**Intro**: ML: an algorithm to automatically learn from data, or from experience, uncover patterns in data, building autonomous agents

Supervised learning: have labeled examples of the correct behavior

Semi-supervised learning: utilizes both labeled and unlabeled data

Reinforcement learning: learning system (agent) interacts with the world and learns to maximize a scalar reward signal

Unsupervised learning: no labeled examples – instead, looking for “interesting” patterns in the data

Artificial intelligence: A subfield of CS, AI refers to computer programs that can solve problems humans are good at (e.g. vision, natural language,)

Machine learning: A subfield of AI focused on learning (i.e., tuning parameters) from data

Neural networks: A parametric model used in ML; (very loosely) based on biological neurons

Deep Learning: Neural networks with multiple layers (i.e., processing steps)

Data science: An emerging eld which applies ML techniques to domain-specific problems

**Data**: Workflow:

1. Should I use ML on this problem?
2. Gather and organize data: (Preprocessing, cleaning, visualizing)
3. Establishing a baseline.
4. Choosing a model, loss, regularization, ...
5. Optimization
6. Hyperparameter search.
7. Analyze performance & mistakes, and iterate back to step 4 (or 2).

Input Vectors/ Training set: a collection of pairs of an input vector x and its corresponding target t.

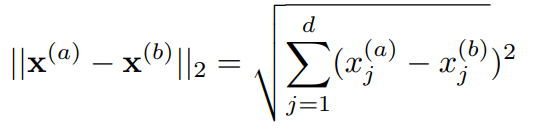
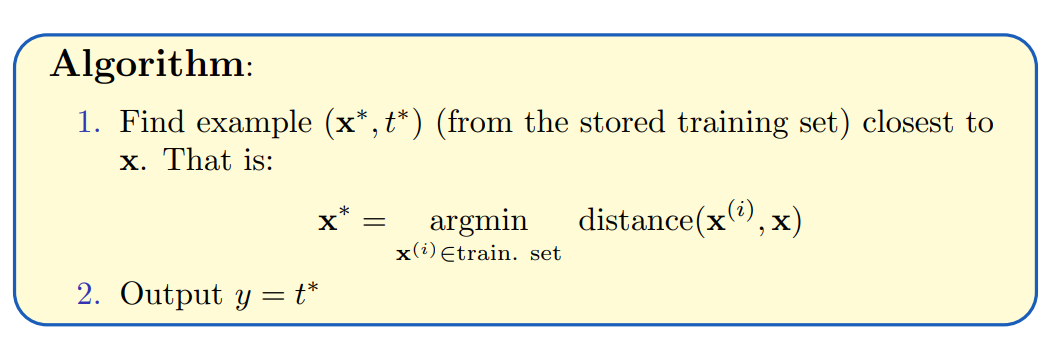
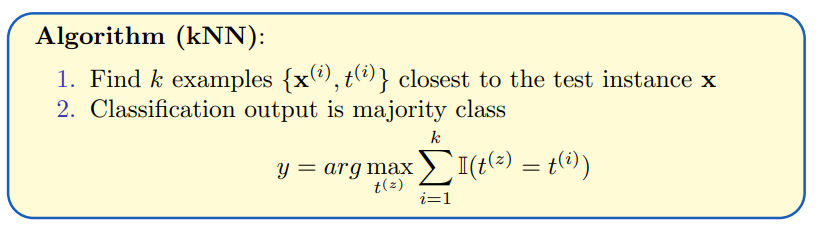
Feature types: categorical feature and numerical feature

Accuracy = correct prediction / test instances

√(X₂-X₁)²+(Y₂-Y₁)²

Error = 1- accuracy = incorrect prediction / test instances

**KNN:** a supervised classification algorithm that predicts the class of an output feature based on the class of other instances with the most similar, or "nearest," input features

Decision boundary: the boundary between regions of input space assigned to different categories.

1. Measure the distance: Calculate the distance between the new data point and all the data points in the dataset.

use the Euclidean distance, which is like measuring the straight-line distance between two points.

1. Find the K nearest neighbors: Identify the K points with the shortest distances to the new point. These are the K nearest neighbors.

For instance, if K = 3, we select the three points that are closest to our new point.

1. Majority voting: Among the K nearest neighbors, count how many points are there for each class type.

Whichever type has the majority becomes our prediction for the new point.

Small k:Good at capturing fine-grained patterns

May overfit, i.e. be sensitive to random idiosyncrasies in the training data

Overfitting happens when a model becomes too complex, capturing noise or irrelevant patterns from the training data.

 model become overly sensitive to local variations in the training data, leading to poor generalization to unseen data.

Large k:Makes stable predictions by averaging over lots of examples

May underfit, i.e. fail to capture important regularities

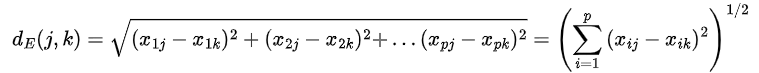
Underfitting occurs when a model is too simple to capture the underlying patterns in the data

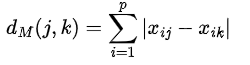
model is too generalized and oversimplifies the underlying patterns in the data.

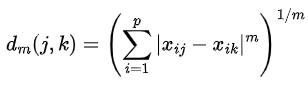
Balancing k:Optimal choice of k depends on number of data points n.

Nice theoretical properties if

Rule of thumb: K=3/ K<sqrt(n)/ using validation set

Euclidean distance: 

Manhattan distance: 

Minkowski distance: 

Simple, No training, Easy to justify classification to customer, Can easily do multi-class.

Limitations: Large dataset: lazy learning technique: in training phase KNN doing nothing, so training is fast but on time of prediction it becomes slow as large dataset come since model has to calculate euclidean distance from given point to all points in the dataset

Curse of Dimensionality:feature space becomes increasingly sparse as the number of dimensions (features) grows. In high-dimensional spaces, the notion of proximity or similarity becomes less meaningful. “most” points are approximately the same distance

Imbalanced dataset: the majority class typically has significantly more samples than the minority class.

large number of neighbors from the majority class can overpower the neighbors from the minority class, dominate the decision-making process, leading to a bias towards the majority class in the predictions.

**Decision Tree:** Make predictions by splitting on features according to a tree structure.

Split continuous features by checking whether that feature is greater than or less than some threshold. Internal nodes test a feature/Branching is determined by the feature value /Leaf nodes are outputs (predictions)

Classification tree: discrete output, leaf value typically set to the most common value

How to automatically find a good hypothesis for training data?

This is an algorithmic question, the main topic of computer science

-For any training set we can construct a decision tree that has exactly the one leaf for every training point, but it probably won’t generalize.

-Decision trees are universal function approximators.

When do we generalize and do well on unseen data?

-Learning theory quantifies ability to generalize as a function of the amount of training data and the hypothesis space

-Occam’s razor: use the simplest hypothesis consistent with data!

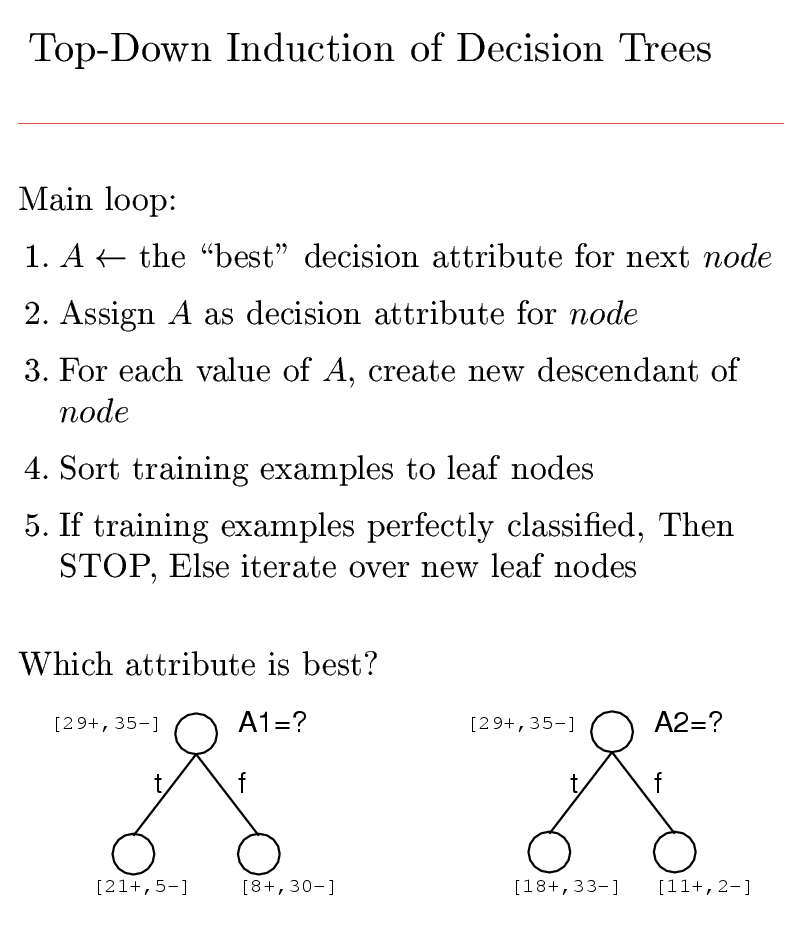
Decision trees: if we were able to find a small decision tree that explains data well, then good generalization guarantees.

Top-Down Induction of Decision Trees: Resort to a greedy heuristic:

Start with the whole training set and an empty decision tree.

Pick a feature and candidate split that would most reduce the loss.

Split on that feature and recurse on subpartitions.



Use counts at leaves to define probability distributions; use a probabilistic notion of uncertainty to decide splits.

information gain: how well a given attribute separates the training examples according to the target classification

Information Gain of A is the expected reduction in entropy of target variable Y for data sample S, due to sorting on variable A 

Entropy information theoretic measure that characterizes the impurity of a labeled set 𝑆.

𝑆 is a sample of training examples

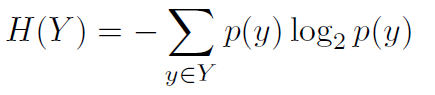
𝑝⊕ is the proportion of positive examples in 𝑆.

𝑝⊖ is the proportion of negative examples in 𝑆.

Entropy measures the impurity of 𝑆 with 2 class.

𝐻 (𝑆) ≡ −𝑝⊕ log2 𝑝⊕ − 𝑝⊖ log2 𝑝⊖

the entropy of a discrete random variable Y



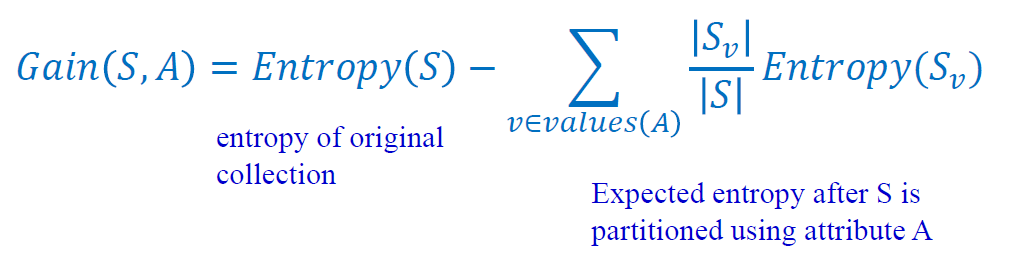
“High Entropy”:

* Variable has a uniform like distribution over many outcomes
* Flat histogram
* Values sampled from it are less predictable

“Low Entropy”

* Distribution is concentrated on only a few outcomes
* Histogram is concentrated in a few areas
* Values sampled from it are more predictable

Gain(S,A) information provided about the target function, given the value of some other attribute A.

Conditional Entropy:

* H is always non-negative
* Chain rule: H(X, Y ) = H(X|Y ) + H(Y ) = H(Y |X) + H(X)
* If X and Y independent, then X does not affect our uncertainty about Y : H(Y |X) = H(Y )
* But knowing Y makes our knowledge of Y certain: H(Y |Y ) = 0
* By knowing X, we can only decrease uncertainty about Y : H(Y |X) ≤ H(Y )

information gain:

IG(Y |X) in Y due to X, or the mutual information of Y and X: IG(Y |X) = H(Y ) − H(Y |X)

IG(split) = H(Y)-P(L)H(Y|L)-P(R)H(Y|R)

measures the informativeness of a variable, which is exactly what we desire in a decision tree split!

The information gain of a split: how much information (over the training set) about the class label Y is gained by knowing which side of a split you’re on.

Good Tree: Not too small or too big (Occam’s Razor: find the simplest hypothesis that fits the observations)

DT vs KNN:

-Simple to deal with discrete features, missing values, and poorly scaled data

-Fast at test time

-More interpretable

-Few hyperparameters

-Can incorporate interesting distance measures Advantages of neural nets over decision trees

-Able to handle attributes/features that interact in very complex ways

Amount of Overfitting: 𝑒𝑟𝑟𝑜𝑟𝑡𝑟𝑢𝑒(ℎ) − 𝑒𝑟𝑟𝑜𝑟𝑡𝑟𝑎𝑖𝑛(ℎ)

Stop growing when data split not statistically significant

Grow full tree, then post-prune

**Ensemble Learning:**

bias: how wrong the expected prediction is (corresponds to underfitting)

variance: the amount of variability in the predictions (corresponds to overfitting)

Bayes error: the inherent unpredictability of the targets

simple model: high bias (because it cannot capture the structure in the data), low variance (because there's enough data to get stable estimates)

complex model: low bias (since it learns all the relevant structure),high variance (it fits the quirks of the data you happened to sample)

Ensembles combine classifiers to improve performance

Boosting

* Reduces bias
* Increases variance (large ensemble can cause overfitting)
* Sequential
* High dependency between ensemble elements

Bagging

* Reduces variance (large ensemble can’t cause overfitting)
* Bias is not changed (much)
* Parallel
* Want to minimize correlation between ensemble elements.

Bagging involves creating different classifiers on different random samples of training data

Random Forests: bagged decision trees

Advantages: It can be used in classification and regression problems.

It solves the problem of overfitting as output is based on majority voting or averaging.

It performs well even if the data contains null/missing values.

Each decision tree created is independent of the other; thus, it shows the property of parallelization.

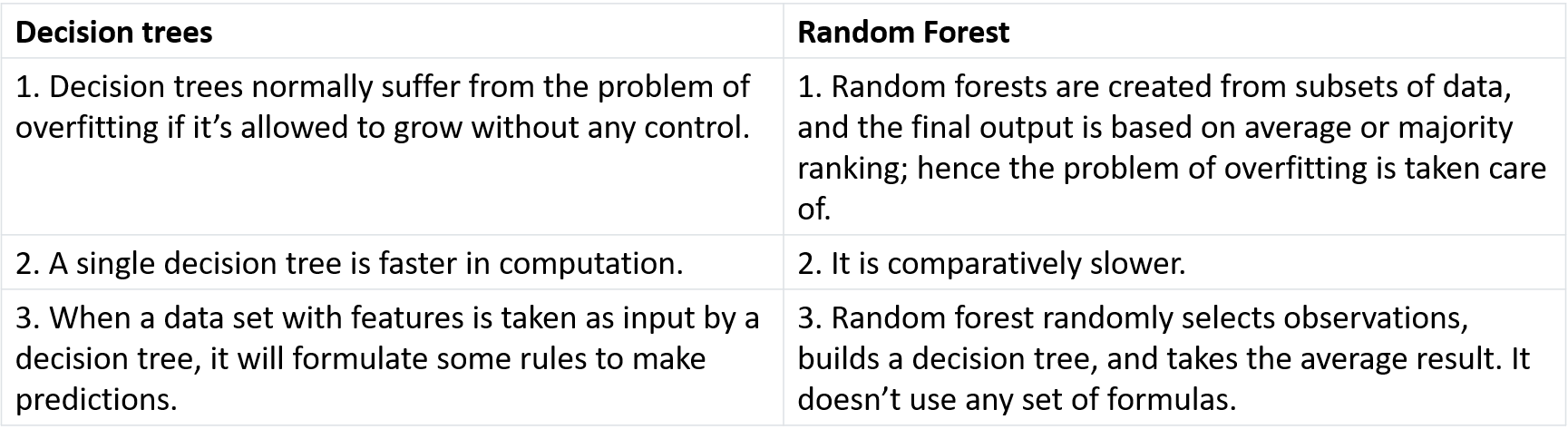
It is highly stable as the average answers given by a large number of trees are taken.

It maintains diversity as all the attributes are not considered while making each decision tree though it is not true in all cases.

It is immune to the curse of dimensionality. Since each tree does not consider all the attributes, feature space is reduced.

Disadvantages: Random forest is highly complex compared to decision trees, where decisions can be made by following the path of the tree.

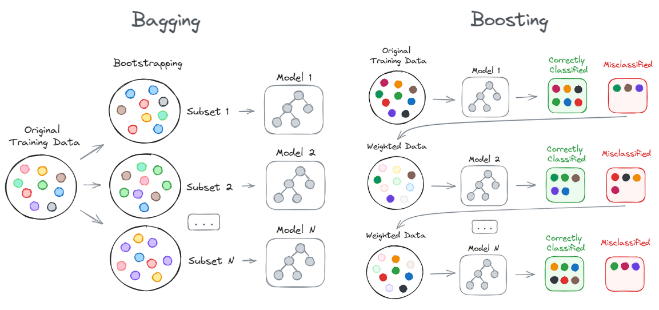
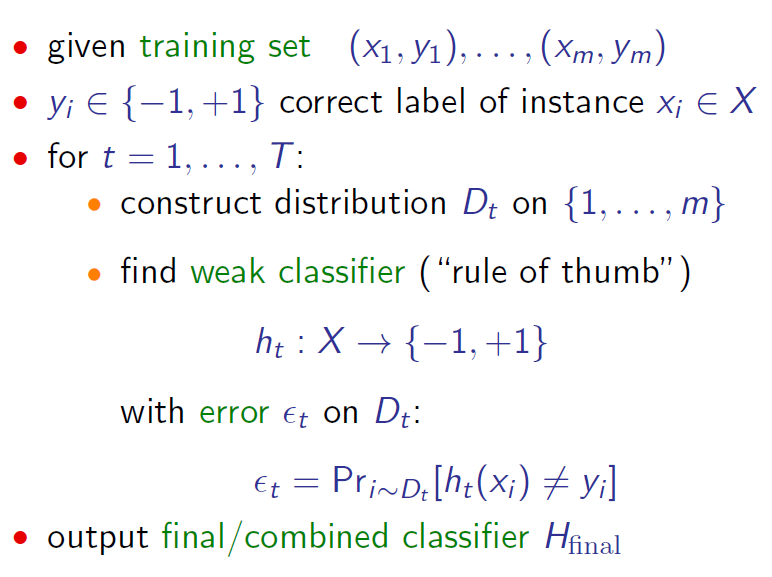
Training time is more than other models due to its complexity. Whenever it has to make a prediction, each decision tree has to generate output for the given input data.

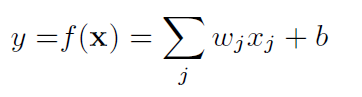


Boosting: Train classifiers sequentially, each time focusing on training examples that the previous ones got wrong.

AdaBoost: Uses all training data in every iteration, But weighs the training instances differently in each iteration, so the classifier is encouraged to get certain instances correct over others

1. At each iteration, re-weight the training samples by assigning larger weights to samples (i.e., data points) that were classified incorrectly.
2. Train a new base classier based on the re-weighted samples.
3. Add it to the ensemble of classifiers with an appropriate weight.
4. Repeat the process many times.

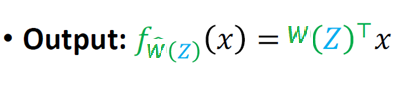
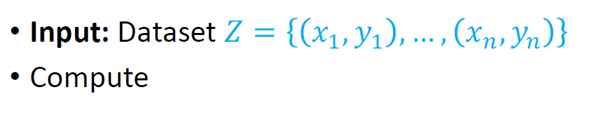


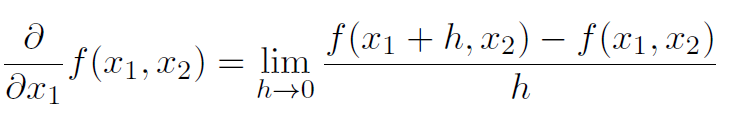
**Linear Regression**

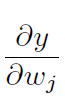
A loss function defines how bad it is.

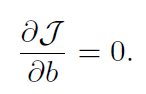
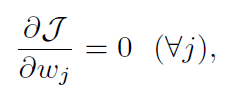
Squared error loss function:

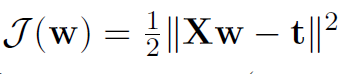
Mean Squared error = (1/n) \* Σ(yᵢ - ȳ)² loss function averaged over all training examples

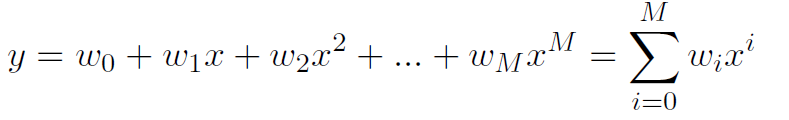


derivative of a multivariate function with respect to one of its arguments. 

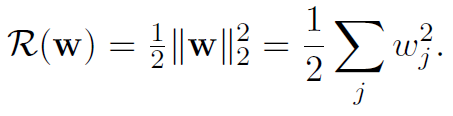


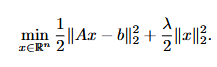
Partial derivative for 1 dimension

Partial derivative for d dimension  w=(XTX)−1 XTy

Polynomial Feature Mapping 

M=0, y=w0​ M = 1, y=w0​+w1​x M=3, y=w0​+w1​x+w2​x2+w3​x3

Regularizer: a function that quantifies how much we prefer one hypothesis vs. another 

Regularized Least Squares 

Linear regression exemplifies recurring themes of this course:

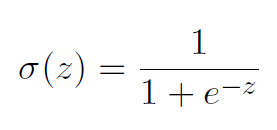
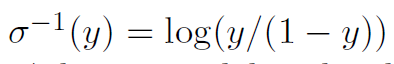
* choose a model and a loss function
* formulate an optimization problem
* solve the minimization problem using one of two strategies
* direct solution (set derivatives to zero)

I gradient descent

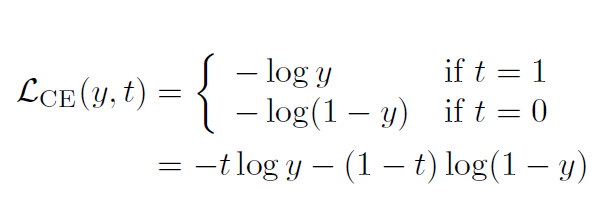
vectorize the algorithm, i.e. represent in terms of linear algebra

make a linear model more powerful using features

improve the generalization by adding a regularizer

Logistic Activation Function  

A linear model with a logistic nonlinearity is known as log-linear:

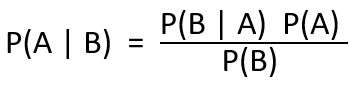
Cross-entropy loss 

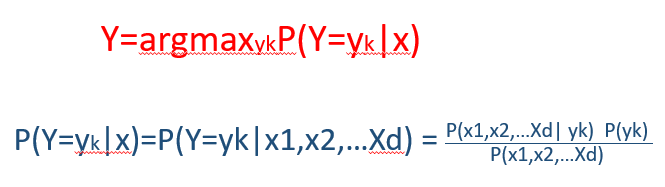
gradient descent: initialize the weights to something reasonable and repeatedly adjust them in the direction of steepest descent.

Gradient of Logistic Loss:

**Naïve Bayes**

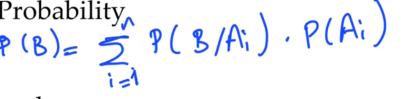
Bayesian inference = Guessing in the style of Bayes, P(Y|X).

Conditional probability: 

Naive Bayes classifier: 

Product Rule: P(A,B) = P(A|B)\*P(B) = P(B|A)\*P(A), P(A|B) = P(A,B)/P(B)

Sum Rule: P(A OR B) = P(A) + P(B)- P(A AND B)

Total Probability: 

Independent: P(A,B) = P(A)\*P(B)

conditional independent: P(A,B|C) = P(A|B,C)\*P(B|C) = P(A|C)\*P(B|C) = P(A|B,C)\*P(A|C)

**Evaluation Validation**

Ensures model generalization to unseen data

Helps detect and prevent overfitting

Provides confidence in model performance

Guides model selection and improvement

Critical for real-world applications and decision-making

Model evaluation is the process of using metrics to assess how well a supervised machine learning model's predictions match observed values.

Train-Test Split: Dividing the dataset into two subsets - one for training and one for testing

Confusion Matrix: a table that summarizes the combinations of predicted and actual values.

* A ***true positive (TP)*** is an outcome that is correctly predicted as positive.
* A ***true negative (TN)*** is an outcome that is correctly predicted as negative.
* A ***false positive (FP)*** is an outcome that is predicted as positive but is actually negative.
* A ***false negative (FN)*** is an outcome that is predicted as negative but is actually positive.

Accuracy: proportion of correct predictions

Acc= (TP + TN) / (TP + TN + FP + FN)

Precision :  the proportion of correct positive predictions

Precision = TP / (TP + FP)

Recall : the proportion of correctly predicted positive instances an

Recall = TP / (TP + FN)

F1-score: combine the precision and the recall into one measure

F1= 2 \* (Precision \* Recall) / (Precision + Recall)

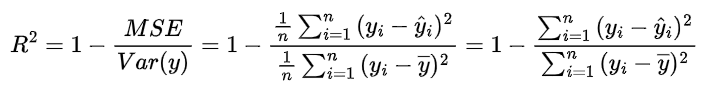
Regression Metrics: how close a model's predictions are to the actual values

Mean Absolute Error (MAE) = Σ|y\_true - y\_pred| / n

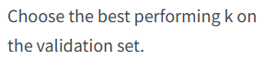
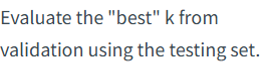
Mean Squared Error (MSE) = Σ(y\_true - y\_pred)² / n

Root Mean Squared Error (RMSE) = √MSE

R-squared (R²) = 1 - (SSres / SStot)



Model validation: the process of evaluating a model's initial performance and adjusting parameter estimates or hyperparameter settings if needed.

Validation on knn   

K-Fold Cross-Validation: Split the data into K folds/For each fold i (i = 1 to K):/Average the performance across all K iterations

Model tuning:  is the process of selecting the best hyperparameter for a model using cross-validation. hyperparameter is a user-defined setting in a machine learning model that is not estimated during model fitting.

Leave-One-Out Cross-Validation (LOOCV): extreme case of k-fold cross-validation, where k equals the number of samples in the dataset

* For a dataset with n samples, LOOCV uses n-1 samples for training and 1 sample for testing1.
* This process is repeated n times, each time leaving out a different sample for testing1.
* The model's performance is evaluated based on how well it predicts the left-out sample in each iteration.
* The final performance estimate is the average of all n iterations

Stratified splits: are evenly split, or balanced, for all levels of the output feature. stratification helps avoid underfitting.

Preventing Overfitting

* Increase training data
* Feature selection/reduction
* Regularization techniques (L1, L2)
* Early stopping in iterative algorithms
* Ensemble methods

Addressing Underfitting

* Increase model complexity
* Add more relevant features
* Reduce regularization
* Train for more epochs (for neural networks)

**Bayesian Networks:** conditional independence between subsets of variables

Encodes the conditional independence relationships between the variables in the graph structure

Is a compact representation of the joint probability distribution over the variables

****

Non Linearity. Bayesian network structure implies conditional independencies!

d-separation reduces statistical independencies (hard) to connectivity in graphs (easy)( If no such path, then X and Z are d-separated with respect to Y)

Using a Bayesian network to compute probabilities is called inference

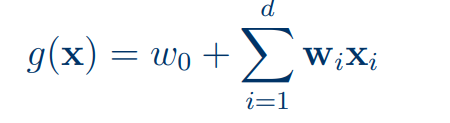
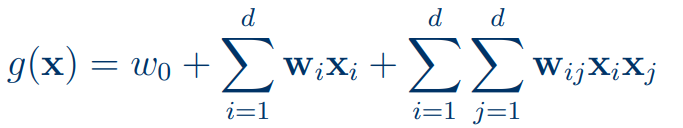
Learning conditional independence Constraints. Search strategy: top-down.

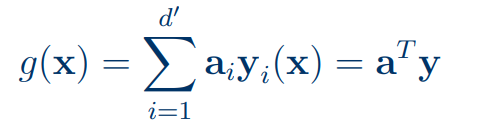
* Start with the saturated (undirected) graph.
* Go link by link and test the independence.
* If independence holds, remove the arc.
* Swing the variables to assess the link direction

**SVM**

discriminant function :used to set a decision boundary between classes

* , if g(x)>0 ‘+’, else ‘-‘

linear discriminant function quadratic discriminant function 

generalized linear discriminant function 

Perceptron Algorithm

* Set t=1, start with the all zero vector 𝑤1.
* Given example 𝑥, predict positive if 𝑤𝑡⋅𝑥≥0
* On a mistake, update as follows:
  + Mistake on positive, then update 𝑤𝑡+1←𝑤𝑡+𝑥
  + Mistake on negative, then update 𝑤𝑡+1←𝑤𝑡−𝑥
* Natural greedy procedure:
  + If true label of x is +1 and 𝑤𝑡 incorrect on x
    - we have 𝑤𝑡⋅𝑥<0, 𝑤𝑡+1⋅𝑥←𝑤𝑡⋅𝑥+𝑥⋅𝑥=𝑤𝑡⋅𝑥+𝑥2, so more chance 𝑤𝑡+1classifies x correctly.
  + Similarly for mistakes on negative examples.

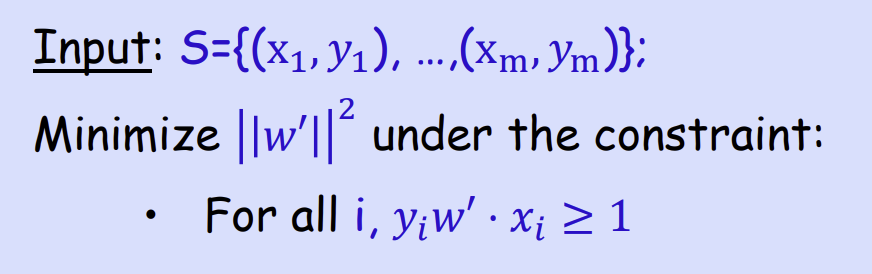
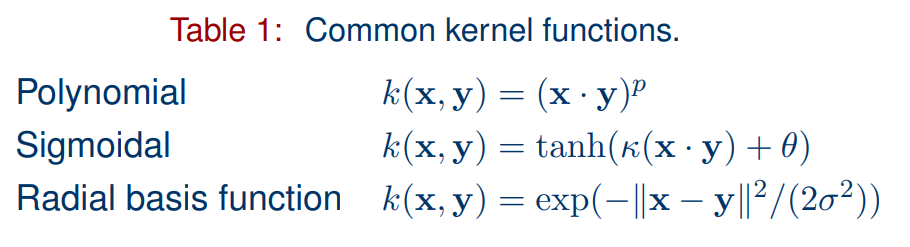
Geometric Margin: Large margin can help prevent overfitting.

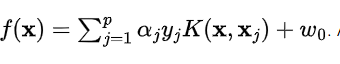
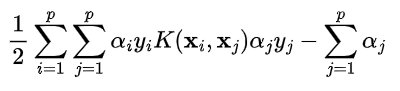
A dataset is completely separated if all instances of each class fall on the same side of a hyperplane with no overlap.

A soft margin is a margin boundary that allows a few instances to be misclassified but with an associated penalty.

A support vector classifier categorizes instances using a hyperplane with a soft margin.

Support Vector Machines (SVMs): Directly optimize for the maximum margin separator

decision function for a new instance u is minimize 

SVMs are quite popular because of their intuitive formulation using computational learning theory and their high performances in practical applications.