**Overfitting:**

1. fitting to noisy point

2. clean training dataset, but because the training set is a limited sample, there might be (combinations of) features that are correlated with the target concept by chance

3. larger the hypothesis class, easier to find a hypothesis that fits the difference between the training data and the true distribution

**Prevent overfitting:**

1. cleaner training data help!

2. more training data help!

3. throwing away unnecessary hypotheses helps!

**Decision Tree:**

Can we find and return the smallest possible decision tree that accurately classifies the training set? -- **NO!** This is an NP-hard problem

**Entropy**

A picture containing object

Description automatically generatedA picture containing object

Description automatically generatedA close up of a logo

Description automatically generated

A picture containing furniture

Description automatically generatedA screenshot of a cell phone

Description automatically generated

**Stopping Criteria:**

- Info gain for all features are zero

- all of the given subset of instances are of the same class

- we’ve exhausted all of the candidate splits

- Use Validation set to check if validation score improves, otherwise, just stop

(Pruning) Build a tree, then use validation set to check if pruning a node will reduce validation error. Stop when further pruning will increase validation error

**Regression Trees:** Use MSE to split: choose the split that greatest reduces the MSE.

A picture containing object

Description automatically generatedA close up of a clock

Description automatically generated

**Supervised Learning:**

**Non-parametric:** model’s complexity changes with learning decision tree, nearest neighbors

**Parametric:** model’s complexity doesn’t change with learning

**Generative classifiers (**Naïve Bayes, Bayesian networks**):**

**1.**Assume some functional form for **P(Y), P(X|Y); 2.**Estimate parameters of **P(X|Y), P(Y)** directly from training data; **3**.Use Bayes rule to calculate **P(Y |X)**

**Discriminative Classifiers (**linear/logistic regression, SVM, NN**):**

**1.**Assume some functional form for **P(Y|X);**

**2.** Estimate parameters of **P(Y|X)** directly from training data

**Evaluation:**

**Learning curve**: x – training set sample size, y – error on *test set*

Reason for developing **Cross-Validation** method:

Not enough data for robust training and test datasets ?

**Confusion Matrix:** X – predicted**;** Y – actual

**Confusion Matrix in 2-class situation:**

TP rate (Recall) = TP/actual positive

FP rate = FP/actual negative

Precision = TP/predicted positive = TP/(TP+FP)

**ROC Curve:** X – FP rate, Y – TP rate

**Precision/Recall curve:** X – Recall, Y – precision

**Get ROC and precision/recall curve from cross validation***:*

1. get predictions for all test sets and plot the curves

2. (For ROC) plots separate plots and plot the average plot for these plots (take average)

Both

• allow predictive performance to be assessed at various levels of confidence

• assume binary classification tasks

• sometimes summarized by calculating area under the curve

• show the fraction of predictions that are false positives

**ROC curves**

• insensitive to changes in class distribution (ROC curve does not change if the proportion of positive and negative instances in the test set are varied)

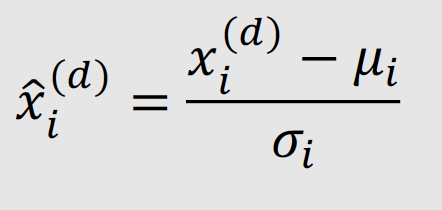
• can identify optimal classification thresholds for tasks with differential misclassification costs

**precision/recall curves**

• well suited for tasks with lots of negative instances

**Standardizing numeric features**

**A close up of a clock

Description automatically generated**

**Strength of KNN:**

• simple to implement

• “training” is very efficient

• adapts well to on-line learning

• robust to noisy training data (when k > 1)

• often works well in practice

**Weakness:**

• sensitive to range of feature values

• sensitive to irrelevant and correlated features, although …

• there are variants (such as locally weighted regression) that learn weights for different features

• later we’ll talk about feature selection methods

• classification/prediction can be inefficient, although edited methods and k-d trees can help alleviate this weakness

• doesn’t provide much insight into problem domain because there is no explicit model

**Linear Regression:**

Close-form solution: w = (X^TX)^(-1)X^Ty

Lasso regularization:A close up of a logo

Description automatically generated lasso penalty: 𝑙1 norm of the parameter, encourages sparsity

A close up of a clock

Description automatically generated, where H(x) is the prediction

A picture containing device

Description automatically generated

Bounded: (0,1); Symmetric: 1- sig(a) = sig(-a)

Gradient: sig(a)’ = sig(a)(1-sig(a))

\* Sigmoid is equivalent to softmax

A close up of a logo

Description automatically generated.

We can interpret it as the probability of y=1.

**Cross Entrophy:**

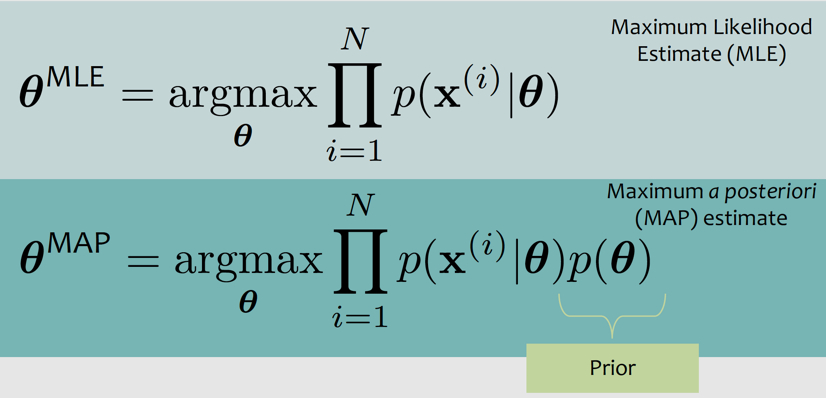
A screenshot of a cell phone

Description automatically generated

no close-form solution, need to use gradient descent

**Naïve Bayes:**

**Naïve Bayes Assumption: Conditional independence of features**

A close up of a logo

Description automatically generated

**NN**

**activation function**

• Threshold t 𝑧 = 𝕀[𝑧 ≥ 0]

• Sigmoid 𝜎 𝑧 = 1/(1 +exp(−𝑧))

• Tanh tanh 𝑧 = 2𝜎(2𝑧) – 1

• Generalizations of ReLU gReLU 𝑧 = max 𝑧, 0 + 𝛼 min{𝑧, 0}

• Leaky-ReLU 𝑧 = max{𝑧, 0} + 0.01 min{𝑧, 0}

•Parametric-ReLU 𝑧 : 𝛼 learnable

Regularization:

1. Data Augment

2. Early Stopping 3. Dropout 4. Batch Normalization