating)

aws training and certification

- Train weak learners (decision stumps, logistic regression) on the new datasets to generate predictions
- Choose the output by combining the individual predictions or voting



Bagging

aws training and certification

High variance but **low** bias?

Use **bagging**:

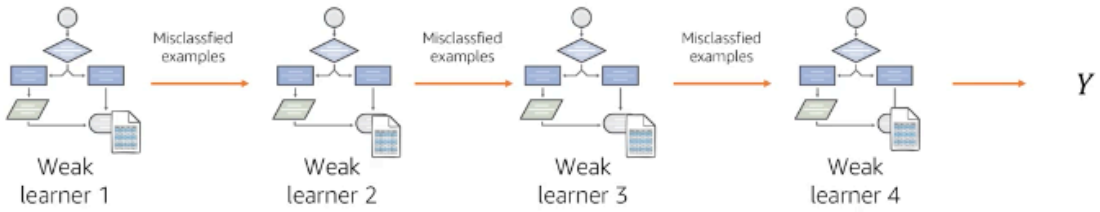
Training many models on random subsets of the data and average/vote on the output.

- Reduces variance
- Keeps bias the same
- sklearn:
  - sklearn.ensemble.BaggingClassifier
  - sklearn.ensemble.BaggingRegressor

Boosting

aws training and certification

- Assign strengths to each weak learner
- Iteratively train learners using misclassified examples by the previous weak learners



Boosting

aws training and certification

Model has a **high bias** and accepts weights on **individual samples**?

Use boosting:

Training a sequence of samples to get a strong model.

- Often times wins on datasets like most Kaggle competitions
- sklearn:
  - sklearn.ensemble.AdaBoostClassifier
  - sklearn.ensemble.AdaBoostRegressor
  - sklearn.ensemble.GradientBoostingClassifier
- XGBoost library

Lesson 9 of 12

Knowledge Check 4

Q1

What is bias in the context of bias/variance tradeoff?

bias is usually representative of overfitting. As bias grows, variance de

Here's what you said:

bias is usually representative of overfitting. As bias grows, variance decreases and vice-versa

Here's what we said:

Bias is the estimation of the difference between the fitted model and actual relationship of response and features, and a high bias model usually indicates lack of fit, i.e. the model is not flexible enough to capture the relationship between response variable (i.e. target or label variable) and explanatory variables (i.e. features). Solutions to reduce high bias include using more complicated models and adding more features (such as new features, transformation of current features, and interactions among current features).

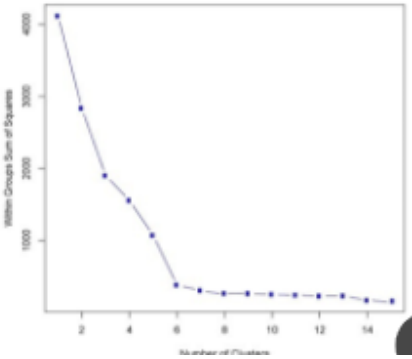
Here's what you said:

Here's what we said:

Both feature selection and feature extraction are used to reduce the dimensionality of the feature space. Feature selection use algorithms to remove some of the features from the model such that the selected features will enable the model to have better performance and there is no change (such as transform, linear combination or non-linear combination) in the selected features themselves. Feature extraction, on the other hand, is using algorithms to combine of the original features (such as linear or non-linear function of original features) to generate a set of new features and the number of new features to be used in the model is generally less than the original number of features.

Q3

I am running k-means clustering on a dataset and I am looking to decide the number of clusters and I get the following graph for the within group sum of squares as a function of cluster number. What is a good number of clusters to pick?



- A. 5
- B. 6
- C. 10
- D. 14

Submit

Q4

Imagine a scenario in the manufacturing industry. The manufacturer needs to know the demand (i.e number of future sales in a specific period) of each of the products to streamline the supply chain. In this scenario, you are asked to help with machine learning. **What will you do to start?**

- A. Do a classification analysis for products.
- B. Start a discussion with the business stakeholder to define the use case of this project and then formulate the business question into a machine learning question.
- C. Do a clustering analysis for the customers.
- D. Study correlations among features.