Machine Learning Cheatsheet

Summary of Machine Learning Algorithms descriptions, advantages and use cases. Inspired by the very good book and articles of *MachineLearningMastery*, with added math, and *ML Pros & Cons* of *HackingNote*. Design inspired by *The Probability Cheatsheet* of W. Chen. Written by Rémi Canard.

General

Definition

We want to learn a target function ***f*** that maps input variables ***X*** to output variable ***Y***, with an error ***e***:

$$Y = f(X) + e$$

Linear, Nonlinear

Different algorithms make different assumptions about the shape and structure of f, thus the need of testing several methods. Any algorithm can be either:

- **Parametric** (or **Linear**): simplify the mapping to a known linear combination form and learning its coefficients.

- **Non parametric** (or **Nonlinear**): free to learn any functional form from the training data, while maintaining some ability to generalize.

Linear algorithms are usually simpler, faster and requires less data, while Nonlinear can be are more flexible, more powerful and more performant.

Supervised, Unsupervised

Supervised learning methods learn to predict Y from X given that the data is labeled.

Unsupervised learning methods learn to find the inherent structure of the unlabeled data.

Bias-Variance trade-off

In supervised learning, the prediction error ***e*** is composed of the **bias**, the **variance** and the **irreducible** part.

Bias refers to **simplifying assumptions** made to learn the target function easily.

Variance refers to sensitivity of the model to changes in the training data.

The **goal of parameterization** is to achieve a **low bias** (underlying pattern not too simplified) and **low variance** (not sensitive to specificities of the training data) **tradeoff**.

Underfitting, Overfitting

In statistics, ***fit*** refers to how well the target function is approximated.

Underfitting refers to poor inductive learning from training data and poor generalization.

Overfitting refers to learning the training data detail and noise which leads to poor generalization. It can be **limited** by using resampling and defining a validation dataset.

Optimization

Almost every machine learning method has an optimization algorithm at its core.

Gradient Descent

Gradient Descent is used to **find the coefficients** of f that **minimizes a cost function** (for example MSE, SSR).

Procedure:

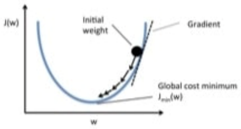
→ Initialization $\theta = 0$ (coefficients to 0 or random)

→ Calculate cost $J(\theta) = evaluate(f(coefficients))$

→ Gradient of cost $\frac{\partial}{\partial\theta j}J(\theta)$ we know the uphill direction

→ Update coeff $\theta j = \theta j - \alpha\frac{\partial}{\partial\theta j}J(\theta)$ we go downhill

The cost updating process is repeated until convergence (minimum found).



Batch Gradient Descend does summing/averaging of the cost over all the observations.

Stochastic Gradient Descent apply the procedure of parameter updating for each observation.

Tips:

- Change **learning rate** α ("size of jump" at each iteration)

- Plot *Cost vs Time* to assess learning rate performance

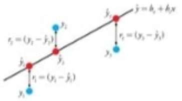
- Rescaling the input variables

- Reduce passes through training set with SGD

- Average over 10 or more updated to observe the learning trend while using SGD

Ordinary Least Squares

OLS is used to find the estimator $\hat{\beta}$ that **minimizes the sum of squared residuals:** $\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 = y - X\hat{\beta}$



Using linear algebra such that we have $\hat{\beta} = (X^TX)^{-1}X^Ty$

Maximum Likelihood Estimation

MLE is used to find the estimators that **minimizes the likelihood function:**

$\mathcal{L}(\theta|x) = f_\theta(x)$ density function of the data distribution

Linear Algorithms

All linear Algorithms assume a linear relationship between the input variables X and the output variable Y.

Linear Regression

Representation:

A LR model representation is a linear equation:

$$y = \beta_0 + \beta_1 x_1 + \dots + \beta_i x_i$$

β_0 is usually called intercept or **bias** coefficient. The dimension of the hyperplane of the regression is its **complexity.**

Usecase example:

- Prediction of customer churn

Nonlinear Algorithms

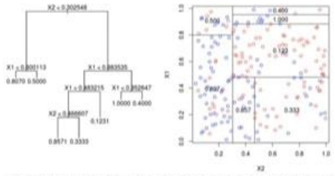
All Nonlinear Algorithms are non-parametric and more flexible. They are not sensible to outliers and do not require any shape of distribution.

Classification and Regression Trees

Also referred as CART or Decision Trees, this algorithm is the foundation of Random Forest and Boosted Trees.

Representation:

The model representation is a **binary tree**, where each **node** is an **input variable** x with a split point and each **leaf** contain an **output variable** y for prediction.



The model actually **split the input space** into (hyper) rectangles, and predictions are made according to the **area** observations *fall* into.

Learning:

Learning of a CART is done by a greedy approach called **recursive binary splitting** of the input space:

At each step, the best **predictor** X_j and the best **cutpoint** s are selected such that $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$ **minimizes the cost**.

- For **regression** the cost is the **Sum of Squared Error:**

$$\sum_{i=1}^{n}(y_i - \hat{y})^2$$

- For **classification** the cost function is the **Gini index:**

$$G = \sum_{i=1}^{n} p_k(1 - p_k)$$

The Gini index is **an indication of how *pure* are the leaves,** if all observations are the same type G=0 (perfect purity), while a 50-50 split for binary would be G=0.5 (worst purity).

The most common **Stopping Criterion** for splitting is a minimum of **training observations per node**.

The simplest form of pruning is **Reduced Error Pruning:** Starting at the leaves, each node is replaced with its most popular class. If the prediction accuracy is not affected, then the change is kept

Advantages:

+ Easy to interpret and no overfitting with pruning

+ Works for both regression and classification problems

+ Can take any type of variables without modifications, and do not require any data preparation

Usecase examples:

- Fraudulent transaction classification

- Predict human resource allocation in companies

Naive Bayes Classifier

Naive Bayes is a **classification** algorithm interested in selecting the **best hypothesis *h* given data *d*** assuming there is no interaction between features.

Representation:

The representation is the based on Bayes Theorem:

$$P(h|d) = \frac{P(d|h) \times P(h)}{P(d)}$$

with naïve hypothesis $P(h|d) = P(x_1|h) \times \dots \times P(x_i|h)$

The prediction is the **Maximum A posteriori Hypothesis:**

$$MAP(h) = \max(P(h|d)) = \max(P(d|h) \times P(h))$$

The denominator is not kept as it is only for normalization.

Learning:

Training is **fast** because only **probabilities** need to be calculated:

$$P(h) = \frac{instances_h}{all\ instances} \text{ and } P(x|h) = \frac{count(x \wedge h)}{instances_h}$$

Variations:

Gaussian Naive Bayes can extend to numerical attributes by assuming a Gaussian distribution.

Instead of $P(x|h)$ are calculated with $P(h)$ during **learning**:

$$\mu(x) = \frac{1}{n}\sum_{i=1}^{n} x_i \text{ and } \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu(x))^2}$$

and MAP for **prediction** is calculated using Gaussian PDF

$$f(x|\mu(x), \sigma) = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Data preparation:

- Change numerical inputs to categorical (binning) or near-Gaussian inputs (remove outliers, log & boxcox transform)

- Other distributions can be used instead of Gaussian

- Log-transform of the probabilities can avoid overflow

- Probabilities can be updated as data becomes available

Advantages:

+ Fast because of the calculations

+ If the naive assumptions works can converge quicker than other models. Can be used on smaller training data.

+ Good for few categories variables

Usecase examples:

- Article classification using binary word presence

- Email spam detection using a similar technique

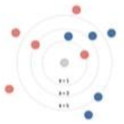
K-Nearest Neighbors

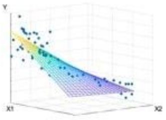
If you are similar to your neighbors, you are one of them.

Representation:

KNN uses the **entire training set, no training** is required.

Predictions are made by searching the ***k* similar instances,** according to a **distance**, and **summarizing the output**.





Learning:

Learning a LR means estimating the coefficients from the training data. Common methods include **Gradient Descent** or **Ordinary Least Squares**.

Variations:

There are extensions of LR training called **regularization** methods, that aim to **reduce the complexity** of the model:

- **Lasso Regression**: where OLS is modified to minimize the sum of the coefficients (L1 regularization)

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}|\beta_j| = RSS + \lambda\sum_{j=1}^{p}|\beta_j|$$

- **Ridge Regression**: where OLS is modified to minimize the squared sum of the coefficients (L2 regularization)

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}\beta_j{}^2 = RSS + \lambda\sum_{j=1}^{p}\beta_j{}^2$$

where $\lambda \geq 0$ is a tuning parameter to be determined.

Data preparation:

- Transform data for linear relationship (ex: log transform for exponential relationship)
- Remove noise such as outliers
- Rescale inputs using standardization or normalization

Advantages:

+ Good regression baseline considering simplicity

+ Lasso/Ridge can be used to avoid overfitting

+ Lasso/Ridge permit feature selection in case of collinearity

Usecase examples:

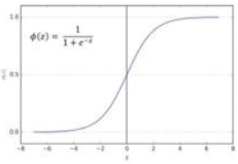
- Product sales prediction according to prices or promotions
- Call-center waiting-time prediction according to the number of complaints and the number of working agents

Logistic Regression

It is the go-to for **binary classification.**

Representation:

Logistic regression a linear method but predictions are transformed using the **logistic function** (or sigmoid):



ϕ is *S*-shaped and map real-valued number in (0,1).

The representation is an equation with binary output:

$$y = \frac{e^{\beta_0+\beta_1 x_1+\cdots+\beta_i x_i}}{1 + e^{\beta_0+\beta_1 x_1+\cdots+\beta_i x_i}}$$

Which actually models the probability of default class:

$$p(X) = \frac{e^{\beta_0+\beta_1 x_1+\cdots+\beta_i x_i}}{1 + e^{\beta_0+\beta_1 x_1+\cdots+\beta_i x_i}} = p(Y = 1|X)$$

Learning:

Learning the Logistic regression coefficients is done using **maximum-likelihood estimation**, to predict values close to 1 for default class and close to 0 for the other class.

Data preparation:

- Probability transformation to binary for classification
- Remove noise such as outliers

Advantages:

+ Good classification baseline considering simplicity

+ Possibility to change cutoff for precision/recall tradeoff

+ Robust to noise/overfitting with L1/L2 regularization

+ Probability output can be used for ranking

Usecase examples:

- Customer scoring with probability of purchase
- Classification of loan defaults according to profile

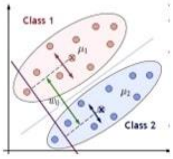
Linear Discriminant Analysis

For **multiclass classification**, LDA is the preferred linear technique.

Representation:

LDA representation consists of **statistical properties** calculated for **each class**: **means** and the **covariance matrix:**

$$\mu_k = \frac{1}{n_k}\sum_{i=1}^{n} x_i \text{ and } \sigma^2 = \frac{1}{n-K}\sum_{i=1}^{n}(x_i - \mu_k)^2$$



LDA assumes **Gaussian** data and attributes of **same σ^2**.

Predictions are made using **Bayes Theorem:**

$$P(Y = k|X = x) = \frac{P(k)\times P(x|k)}{\sum_{l=1}^{K} P(l)\times P(x|l)}$$

to obtain a discriminate function (latent variable) for each class *k*, estimating $P(x|k)$ with a Gaussian distribution:

$$D_k(x) = x \times \frac{\mu_k}{\sigma^2} - \frac{\mu_k{}^2}{2\sigma^2} + \ln(P(k))$$

The class with largest **discriminant value** is the **output class**.

Variations:

- **Quadratic DA**: Each class uses its own variance estimate
- **Regularized DA**: Regularization into the variance estimate

Data preparation:

- Review and modify univariate distributions to be Gaussian
- Standardize data to $\mu = 0, \sigma = 1$ to have same variance
- Remove noise such as outliers

Advantages:

+ Can be used for dimensionality reduction by keeping the latent variables as new variables