DSA5205 Data Science for Quantitative Finance

AY2024/25 Sem1 By Zhao Peiduo

Introduction and Background

Not Examinable

Big Data and "V"s

Big Data is characterized by several "V"s:

- **Volume**: Massive amounts of data
- **Velocity**: The speed of data processing
- **Variety**: Different types of data (structured/unstructured)
- **Veracity**: The quality or trustworthiness of the data
- **Value**: The potential benefit derived from analyzing the data

Machine Learning Overview

- **Supervised Learning**: Learn a function from labeled data
- **Unsupervised Learning**: Find patterns without labeled data
- **Reinforcement Learning**: Learn from rewards and punishments

Quantitative Finance

Quantitative finance uses statistical and mathematical models to analyze financial markets and manage risks. Common models include:

- **CAPM**: Capital Asset Pricing Model
- **Black-Scholes**: Option pricing model

The Data Science Process

1. Problem definition
2. Data collection
3. Data cleaning and preprocessing
4. Model building (Machine learning, statistics)
5. Evaluation and interpretation
6. Reporting and visualization

Challenges in Data Science

- **Data Cleaning**: Handling missing values, outliers
- **Model Selection**: Choosing the right model
- **Ambiguity**: Dealing with uncertainty in data

Distribution and Risks

Moments

Let X be a random variable. The k^{th} moment of X is defined as: $E(X^k)$

The first moment is the expectation. The k^{th} central moment is: $\mu_k = E[(X - E(X))^k]$

Variance:

Skewness

Skewness is a measure of symmetry. For a continuous random variable Y: $Sk(Y) = E\left[\frac{(Y-E(Y))^3}{\sigma^3}\right]$

For a discrete random variable: $Sk(Y) = \sum \frac{(y-E(Y))^3}{\sigma^3} f(y)$

Kurtosis

Kurtosis measures tail thickness. The kurtosis of a random variable Y is: $Kur(Y) = E\left[\frac{(Y-\mu_Y)^4}{\sigma_Y^4}\right]$

For a normal distribution $Y \sim N(\mu, \sigma^2)$, the kurtosis is 3. The excess kurtosis is: $Kur(Y) - 3$

Heavy-Tailed Distributions

A distribution is heavy-tailed if its tails are thicker than the normal distribution. A right Pareto tail has the form: $f(y) \sim Ay^{-(a+1)} \quad \text{as} \quad y \to \infty$

The parameter a is called the tail index.

Student's t-Distributions

The probability density function (pdf) of the t-distribution with ν degrees of freedom is: $f(x) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\Gamma\left(\frac{\nu}{2}\right)}\left(1+\frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}}$

Mean: $E[X] = 0$ for $\nu > 1$

Variance: $Var(X) = \frac{\nu}{\nu-2}$ for $\nu > 2$

Quantile-Based Measures

Quantiles of a distribution: Let $X \sim F(x)$. The p^{th} quantile of $F(x)$ is defined as: $q_p = F^{-1}(p)$

Interquartile range (IQR): $IQR = q_{0.75} - q_{0.25}$

Risk Measures

Value at Risk (VaR) is defined as: $VaR_\alpha = q_\alpha(F_L) \quad \text{such that} \quad P(L \geq VaR_\alpha) = \alpha$

Expected Shortfall (ES) is: $ES_\alpha = E(L|L \geq VaR_\alpha)$

Jarque-Bera Test

The Jarque-Bera test is a test of normality based on skewness and kurtosis. For a sample $Y_1, \ldots, Y_n$:

$$Sk = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \bar{Y}}{s}\right)^3$$

$$Kur = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \bar{Y}}{s}\right)^4$$

$$JB = n\left(\frac{Sk^2}{6} + \frac{(Kur-3)^2}{24}\right)$$

Under the null hypothesis of normality, the test statistic JB follows a chi-squared distribution with 2 degrees of freedom.

Heavy-Tailed and Skewed Distributions

A generalized error distribution (GED) with shape parameter ν is used for modeling heavy tails and skewness:

$$f_{\text{GED}}(y|\nu) = k(\nu)\exp\left(-\frac{1}{2}\left|\frac{y}{\lambda_\nu}\right|^\nu\right), \quad -\infty < y < \infty$$

The tail behavior is controlled by ν, with smaller values indicating heavier tails.

Profile Likelihood

Profile likelihood is used for constructing confidence intervals by fixing one parameter at a time and maximizing over the others:

$$L_{\max}(\theta_1) = \max_{\theta_2} L(\theta_1, \theta_2)$$

Generalized Error Distributions (GED)

The Generalized Error Distribution (GED) has a flexible tail weight controlled by the shape parameter ν.

If $Y \sim GED(\mu, \sigma, \nu)$, its pdf is:

$$f_{\text{GED}}(y|\nu) = k(\nu)\exp\left(-\frac{1}{2}\left|\frac{y}{\lambda_\nu}\right|^\nu\right)$$

The distribution is symmetric about μ, with $\nu = 2$ corresponding to the normal distribution, and $\nu = 1$ being the double-exponential. The tail weight increases as ν decreases.

Quantile-Quantile (QQ) Plot

QQ plots are used to compare the quantiles of a dataset to those of a theoretical distribution, typically the normal distribution.

If the data follows the theoretical distribution, the points should lie approximately on the 45-degree line. Deviations indicate skewness, heavy tails, or other non-normal behavior.

Fisher Information

Fisher information is defined as the expected value of the negative second derivative of the log-likelihood function.

For a parameter θ:

$$I(\theta) = -E\left[\frac{\partial^2}{\partial\theta^2}\log L(\theta)\right]$$

Fisher information quantifies the amount of information that an observable random variable Y carries about an unknown parameter θ. The standard error of an estimator is inversely proportional to the square root of the Fisher information.

Central Limit Theorem for the MLE

The CLT for the Maximum Likelihood Estimator (MLE) states:

$$\hat{\theta} \sim N\left(\theta, I(\hat{\theta})^{-1}\right)$$

Large-sample confidence interval for the MLE of $\hat{\theta}$ is given by:

$$\hat{\theta} \pm s_{\hat{\theta}} z_{\alpha/2}$$

Use the observed Fisher information to approximate $I(\theta)$:

$$I^{obs}(\theta) = -\left[\frac{d^2}{d\theta^2}\log L(\theta)\right]$$

The standard error is given by:

$$s_{\hat{\theta}}^{obs} = \frac{1}{\sqrt{I_{obs}(\theta)}}$$

Replace $s_{\hat{\theta}}^{A}$ with $s_{\hat{\theta}}^{obs}$ in the confidence interval for more convenience and accuracy. Fisher information matrix for the multivariate case is replaced by the Hessian matrix.

MLE has a small bias:

$$BIAS(\hat{\theta}_{ML}) = E(\hat{\theta}_{ML}) - \theta \sim -\frac{A}{n}, \quad \text{as } n \to \infty$$

Variance of the MLE:

$$VAR(\hat{\theta}_{ML}) \sim \frac{B}{n}, \quad \text{as } n \to \infty$$

Likelihood Ratio Tests (LRT)
Suppose that θ is a parameter vector,

$$H_0 : \theta = \theta_0 \quad vs \quad H_1 : \theta \neq \theta_0$$

The rejection region is the set of possible samples that lead us to reject H_0. We need to keep the probability of a type I error below a prespecified small value called the significance level α.
Let $\hat{\theta}_{ML}$ be the MLE without restrictions (full model) and $\hat{\theta}_{0,ML}$ be the value of θ that maximizes $L(\theta)$ under H_0 (reduced model). The likelihood ratio test rejects H_0 if:

$$2\left[\log L(\hat{\theta}_{ML}) - \log L(\hat{\theta}_{0,ML})\right] \geq c$$

where c is a critical value which gives a level that is exactly equal to α or $c = \chi^2_{\alpha,m}$, the α-upper quantile value of the chi-squared distribution with m degrees of freedom.

Robust Estimation
Robust estimators are resistant to outliers and deviations from model assumptions.
An example is the trimmed mean, where a proportion of the largest and smallest values are excluded. Another is the Median Absolute Deviation (MAD), which is a robust measure of scale:

$$\hat{\sigma}_{MAD} = 1.4826 \times \text{median}\left(|Y_i - \text{median}(Y)|\right)$$

MAD is less sensitive to extreme values than standard deviation.

Parsimonious
Tradeoff between bias and variance. The optimal situation is little bias without excessive parameters.

AIC and BIC
The Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) are used to compare models. Both criteria balance goodness of fit with model complexity:

$$AIC = -2\log L(\hat{\theta}_{\text{MLE}}) + 2p$$

$$BIC = -2\log L(\hat{\theta}_{\text{MLE}}) + p\log n$$

Where p is the number of parameters and n is the sample size. Lower values of AIC or BIC indicate better models, but BIC penalizes complexity more than AIC.

Variance-Covariance Method
In the variance-covariance method, the loss distribution is assumed to be multivariate normal. The linearized loss is approximated by:

$$L \sim N(\mu, \sigma^2)$$

Value at Risk (VaR) and Expected Shortfall (ES) are estimated from the distribution. For normal distribution:

$$\hat{VaR}_\alpha = \mu + \sigma\Phi^{-1}(1-\alpha)$$

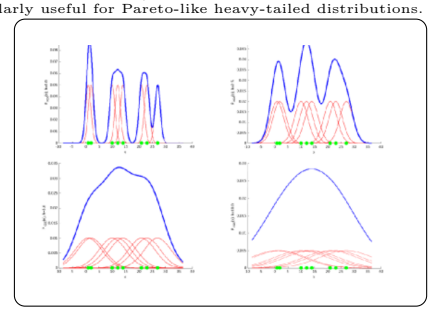
$$\hat{ES}_\alpha = \mu + \sigma\frac{\phi(\Phi^{-1}(1-\alpha))}{\alpha}$$

where ϕ is the standard normal density and Φ^{-1} is the inverse cumulative distribution function (quantile) of the normal distribution.
Hill Estimator
The Hill estimator is used to estimate the tail index of heavy-tailed distributions. For order statistics $X_{(1)}, X_{(2)}, \cdots, X_{(n)}$, the Hill estimator is defined as:

$$\hat{\gamma}_H = \frac{1}{k}\sum_{i=1}^{k}\log\left(\frac{X_{(i)}}{X_{(k+1)}}\right)$$

The parameter k determines how many of the extreme values (largest observations) are used. A Hill plot helps to determine the appropriate value of k. The Hill estimator is particularly useful for Pareto-like heavy-tailed distributions.

Bandwidth Selection for Smoothing Kernels:
- **High** bandwidth (h): Leads to *over-smoothing*, which can mask the data distribution.
- **Low** bandwidth (h): Results in spiky and *hard-to-interpret* density estimation.



Nonparametric Methods and Resampling

Histogram
A histogram divides the sample space into bins and approximates the density at each bin's center:

$$p_H(X) = \frac{\text{Number of } x(k) \text{ in the same bin as } x}{\text{Width of bin}}$$

Hyperparameters: bin width and starting position of the first bin.
Kernel Density Estimation (KDE)
KDE estimates the probability density of a random variable, where $K(u)$ is the kernel function, and h is the bandwidth (smoothing parameter):

$$\hat{p}_{KDE}(x) = \frac{1}{nh^D}\sum_{i=1}^{n}K\left(\frac{x-x_i}{h}\right)$$

Non-Parametric Density Estimation
The general expression for non-parametric density estimation becomes:

$$p(x) = \frac{k}{NV}$$

V	volume surrounding x
N	total number of examples
k	number of examples inside V

Parzen Windows
For Parzen window estimation, the kernel function is:

$$K(u) = \begin{cases} 1 & \text{if } |u| \leq \frac{1}{2}, \forall j = 1 \ldots D \\ 0 & \text{otherwise.} \end{cases}$$

To understand the role of the kernel function, we compute the expectation of the estimate $p_{KDE}(X)$:

$$\mathbb{E}[p_{KDE}(x)] = \frac{1}{Nh^D}\sum_{n=1}^{N}\mathbb{E}\left[K\left(\frac{x-x^{(n)}}{h}\right)\right] = \frac{1}{h^D}\mathbb{E}\left[K\left(\frac{x-x^{(n)}}{h}\right)\right] = \frac{1}{h^D}\int K\left(\frac{x-x'}{h}\right)p(x')\,dx'.$$

- Where we have assumed that vectors $x^{(n)}$ are drawn independently from the true density $p(x)$.
- We can see that the expectation of $p_{KDE}(X)$ is a convolution of the true density $p(x)$ with the kernel function.
  - Thus, the kernel width h plays the role of a smoothing parameter: the wider h is, the smoother the estimate $p_{KDE}(X)$.
  - For $h \to 0$, the kernel approaches a Dirac delta function, and $p_{KDE}(X)$ approaches the true density.
  - However, in practice, we have a finite number of observations, so h cannot be made arbitrarily small since the density estimate $p_{KDE}(X)$ would then degenerate to a set of impulses located at the training data points.

Smooth Kernels
The Parzen window yields density estimates that have discontinuities and weights equally all points x_i, regardless of their distance to the estimation point. For these reasons, the Parzen window is commonly replaced with a smooth kernel function $K(u)$
Let $K(u)$ be Unimodal pdf, such as the Gaussian, then its corresponding density estimate is:

$$K(x) = (2\pi)^{-D/2}e^{-1/2x'x}$$

$$p_{KDE}(x) = \frac{1}{Nh^D}\sum_{n=1}^{N}K\left(\frac{x-x^{(k)}}{h}\right)$$

Usually, but not always, $K(u)$ will be radially symmetric and $\int_{\mathbb{R}^D}K(x)dx = 1$
Bandwidth Selection
The optimal bandwidth h minimizes the mean squared error (MSE) between the KDE and the true density:

$$MSE = \mathbb{E}[(\hat{p}_{KDE}(x) - p(x))^2] = \text{Bias}^2 + \text{Variance}.$$

Assume a standard density function and find the value of the bandwidth that minimizes the integral of the square error (MISE):

$$h_{\text{MISE}} = \arg\min\left\{\mathbb{E}\left[\int\left(p_{KDE}(x) - p(x)\right)^2 dx\right]\right\}.$$

The optimal bandwidth for a Gaussian kernel is:

$$h^* = 1.06 \cdot \sigma \cdot n^{-1/5}$$

Or we can use the more robust inter quantile range (IQR) instead of sample variance:

$$h^* = 0.9AN^{-1/5} \quad \text{where} \quad A = \min\left(\sigma, \frac{\text{IQR}}{1.34}\right)$$

Maximum Likelihood Cross-validation

- The ML estimate of h is degenerate since it yields $h_{ML} = 0$, a density estimate with Dirac delta functions at each training data point
- A practical alternative is to maximize the "pseudo-likelihood"computed using leave-one-out cross-validation

$$h^* = \arg\max \left\{ \frac{1}{N} \sum_{n=1}^{N} \log p_{-n} \left(x^{(n)} \right) \right\}$$

Where

$$p_{-n} \left(x^{(n)} \right) = \frac{1}{(N-1)h} \sum_{m=1, m \neq n}^{N} K \left(\frac{x^{(n)} - x^{(m)}}{h} \right)$$

Multivariate Density Estimation

$$p_{KDE}(x) = \frac{1}{Nh^D} \sum_{n=1}^{N} K \left(\frac{x - x^{(n)}}{h} \right)$$

- Notice that the bandwidth h is the same for all the axes, so this density estimate will weight all the axes equally.
- If one or several of the features has larger spread than the others, we should use a vector of smoothing parameters or even a full covariance matrix, which complicates the procedure.

There are two basic alternatives to solve the scaling problem without having to use a more general KDE:

- Pre-scaling each axis (normalize to unit variance, for instance).
- Pre-whitening the data (linearly transform so $\Sigma = I$), estimate the density, and then transform back [Fukunaga].
  - The whitening transform is $y = \Lambda^{-1/2} M^T x$, where Λ and M are the eigenvalue and eigenvector matrices of Σ.
  - Fukunaga's method is equivalent to using a hyper-ellipsoidal kernel.

Product Kernels

$$p_{PKDE}(x) = \frac{1}{N} \sum_{i=1}^{N} K(x, x^{(n)}, h_1, \ldots, h_D)$$

where

$$K(x, x^{(n)}, h_1, \ldots, h_D) = \frac{1}{h_1 \ldots h_D} \prod_{d=1}^{D} K_d \left(\frac{x_d - x_d^{(n)}}{h_d} \right)$$

- The product kernel consists of the product of one-dimensional kernels:
  - Typically, the same kernel function is used in each dimension ($K_d(x) = K(x)$), and only the bandwidths are allowed to differ.
  - Bandwidth selection can then be performed with any of the methods presented for univariate density estimation.
- Note that although $K(x, x^{(n)}, h_1, \ldots, h_D)$ uses kernel independence, this does not imply we assume the features are independent:
  - If we assumed feature independence, the density estimation (DE) would have the expression:

$$p_{\text{FEAT-IND}}(x) = \prod_{d=1}^{D} \frac{1}{Nh_d} \sum_{i=1}^{N} K_d \left(\frac{x_d - x_d^{(i)}}{h_d} \right).$$

Transformation KDE Algorithm

1. Start with data $Y_1, \ldots, Y_n$.
2. Transform the data to $X_1 = g(Y_1), \ldots, X_n = g(Y_n)$; g is monotonic.
3. Let f_X be the usual KDE using $X_1, \ldots, X_n$.
4. $f_Y(y) = f_X \{g(y)\} \left| g'(y) \right|$. Plugging in $y = g^{-1}(x)$, then

$$f_Y(g^{-1}(x)) = f_X \{x\} \left| g'(g^{-1}(x)) \right|$$

K-fold Cross-validation in Detail

Let the K parts be $C_1, C_2, \ldots, C_K$, where C_k denotes the indices of the observations in part k. There are n_k observations in part k: if n is a multiple of K, then $n_k = n/K$.

Compute:

$$CV_{(K)} = \sum_{k=1}^{K} \frac{n_k}{n} MSE_k$$

where $MSE_k = \sum_{i \in C_k} (y_i - \hat{y}_i)^2 / n_k$ and $\hat{y}_i$ is the fit for observation i, obtained from the data with part k removed. Setting $K = n$ yields n-fold or *leave-one-out cross-validation* (LOOCV).

With least-squares linear or polynomial regression, the following formula holds:

$$CV_{(K)} = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{y_i - \hat{y}_i}{1 - h_i} \right)^2$$

where $\hat{y}_i$ is the ith fitted value from the original least squares fit, and h_i is the leverage (diagonal of the "hat"matrix; see ESL book for details). This is like the ordinary MSE, except the ith residual is divided by $1 - h_i$.

Multivariate Methods and Factor Models

Covariance Matrices

Covariance measures the direction but not the strength of a linear relationship between two random variables X and Y. The covariance matrix of a random vector $\mathbf{Y} = (Y_1, \ldots, Y_d)$ is defined as:

$$\text{COV}(\mathbf{Y}) = \mathbb{E}\left[(\mathbf{Y} - \mathbb{E}[\mathbf{Y}])(\mathbf{Y} - \mathbb{E}[\mathbf{Y}])^\top \right].$$

The diagonal elements are the variances of individual components.
The correlation $\rho_{X,Y}$ is defined as:

$$\rho_{X,Y} = \text{Cor}(X, Y) = \frac{\text{Cov}(X, Y)}{\text{SD}(X)\text{SD}(Y)} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y} = \text{Scaled Covariance}$$

Linear Combinations of Random Variables

Let $w = (w_1, \ldots, w_d)^\top$ be a vector of weights, and $Y = (Y_1, \ldots, Y_d)^\top$ be a random vector.
Let Z be a weighted average of the components of Y as

$$Z = w^\top Y = \sum_{i=1}^{d} w_i Y_i$$

Then

$$E(Z) = w^\top E(Y) \quad \text{and} \quad \text{Var}(Z) = w^\top \text{Cov}(Y) w.$$

Let Y_1 and Y_2 be two random vectors of dimensions n_1 and n_2, respectively. Then $\text{Cov}(Y_1, Y_2)$ is defined as an $n_1 \times n_2$ matrix whose (i, j)-th element is the covariance between the i-th component of Y_1 and the j-th component of Y_2. Furthermore:

$$\text{Cov}(w_1^\top Y_1, w_2^\top Y_2) = w_1^\top \text{Cov}(Y_1, Y_2) w_2$$

for constant vectors w_1 and w_2 of lengths n_1 and n_2.
If $Y_1, \ldots, Y_d$ are independent, or at least uncorrelated, then:

$$\text{Var}(w^\top Y) = \text{Var}\left(\sum_{i=1}^{d} w_i Y_i \right) = \sum_{i=1}^{d} w_i^2 \text{Var}(Y_i).$$

Examples:

- If $w^\top = \left(\frac{1}{d}, \ldots, \frac{1}{d} \right)$, then $w^\top Y = \overline{Y} \quad \Longrightarrow \quad \text{Var}(\overline{Y}) = \frac{1}{d^2} \sum_{i=1}^{d} \text{Var}(Y_i)$
- $\text{Var}(Y_1 - Y_2) = \text{Var}(Y_1) + \text{Var}(Y_2)$.

Multivariate Normal Distribution

A random vector $\mathbf{Y} = (Y_1, \ldots, Y_d)$ follows a multivariate normal distribution if its probability density function (PDF), where, $\boldsymbol{\mu}$ is the mean vector and Σ is the covariance matrix:

$$\phi_d(\mathbf{y}; \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{y} - \boldsymbol{\mu})^\top \Sigma^{-1} (\mathbf{y} - \boldsymbol{\mu}) \right).$$

Multivariate t-Distribution

The random vector Y has a multivariate $t_\nu(\mu, \Lambda)$ distribution if:

$$Y = \mu + \sqrt{\frac{\nu}{W}} Z$$

where:

- W is chi-squared distributed with ν degrees of freedom.
- $Z \sim N(0, \Lambda)$ is multivariate normal with zero mean and covariance Λ.
- W and Z are independent.

Key Properties for Multivariate t-Distribution:

- $\nu > 1$: μ is the mean vector.
- $0 < \nu \leq 1$: The expectation of Y does not exist. Here, μ is the center of density ellipses and the mode.
- $\nu > 2$: The covariance matrix of Y exists, and is given by:

$$\Sigma = \frac{\nu}{\nu - 2} \Lambda$$

  - Λ is also called the scale matrix.
  - Y and Z have the same correlation matrices.

Univariate Case: If $Y \sim t_\nu(\mu, \Lambda)$, then $w^\top Y$ has a univariate t-distribution with:

- **Mean:** $w^\top \mu$,
- **Variance:** $\frac{\nu}{\nu - 2} w^\top \Lambda w$.

Elliptically Contoured Densities

A d-variate multivariate density f is elliptically contoured if it can be expressed as:

$$f(y) = [\det(\Lambda)]^{-1/2} g((y - \mu)^T \Lambda^{-1} (y - \mu))$$

- g is a nonnegative-valued, decreasing function of $x \geq 0$, and $1 = \int g(\|y\|^2) dy$
- μ is the mean vector
- The covariance matrix Σ is a scalar multiple of Λ

For each fixed $c > 0$, $\mathcal{E}(c) = \{y : (y - \mu)^T \Lambda^{-1} (y - \mu) = c\}$ is an ellipse centered at μ, and if $c_1 > c_2$, then $\mathcal{E}(c_1)$ is inside $\mathcal{E}(c_2)$.

Eigenvalue-Eigenvector Decomposition to Find the Axes

$$\Sigma = O\,\mathrm{diag}(\lambda_i)O^T$$

where O is an orthogonal matrix whose columns are the eigenvectors of Σ and $\lambda_1, \ldots, \lambda_d$ are the eigenvalues of Σ.

- R can be found using the function `eigen()`.
- If o_i is the i-th column of O, the i-th axis of $\mathcal{E}(c)$ is:

$$\{\mu + ko_i : -\infty < k < \infty\}$$

- The axes are mutually perpendicular.

If the parameter set θ is an m-dimensional vector, then the Fisher information matrix is an $m \times m$ square matrix.

Fisher Information Matrix

Definition: $I(\theta)$ denotes the Fisher information matrix. The (i, j)-th entry is:

$$I_{i,j}(\theta) = -E\left[\frac{\partial^2}{\partial\theta_i \partial\theta_j} \log\{L(\theta)\}\right]$$

The standard errors are the square roots of the diagonal entries of the inverse of the Fisher information matrix.

Properties Under suitable assumptions, for **large enough sample sizes**, the MLE is approximately normally distributed with mean equal to the true parameter vector and with covariance matrix equal to the inverse of the Fisher information matrix.

Computation The computation of the expectation in $I(\theta)$ is challenging. Use the **observed Fisher information matrix**:

$$I^{\text{obs}}_{i,j}(\theta) = -\frac{\partial^2}{\partial\theta_i \partial\theta_j} \log\{L(\theta)\}$$

- Inverting the $I^{\text{obs}}(\theta)$ allows obtaining standard errors of the MLE.

Copulas

Suppose that $\mathbf{Y} = (Y_1, \ldots, Y_d)$ has a multivariate CDF $F_{\mathbf{Y}}$ with continuous marginal univariate CDFs $F_{Y_1}, \ldots, F_{Y_d}$. Then each of $F_{Y_1}(Y_1), \ldots, F_{Y_d}(Y_d)$ is distributed Uniform$(0, 1)$, and thus the CDF of $(F_{Y_1}(Y_1), \ldots, F_{Y_d}(Y_d))$ is a copula.

- This CDF is called the **copula** of $\mathbf{Y}$ and is denoted by $C_{\mathbf{Y}}$.
- $C_{\mathbf{Y}}$ contains all information about dependencies among the components of $\mathbf{Y}$.
- $C_{\mathbf{Y}}$ has no information about the marginal CDFs of $\mathbf{Y}$.
- All d-dimensional copula functions $C_{\mathbf{Y}}$ have domain $[0, 1]^d$ and range $[0, 1]$.

By definition of a CDF:

$$C_{\mathbf{Y}}(u_1, \ldots, u_d) = P\left(F_{Y_1}(Y_1) \le u_1, \ldots, F_{Y_d}(Y_d) \le u_d\right) = P\left(Y_1 \le F^{-1}_{Y_1}(u_1), \ldots, Y_d \le F^{-1}_{Y_d}(u_d)\right)$$

$$= F_{\mathbf{Y}}(F^{-1}_{Y_1}(u_1), \ldots, F^{-1}_{Y_d}(u_d))$$

Next, letting $u_j = F_{Y_j}(y_j)$ for $j = 1, \ldots, d$, we see that:

$$F_{\mathbf{Y}}(y_1, \ldots, y_d) = C_{\mathbf{Y}}(F_{Y_1}(y_1), \ldots, F_{Y_d}(y_d))$$

The density associated with $C_{\mathbf{Y}}$:

$$c_{\mathbf{Y}}(u_1, \ldots, u_d) = \frac{\partial^d}{\partial u_1 \cdots \partial u_d} C(u_1, \ldots, u_d)$$

The density of $\mathbf{Y}$:

$$f_{\mathbf{Y}}(y_1, \ldots, y_d) = c_{\mathbf{Y}}(F_{Y_1}(y_1), \ldots, F_{Y_d}(y_d)) f_{Y_1}(y_1) \cdots f_{Y_d}(y_d)$$

where $f_{Y_1}(y_1), \ldots, f_{Y_d}(y_d)$ are the univariate marginal densities of $Y_1, \ldots, Y_d$, respectively.

Special Copulas

- **The d-dimensional independence copula C_0:**
  - The CDF of d mutually independent $U \sim$ Uniform$(0, 1)$ random variables: $C_0(u_1, \ldots, u_d) = u_1 \cdots u_d$
  - $c_0(u_1, \ldots, u_d) = 1$ on $[0, 1]^d$, and 0 elsewhere.
- **The d-dimensional co-monotonicity copula C_+**(Perfect Positive Dependence):
  - The CDF of $\mathbf{U} = (U, \ldots, U)$ where $U \sim$ Uniform$(0, 1)$:

$$C_+(u_1, \ldots, u_d) = P(U \le u_1, \ldots, U \le u_d) = P(U \le \min(u_1, \ldots, u_d)) = \min(u_1, \ldots, u_d)$$

  - C_+ is an upper bound for all copula functions:

$$C(u_1, \ldots, u_d) \le C_+(u_1, \ldots, u_d) \quad \forall u_1, \ldots, u_d \in [0, 1]^d.$$

- **The two-dimensional counter-monotonicity copula C_-:**
  - Characterizes perfect negative dependence.
  - The CDF of $(U, 1 - U)$:

$$C_-(u_1, u_2) = P(U \le u_1, 1 - U \le u_2) = P(1 - u_2 \le u_1) = \max(u_1 + u_2 - 1, 0)$$

  - All two-dimensional copula functions are bounded below by C_-.
  - A counter-monotonicity copula cannot have $d > 2$.

Gaussian Copulas

- **Definition** Gaussian copula is based on a multivariate normal distribution with a $d \times d$ correlation matrix Ω. The copula depends only on the dependencies within $\mathbf{Y}$ (via Ω), not on the univariate marginal distributions.
- **Properties:**
  - Denoted as $C_{\text{Gaussian}}(u_1, \ldots, u_d \mid \Omega)$.
  - If $\mathbf{Y}$ follows a Gaussian copula, it is said to have a **meta-Gaussian distribution**, and the marginal distributions can be arbitrary.
  - **Special cases:**
    - When Ω is the identity matrix, the copula reduces to the **d-dimensional independence copula** C_0.
    - When all correlations in Ω approach $+1$, the copula converges to the **co-monotonicity copula** C_+.
    - When $d = 2$ and pairwise correlations converge to -1, the copula converges to the **counter-monotonicity copula** C_-.
- **Applications:** Gaussian copulas model dependencies while allowing flexibility in marginal distributions.

Archimedean Copulas

- **Definition** Based on a strict generator function φ, Archimedean copulas have the form:

$$C_\varphi(u_1, \ldots, u_d) = \varphi^{-1}(\varphi(u_1) + \cdots + \varphi(u_d)).$$

- **Properties of the Generator φ:**
  - φ is continuous, strictly decreasing, and convex, mapping $[0, 1] \to [0, \infty)$.
  - $\varphi(1) = 0$, $\varphi(0) = \infty$.
  - $\varphi'(t) < 0$.
- **Special Cases:**
  - The independence copula C_0 is a special Archimedean copula with $\varphi(u) = -\log(u)$.
  - Archimedean copulas are symmetric and suitable for modeling pairwise and higher-dimensional dependencies.
- **Examples of Archimedean Copulas:**
  - Frank Copula
  - Clayton Copula
  - Gumbel Copula
  - Joe Copula
- **Applications:** Archimedean copulas are useful in bivariate and high-dimensional cases, where all pairwise dependencies share similar structures.

Rank Correlation

- **Definition:** A rank correlation measures the relationship between rankings of different variables.
- **Characteristics:**
  - It is a non-parametric method and depends only on the ranks, not the actual values.
  - **Advantages:**
    - Robust against outliers.
    - Can capture non-linear monotonic relationships.
  - Does not require normally distributed data.
- **Common Methods:**
  - Kendall's Tau (τ)
  - Spearman's Rank Correlation Coefficient (ρ)

Kendall's Tau (τ)

- **Definition:** Measures the strength of association between two variables by comparing the concordant and discordant pairs of data.
- **Formula:**

$$\tau = \frac{\text{Number of Concordant Pairs} - \text{Number of Discordant Pairs}}{\binom{n}{2}}$$

  where $\binom{n}{2}$ is the total number of possible pairs.
- **Definitions:**
  - **Concordant Pair:** Two observations (x_i, y_i) and (x_j, y_j) are concordant if the ranks agree (both increase or both decrease).
  - **Discordant Pair:** Two observations (x_i, y_i) and (x_j, y_j) are discordant if the ranks disagree (one increases and the other decreases).
- **Special Cases:**
  - $\tau = 1$: Perfect agreement in ranks.
  - $\tau = -1$: Perfect disagreement in ranks.
  - $\tau = 0$: No association.
- **Sample Kendall's Tau:**

$$\hat{\tau} = \frac{2}{n(n-1)} \sum_{i<j} \mathrm{sgn}(x_i - x_j) \cdot \mathrm{sgn}(y_i - y_j)$$

  where $\mathrm{sgn}(\cdot)$ is the sign function.

Spearman's Rank Correlation Coefficient (ρ)

- **Definition:** Measures the rank correlation between two variables, assessing the strength and direction of a monotonic relationship.
- **Formula:**

$$\rho = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2 - 1)}$$

  where d_i is the difference between the ranks of corresponding variables, and n is the number of observations.
- **Special Cases:**
  - $\rho = 1$: Perfect positive monotonic relationship.
  - $\rho = -1$: Perfect negative monotonic relationship.
  - $\rho = 0$: No monotonic relationship.
- **Applications:**
  - Suitable for ordinal data or when assumptions of linearity and normality are violated.
  - Can be used for detecting and quantifying non-linear monotonic relationships.

Curse of Dimensionality High-dimensional data often leads to sparsity, making patterns and structures hard to identify. Moreover, computational complexity increases exponentially with the number of dimensions and distance metrics (e.g., Euclidean distance) lose effectiveness in high dimensions. Dimensionality reduction techniques, such as PCA are used to extract a smaller set of features that capture most of the information in the data.

Principal Components Analysis (PCA)

- **Definition:** PCA is a dimensionality reduction technique that transforms the data into a new set of variables (principal components), which are uncorrelated and ordered by the amount of variance they capture in the data.
- **Objectives:**
  - Reduce the dimensionality of data while retaining as much variance as possible.
  - Identify new axes (principal components) that maximize the variance in the data.
- **How It Works:**
  - The data is centered (mean-subtracted) to have zero mean.
  - The covariance matrix of the data is computed.
  - **Eigen-decomposition:** Eigenvalues and eigenvectors of the covariance matrix are calculated:

$$\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$$

    where:
    - $\mathbf{\Lambda}$: Diagonal matrix of eigenvalues.
    - $\mathbf{V}$: Matrix of eigenvectors (principal components).
  - Principal components corresponding to the largest eigenvalues are selected.
  - The original data is projected onto the selected principal components.

PCA: Ordering

- The Principal Component (PC) transformation is defined as:

$$Y = \Gamma^T (X - \mu_X)$$

  where X is the original data, μ_X is the mean of X, and Γ is the matrix of eigenvectors.
- Suppose $\text{var}(X) = \Sigma$, where Σ is the covariance matrix. Let Λ and Γ be the diagonal eigenvalue matrix and the matrix of eigenvectors of Σ, respectively. Then:

$$\Sigma = \Gamma\Lambda\Gamma^T$$

- The variance of the transformed data Y is:

$$\text{var}(Y) = \text{var}(\Gamma^T X) = \Gamma^T \text{var}(X)\Gamma = \Gamma^T \Sigma\Gamma = \Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_p)$$

  where $\lambda_1, \lambda_2, \ldots, \lambda_p$ are the eigenvalues of Σ.
- Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$. Maximizing the variance of $\delta^T X$ leads to the choice $\delta_1 = \gamma_1$, where γ_1 is the eigenvector corresponding to the largest eigenvalue λ_1 of Σ.
- The variances of the PCs are:

$$\text{var}(Y_1) = \lambda_1, \quad \text{var}(Y_2) = \lambda_2, \quad \ldots, \quad \text{var}(Y_p) = \lambda_p$$

- Key properties of PCs:
  - PCs have zero means: $\text{E}(Y_j) = 0$.
  - Variance of the j-th PC: $\text{var}(Y_j) = \lambda_j$.
  - PCs are uncorrelated.
- Since $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$, we have:

$$\text{var}(Y_1) \geq \text{var}(Y_2) \geq \cdots \geq \text{var}(Y_p)$$

Regression & Prediction

Linear Regression

Linear regression models the relationship between the outcome Y and predictors $X_1, X_2, \ldots, X_d$:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_d X_d + \epsilon$$

The coefficients β are estimated by minimizing the sum of squared residuals (errors):

$$J(\beta) = \frac{1}{2n} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$

Gradient Descent

To optimize the cost function $J(\beta)$, gradient descent iteratively updates the parameters β as follows, where α is the learning rate. Gradient updates:

$$\beta_j \leftarrow \beta_j - \alpha \frac{\partial J}{\partial \beta_j}$$

For Linear Regression:

$$\frac{\partial J}{\partial \beta_j} = \frac{1}{n} \sum_{i=1}^{n} (h_\beta(X^{(i)}) - Y^{(i)}) X_j^{(i)}$$

Close Form Solution for Linear Regression:

$$\theta = \left(X^T X\right)^{-1} X^T y$$

Polynomial Regression

Extends linear regression by allowing higher-order terms, which increases flexibility but can lead to overfitting:

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \ldots + \beta_p X^p + \epsilon$$

Logistic Regression

- **Objective:** Predict the probability of an instance belonging to a class.
- **Sigmoid Function:** The model uses the sigmoid (logistic) function:

$$g(z) = \frac{1}{1 + e^{-z}}$$

- **Logistic Regression Model:**

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

- **Interpretation:**
  - Output $h_\theta(x)$ represents the probability that the instance belongs to the positive class ($y = 1$).
  - A decision threshold is typically applied to classify instances, e.g., $h_\theta(x) \geq 0.5 \implies y = 1$.

Deriving the Cost Function via Maximum Likelihood Estimation

- The likelihood function for m examples:

$$\mathcal{L}(\theta) = \prod_{i=1}^{m} (h_\theta(x^{(i)}))^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}}$$

- The log-likelihood is:

$$\ell(\theta) = \sum_{i=1}^{m} \left[y^{(i)} \log\left(h_\theta(x^{(i)})\right) + (1 - y^{(i)}) \log\left(1 - h_\theta(x^{(i)})\right) \right]$$

- The cost function (to minimize):

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log\left(h_\theta(x^{(i)})\right) + (1 - y^{(i)}) \log\left(1 - h_\theta(x^{(i)})\right) \right]$$

Gradient Descent for Regularized Logistic Regression

- The cost function with regularization:

$$J_{\text{reg}}(\theta) = -\sum_{i=1}^{n} \left[y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log\left(1 - h_\theta(x^{(i)})\right) \right] + \frac{\lambda}{2} \left\| \theta_{[1:d]} \right\|_2^2$$

  where:
  - $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$ is the hypothesis function (sigmoid).
  - λ is the regularization parameter.
  - $\theta_{[1:d]}$ refers to all parameters except the intercept θ_0.
- **Goal:** Minimize $J_{\text{reg}}(\theta)$ with respect to θ.
- **Initialization:** Initialize the parameter vector θ.
- **Update Rule:** Repeat until convergence: $\theta_j \leftarrow \theta_j - \alpha \frac{\partial J_{\text{reg}}(\theta)}{\partial \theta_j}$
- **Detailed Updates:** (Simultaneous update for $j = 0, \ldots, d$)
  - For $j = 0$ (no regularization for the intercept term):

$$\theta_0 \leftarrow \theta_0 - \alpha \sum_{i=1}^{n} (h_\theta(x^{(i)}) - y^{(i)})$$

  - For $j = 1, \ldots, d$ (with regularization):

$$\theta_j \leftarrow \theta_j - \alpha \left[\sum_{i=1}^{n} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \lambda \theta_j \right]$$

Linear Basis Function Models

- **Polynomial Basis Functions:**

$$\phi_j(x) = x^j$$

  - These are **global** basis functions; a small change in x affects all basis functions.

- **Gaussian Basis Functions:**

$$\phi_j(x) = \exp\left(-\frac{(x-\mu_j)^2}{2s^2}\right)$$

  - These are **local** basis functions; a small change in x only affects nearby basis functions.
  - Parameters μ_j and s control location and scale (width), respectively.

- **Sigmoidal Basis Functions:**

$$\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right), \quad \sigma(a) = \frac{1}{1+\exp(-a)}$$

  - These are also **local** basis functions; a small change in x only affects nearby basis functions.
  - Parameters μ_j and s control location and scale (slope), respectively.

- **Basic Linear Model:**

$$h_\theta(x) = \sum_{j=0}^{n} \theta_j x_j$$

- **Generalized Linear Model:**

$$h_\theta(x) = \sum_{j=0}^{n} \theta_j \phi_j(x)$$

By replacing data with outputs of the basis functions, fitting the generalized model is the same as fitting the basic model. Unless using kernel tricks (as in support vector machines), cluttering with basis functions is unnecessary.

Linear Splines

- A **linear spline** with knots at ξ_k, $k = 1, \ldots, K$, is a piecewise linear polynomial that is continuous at each knot. The model can be represented as:

$$y = \beta_0 + \beta_1 x + \beta_2 b_2(x) + \cdots + \beta_{K+1} b_{K+1}(x) + \epsilon$$

where the $b_k(x)$ are **basis functions** defined as:

$$b_1(x) = x, \quad b_{k+1}(x) = (x-\xi_k)_+ \quad \text{for } k = 1, \ldots, K$$

- The $(\cdot)_+$ denotes the positive part function:

$$(x-\xi_k)_+ = \begin{cases} x-\xi_k, & \text{if } x > \xi_k \\ 0, & \text{otherwise} \end{cases}$$

Cubic Splines

- A **cubic spline** with knots at ξ_k, $k = 1, \ldots, K$, is a piecewise cubic polynomial with continuous derivatives up to order 2 at each knot.
- The model can be represented as:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 b_4(x) + \cdots + \beta_{K+3} b_{K+3}(x) + \epsilon$$

where the $b_k(x)$ are **truncated power basis functions** defined as:

$$b_{k+3}(x) = (x-\xi_k)_+^3 \quad \text{for } k = 1, \ldots, K$$

- The $(\cdot)_+$ is the positive part function:

$$(x-\xi_k)_+ = \begin{cases} x-\xi_k, & \text{if } x > \xi_k \\ 0, & \text{otherwise} \end{cases}$$

Model Selection Techniques

Subset Selection

Subset selection identifies a subset of predictors that best relate to the response. The best subset is chosen based on criteria like Residual Sum of Squares (RSS), R^2, AIC, BIC, or cross-validation error (C_p).

Best Subset Selection:

1. Start with the null model M_0 which has no predictors.
2. For each k, fit models with exactly k predictors and select the one with the smallest RSS or highest R^2.
3. Use cross-validation or another criterion to choose the best model among all.

Stepwise Selection

Stepwise selection simplifies the search process to avoid overfitting and high variance of the coefficient estimates:

- **Forward Stepwise:** Start with no predictors and iteratively add the one that improves the model the most based on smallest RSS or highest R^2. After p iterations we will end up with $M_0, \ldots, M_p$ models, and we use AIC, BIC, cross validation error or adjusted R^2 to select the best model.
- **Backward Stepwise:** Start with all predictors and iteratively remove the least useful. Again we end up with $M_p, \ldots, M_0$ and we select the best using AIC, BIC, cross validation error or adjusted R^2.

Model Selection Techniques

Regularization / Shrinkage Methods

Shrinkage methods penalize the model's complexity, reducing variance:

- **Ridge Regression**
  - **Objective Function:**

$$J(\theta) = \frac{1}{2n}\sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2}\sum_{j=1}^{p}\theta_j^2$$

    where:
    - The first term measures how well the model fits the data.
    - The second term is the **regularization term**.
    - $\lambda \geq 0$ is a **tuning parameter**, to be determined separately.
    - There is no regularization on θ_0.
  - **Key Points:**
    - The second term is called the **shrinkage penalty**, which ensures that $\theta_1, \ldots, \theta_p$ are close to zero, shrinking the estimates of θ toward zero.
    - The tuning parameter λ controls the relative impact of these two terms on the regression coefficients:
      - $\lambda = 0$: No regularization (equivalent to ordinary least squares).
      - $\lambda > 0$: Increasing λ reduces the magnitude of coefficients, improving generalization by reducing overfitting.
    - Selecting a good value for λ is critical; cross-validation is typically used for this purpose.
- **Lasso**
  - Ridge regression does not perform feature selection: it includes all predictors in the final model.
  - The **Lasso** (Least Absolute Shrinkage and Selection Operator) is an alternative that addresses this issue by introducing an ℓ_1-norm regularization.
  - **Objective Function:**

$$J(\theta) = \frac{1}{2n}\sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{p}|\theta_j|$$

    - The first term measures the model fit, while the second term enforces sparsity by shrinking some coefficients to exactly zero.
    - The Lasso performs both regularization and **variable selection**, producing sparse models.
  - **Key Points:**
    - The Lasso uses an ℓ_1-penalty instead of the ℓ_2-penalty in Ridge Regression.
    - Models irrelevant or weak predictors by shrinking their coefficients to exactly zero.
    - Cross-validation is used to determine the optimal value of λ, as in Ridge Regression.

Choosing the Optimal Model

Model selection is based on minimizing test error, often estimated via cross-validation or information criteria like AIC and BIC:

$$AIC = -2\log(L) + 2 \cdot d \quad \text{and} \quad BIC = -2\log(L) + \log(n) \cdot d$$

Here, L is the likelihood of the model, and d is the number of parameters.

Cross-Validation

- Split the data into K folds.
- Train the model on $K-1$ folds and validate on the remaining fold.
- Repeat for each fold and average the validation errors.

This procedure provides a direct estimate of test error.

Time Series

Not Examinable

Time Series Basics

- A **time series** is sequential data indexed over time.
- A **stochastic process** is a theoretical construct for time series, where data points are random variables.

Stationarity

- **Stationarity**:
  - Time-invariant statistical properties.
  - A process is stationary if $P(Y_t = y) = P(Y_{t+k} = y)$.
- **Stationary Process**:
  - Oscillates around a mean (mean-reversion).
  - Example: AR(1) stationarity condition: $-1 < \phi < 1$.

Autocorrelation

- **Autocorrelation Function (ACF)**: Measures linear dependence between observations k-time units apart.
- **Autocovariance**:

$$\gamma_k = \mathbb{E}[(Y_t - \mu)(Y_{t-k} - \mu)]$$

AR and MA Models

- **AR(p)** (Autoregressive Model):

$$Y_t = \delta + \phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p} + \epsilon_t$$

- **MA(q)** (Moving Average Model):

$$Y_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q}$$

- **ARMA(p,q)** (Combination of AR and MA):

$$Y_t = \delta + \phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q}$$

Trend Stationary vs. Difference Stationary

- **Trend Stationary**:

$$Y_t = \mu + \beta t + \epsilon_t$$

Detrend to achieve stationarity.

- **Difference Stationary**:

$$\Delta Y_t = Y_t - Y_{t-1}$$

Differencing removes non-stationarity.

ARIMA Models ARIMA(p,d,q)

- p: Autoregressive lags.
- d: Differencing order.
- q: Moving average lags.

Support Vector Machines

The objective of the support vector machine (SVM) algorithm is to find a hyperplane in a $(d-1)$-dimensional space (where d is the number of features) that distinctly classifies the data points. In other words, a "good"separator maximizing the margin between different classes.

Maximizing Margin

- Increasing the margin reduces model complexity, which helps in better generalization.
- Lesson from learning theory: If the hypothesis space H is constrained in size and/or the training dataset is large, low training error is likely to indicate low generalization error.

Linear Separators A linear separator is defined by a hyperplane: $\theta^T x = 0$. The decision function is given by:$h(x) = \text{sign}(\theta^T x)$

SVM Optimization

- The primal problem is given by:

$$\min_{\theta} \frac{1}{2} \sum_{j=1}^{d} \theta_j^2 \quad \text{s.t.} \quad y_i(\theta^T x_i) \geq 1, \ \forall i$$

- The Lagrangian is used to transform the primal problem into a dual problem:

$$\mathcal{L}(\theta, \alpha) = \frac{1}{2} \sum_{j=1}^{d} \theta_j^2 - \sum_{i=1}^{n} \alpha_i (y_i(\theta^T x_i) - 1)$$

$$\text{with } \alpha_i \geq 0 \quad \forall i$$

- The dual formulation allows the use of kernel functions, which makes the problem computationally feasible in higher dimensions.

SVM Dual Representation Maximize:

$$J(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

Subject to $\alpha_i \geq 0 \quad \forall i$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$. $\langle x_i, x_j \rangle$ can be replaced by $K\langle x_i, x_j \rangle$ for the kernel trick.

Decision Function:

$$h(x) = \text{sign}\left(\sum_{i \in S} \alpha_i y_i \langle x, x_i \rangle \right)$$

Kernel Trick

- SVMs can be extended to non-linear classification by using kernel functions to map the input features into a higher-dimensional space.
- Common kernels include:
  - Polynomial Kernel (set d=1 and c = 0 for linear kernel): $K(x_i, x_j) = (\langle x_i, x_j \rangle + c)^d$
  - Gaussian Kernel (RBF): $K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2} \right)$
  - Sigmoid Kernel: $K(x_i, x_j) = \tanh(\alpha x_i^T x_j + c)$
  - Cosine Similarity Kernel:

$$K(x_i, x_j) = \frac{x_i^T x_j}{\|x_i\| \|x_j\|}$$

  - Chi-squared Kernel:

$$K(x_i, x_j) = \exp\left(-\gamma \sum_k \frac{(x_{ik} - x_{jk})^2}{x_{ik} + x_{jk}} \right)$$

- The kernel trick allows SVMs to create complex decision boundaries without explicitly computing the higher-dimensional feature mappings.

Slack Variables and Soft Margin

- In cases where the data are not linearly separable, slack variables ξ_i are introduced to allow some misclassifications.
- The objective function is modified to minimize both the margin and the penalty for misclassified points:

$$\min_{\theta, \xi} \frac{1}{2} \sum_{j=1}^{d} \theta_j^2 + C \sum_{i=1}^{n} \xi_i \quad \text{s.t.} \quad y_i(\theta^T x_i) \geq 1 - \xi_i, \ \xi_i \geq 0$$

- The parameter C controls the trade-off between maximizing the margin and minimizing classification error.

Tree Based Methods

Tree-based methods involve stratifying or segmenting the predictor space into a number of simple regions.These methods can be used for both regression and classification tasks.

Decision Trees

- Decision trees divide the feature space into axis-parallel (hyper-) rectangles.
- Each internal node represents a test on an attribute, each branch represents an outcome of the test, and each leaf node represents a class label or a continuous value (for regression).

Splitting Criteria

- Decision trees use splitting criteria such as Gini impurity, entropy, or mean squared error to determine the best attribute to split the data.
- extbfGini Impurity: Measures the impurity of a node; lower values indicate better splits.
- extbfEntropy: Measures the uncertainty or impurity; lower entropy after splitting indicates better separation of classes.

Basic Algorithm for Top-Down Induction of Decision Trees

- **Steps:**
  1. Initialize the root as the root of the decision tree.
  2. Main loop:
     - Select A^*, the **best decision attribute** for the next node.
     - Assign A^* as the decision attribute for the current node.
     - For each value of A^*, create a new descendant node.
     - Sort training examples to leaf nodes.
     - If training examples are perfectly classified, stop. Else, recurse over new leaf nodes.
- **Challenge:** How to choose the best attribute for the split?

Expected Information as Uncertainty or Entropy

- For a discrete random variable X with possible outcomes $\{x_1, x_2, \ldots, x_m\}$ and probabilities $P(X = x_i)$, the **entropy** $H(X)$ is defined as:

$$H(X) = -\sum_{i=1}^{m} P(X = x_i) \log_2 P(X = x_i)$$

- Entropy represents the expected (mean or average) amount of information obtained by learning the outcome of X.

Conditional Entropy

- The entropy $H(X)$ of a random variable X is:

$$H(X) = -\sum_{i=1}^{m} P(X = x_i) \log_2 P(X = x_i)$$

- **Specific Conditional Entropy:** The entropy $H(X \mid Y = y)$ of X given $Y = y$ is:

$$H(X \mid Y = y) = -\sum_{i=1}^{m} P(X = x_i \mid Y = y) \log_2 P(X = x_i \mid Y = y)$$

- **Conditional Entropy:** The conditional entropy $H(X \mid Y)$ is:

$$H(X \mid Y) = \sum_{j=1}^{k} P(Y = y_j) H(X \mid Y = y_j)$$

- **Mutual Information (Information Gain):** The information gain between X and Y is:

$$I(X; Y) = H(X) - H(X \mid Y)$$

- Information gain is used to select the attribute that best splits the data at each node in a decision tree.

Bootstrap Bootstrap estimates the uncertainty of an estimator by resampling:

$$SE_B(\hat{\alpha}) = \sqrt{\frac{1}{B-1} \sum_{r=1}^{B} (\hat{\alpha}_r^* - \bar{\alpha}^*)^2}.$$

Bagging (Bootstrap Aggregation)

- Bagging is used to reduce the variance of decision trees by creating multiple bootstrapped datasets from the original training data and fitting a separate decision tree to each.
- The final prediction is made by averaging (for regression) or taking a majority vote (for classification).

Random Forests

- Random forests improve upon bagging by adding an element of randomness: at each split, a random subset of features is considered.
- This helps to reduce correlation between individual trees, leading to better model performance.

Boosting

- Boosting is an ensemble technique that builds trees sequentially, with each tree trying to correct the errors of the previous one.
- Common algorithms include AdaBoost and Gradient Boosting.

Strengths and Weaknesses of Tree-Based Methods

- **Strengths**:
  - Easy to interpret and visualize.
  - Can handle both numerical and categorical data.
  - Non-linear relationships between features do not affect tree performance.
- **Weaknesses**:
  - Prone to overfitting, especially with deep trees.
  - Unstable; small changes in the data can lead to different splits.

Neural Networks

Neural networks are a set of algorithms designed to recognize patterns, inspired by the structure and function of the human brain. They consist of layers of interconnected nodes (neurons) that can learn to approximate complex functions.

Perceptron

- The perceptron is the simplest type of artificial neural network, consisting of a single layer.
- It makes predictions based on a linear combination of input features and a step activation function.

Activation Functions

- Activation functions introduce non-linearity into the network, enabling it to learn complex patterns.
- Common activation functions include:
  - **Step Activation Function:**
$$h_\theta(x) = \begin{cases} 1, & \text{if } \sum \theta_i x_i > b \\ 0, & \text{otherwise} \end{cases}$$
  - **Sigmoid (logistic unit)**: $g(z) = \frac{1}{1+e^{-z}}$
  - **Tanh**: $g(z) = \tanh(z)$
  - **ReLU (Rectified Linear Unit)**: $g(z) = \max(0, z)$
  - **Leaky ReLU**: $g(z) = \max(\alpha z, z)$, where α is a small constant
  - Others: Maxout, ELU

Feedforward Neural Network

- A feedforward neural network consists of an input layer, one or more hidden layers, and an output layer.
- Information flows in one direction, from input to output, without any feedback loops.

Loss Function

- A **loss function** (cost function) measures how well the current model predicts or how far the predictions are from the real answers.
- Common loss functions:
  - Hinge loss
  - Softmax loss
  - Mean Squared Error (MSE)
  - Cross-entropy loss

Example: For $h_\Theta(x) \in \mathbb{R}^K$, where $(h_\Theta(x))_i =$ the i-th output, the loss function $J(\Theta)$ is defined as:

$$J(\Theta) = -\frac{1}{n}\left[\sum_{i=1}^{n}\sum_{k=1}^{K} y_{ik}\log\left(h_\Theta(x_i)\right)_k + (1-y_{ik})\log\left(1-(h_\Theta(x_i))_k\right)\right] + \frac{\lambda}{2n}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\Theta_{ji}^{(l)}\right)^2$$

Training NN via Gradient Descent with Backpropagation

- **Given:** Training set $\{(x_1, y_1), \ldots, (x_n, y_n)\}$
- Initialize all $\Theta^{(l)}$ randomly (not to 0!).
- Loop (each iteration is called an **epoch**):
  - Set $\Delta_{ij}^{(l)} = 0 \quad \forall l, i, j$ (used to accumulate gradients).
  - IF we use mini batch THEN Sample m training instances: $X = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ without replacement
  - For each training instance (x_t, y_t):
    - Set $a^{(1)} = x_t$.
    - Compute $\{a^{(2)}, \ldots, a^{(L)}\}$ via forward propagation.
    - Compute $z^{(L)} = a^{(L)} - y_t$.
    - Compute errors $\delta^{(L-1)}, \ldots, \delta^{(2)}$ via backpropagation.
    - Compute gradient:
$$\Delta_{ij}^{(l)} = \Delta_{ij}^{(l)} + a_j^{(l)}\delta_i^{(l+1)}$$
  - Compute average regularized gradient:
$$D_{ij}^{(l)} = \begin{cases} \frac{1}{m}\Delta_{ij}^{(l)} + \lambda\Theta_{ij}^{(l)}, & \text{if } j \neq 0 \\ \frac{1}{m}\Delta_{ij}^{(l)}, & \text{otherwise} \end{cases}$$
- Update weights via gradient step:
$$\Theta_{ij}^{(l)} = \Theta_{ij}^{(l)} - \alpha D_{ij}^{(l)}$$
- Until all training instances are seen or weights converge or max #epochs is reached.