supervised learning: learning model from label

unsupervised learning: extract meaningful information without label (clustering / dimensionality reduction)

Reinforced learning: improve the performance based on interaction with environment. The feedback is a measure of how well the action was measured by a reward function.

Decision Tree	non
Linear SVM	linear
SVM with kernel	non
Linear Regression	a classifier
Logistic Regression	linear
Naive Bayes	linear

Linear Regression: Least Square Fitting

minimize: $\sum_{i}^{n}[y_i - (\beta_0 + \beta_1 x_i)]^2$

$$S_{xy} = \sum_{i=1}^{n} x_i y_i - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)$$

$$S_{xx} = \sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2$$

$$S_{yy} = \sum_{i=1}^{n} y_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} y_i\right)^2$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \qquad \hat{\beta}_1 = \frac{S_{xy}}{S_{xx}}$$

Multiple. $\hat{\beta} = (X^T X)^{-1} X^T Y$

Decision Trees:

when have N attributes, $2^{(2^N)}$ trees

Choose feature $X_i = H(Y|X_i) \downarrow$

$I(X;Y) = H(Y) - H(Y|X) \uparrow$

$H(X) = -\sum p(x) \log_2 p(x)$

$H(Y|X) = \sum p(x) H(Y|X=x)$

$I(X,Y) = \sum\sum P_{X,Y}(x,y) \log\left(\frac{P_{X,Y}(x,y)}{P_X(x) P_Y(y)}\right)$

$\text{Gini} = 1 - \sum P_j^2$

Classification Error $= 1 - \max P_j$

Decision Tree Depth $\uparrow \rightarrow$ overfitting

Ensemble learning - Random Forest

reduce overfitting and variance without decreasing performance

Bagging - Bootstrap Aggregating

Bootstraping: Random sampling with replacement

train multiple decision trees & search all features to split on for each tree

Aggregating: Combine multiple predictions via averaging or majority vote

Support Vector Machines

kernel: mapping to higher-dimensional space

complexity depends on the number of training samples, not dimensionality

larger margin — lower error

$\phi(w) = w^T w$ is minimized

Soft margin is allowed errors.

$\phi(w) = w^T w + \boxed{C} \sum \xi_i$

penalty $C \rightarrow 0$ underfitting

$C \rightarrow \infty$ overfitting

SVM Weakness:

① sensitive to noise

② Standard SVM only consider 2 classes
↓ build multiple SVMs

③ select a specific kernel and parameters is usually done by see and try

Naive Bayes $\rightarrow$ MAP

$$\arg\max_y P(y|x) = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

{ Logistic Regression directly compute $P(y|x)$
{ Naive Bayes use Bayes Therom to compute $P(y|x)$

NB + Gaussian Basis Function

$\approx$ LR + sigmoid

Logistic Regression model the $P(y|x)$ as a logistic function. The logit is a weighted linear combination of the features. linear regression itself is based on linear feature function to regress.

$\rightarrow$ Or using total probability:

$P(S) = P(S|yes)P(yes) + P(S|no)P(no) = (2/9)(9/14) + (3/5)(5/14) = \frac{5}{14}$

NB: $P(yes|S,W) = P(S,W|yes)P(yes)/P(S,W) = \frac{10}{37}$

NB: $P(S,W|yes) = P(S|yes)P(W|yes) = \frac{6}{81}$

$\rightarrow P(S,W) = P(S,W|yes) + P(S,W|no)P(no)$

$= P(S|yes)P(W|yes)P(yes) + P(S|no)P(W|no)P(no)$

$= \frac{37}{210}$

NB: input features x_i is independent given label Y

LR: Least square is not suitable, Maximum likelihood Estimation instead

Performance Metrix

Actual value

Predicted value		
	TP	FP
	FN	TN

(TPR) Recall: $\frac{TP}{FN+TP}$ when false negatives is catastrophic. e.g. disease detection

(PPV) Precision: $\frac{TP}{TP+FP}$ when being right (positive prediction is correct) outweighs detecting all positives. e.g. recommendation system

$NPV = \frac{TN}{TN+FN}$

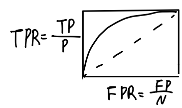
① Training set:
↓
② Validation set: hyperparameter tuning and model selection
↓
③ Test data

K-fold cross validation is used when we have little data

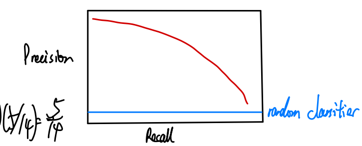
F1-score: $2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

(TNR) Specificity $= \frac{TN}{TN + FP}$

ROC Curves: the trade-off between TP rate and FP rate



PR Curve: between TP rate and positive predictive value



Occam's Razor

Given 2 models with similar generalization errors, the simple model is preferred.

Probabilistic classifiers: yields a probability distribution. includes: Random forest, NB, LR

Deterministic classifiers: no model $\rightarrow$ seperate feature space includes: Decision Tree, SVM

Generative classifiers: learn $P(x,y)$, caculate $P(y|x)$ to find max $P(y)$ NB, Bayesian Network, Hidden Markov

Discriminative classifiers: learn $P(y|x)$ directly LR, SVM

NN: supervised learning

perceptron : $y = \sigma\left(b + \sum_{i=1}^{m} w_i x_i\right) = \{0,1\}$

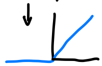
Multilayer perceptron: $y_j = \sigma\left(\sum_{i=1}^{m} w_{ij} x_i + b_j\right)$

↳ feed forward network

Common Activation Functions:

Step / Sigmoid / tanh / ReLu / Leaky ReLu

Maxout / ELU



NN = Function Approximation

Feedforward NN : no loops input → hidden layers → output

Recurrent : use feedback

RNN: ① employ feedback
not necessarily stable
② forecasting time series data
language translation

Hopfield Networks: fully connected

CNN → Image

Transformer Networks: Natural language Processing

Generative Adversarial Network:

Single perceptron can find linear max margin ✓
LR solution ✓

NB → interest in P(labels|features)

B: $P(S|(F_1,F_2)) = P((F_1,F_2)|S) \times P(S) / P(F_1,F_2)$

NB: $P(S|(F_1,F_2)) = P((F_1,F_2)|S) \times P(S) / P(F_1,F_2)$
$= \dfrac{P(F_1|S) \times P(F_2|S) \times P(S)}{P(F_1) \times P(F_2)}$

NN optimization: ① Gradient Descent
② Backpropagation

Computing & Intelligence drive IoT

IoT Architecture:

Application Domain ⟷	Network D ⟷	Device D
app	M2M	Device
...	Communication Network	M2M Area Network
	Access Network	IoT Gateway

↑ Bottom End : Basic, Resource constrained, environment monitor
In the Middle: support localization, full protocol, in Home, industrial
↓ Top End: Military / Medical use

Technical Issue: ① Naming, Addressing, Routing
② Scalability for network architecture
③ Power saving & management
④ Security

Challenges: ① Stringent Latency Requirements
② Network Bandwidth Constrain
③ Resource Constrained Device