Pattern Recognition and Machine Learning

Yunlong Song

Ever tried. Ever failed. No matter. Try Again. Fail again. Fail better.
— Samuel Beckett

Introduction

Pattern Recognition: automatic discovery of regularities in data and their use for making predictions.

Machine Learning: a large set of input vectors or a training set is used to tune the parameters of an adaptive model.

Probability Theory

- Sum rule: $p(x) = \sum_y p(x, y)$
- Product rule: $p(x, y) = p(y|x)p(x) = p(x|y)p(y)$
- Bayes' theorem: $p(H|E) = \frac{p(H)p(E|H)}{p(E)}$
- Bayes theorem: posterior $= \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}$
- Probability that x lies in the range $[a, b]$:
  $p(a \le x \le b) = \int_a^b p(x)dx$,
  where $p(x)$ is the probability density function over x.
- Cumulative distribution function:
  $P(z) = p(x \in (-\infty, z)) = \int_{-\infty}^{x} p(x')dx'$
- Expectation: $E[f] = \int f(x)p(x)dx = \sum_x f(x)p(x)$
- Variance: $\text{var}[f] = E[(f(x) - E[f(x)])^2] = E[f^2] - E[f]^2$
- Covariance for two variables x and y:
  $\text{cov}[x, y] = E[(x - E[x])(y - E[y])] = E[xy] - E[x]E[y]$

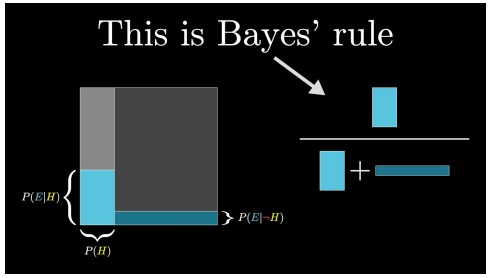

Figure 1: Bayes' theorem. (Source: 3blue1brown)

Gaussian Distribution

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x - \mu)^2\right\}$$

$$\text{Mean:} \quad E[x] = \int x\mathcal{N}(x|\mu, \sigma^2)dx = \mu$$

$$\text{Variance:} \quad \text{var}[x] = E[x^2] - E[x]^2 = \sigma^2$$

I.I.D. Data: Assume that the data points are independently and identically distributed (i.i.d.).

$$p(\mathbf{x}|\mathbf{w}) = \prod_{n=1}^{N} p(x_n|\mathbf{w})$$

$$\text{Log likelihood:} \quad \ln p(\mathbf{X}|\mathbf{w}) = \sum_{n=1}^{N} \ln p(x_n|\mathbf{w})$$

Information Theory

Entropy: The entropy of a discrete random variable x is defined as

$$H[x] = -\sum_x p(x) \ln p(x)$$

$$H[x] \ge 0$$

$H[x] = 0$ if and only if $p(x) = 1$ for some x

Kullback-Leibler Divergence: The Kullback-Leibler divergence between two probability distributions $p(x)$ and $q(x)$ is defined as

$$D_{\text{KL}}(p||q) = -\sum_x p(x) \ln \frac{q(x)}{p(x)}$$

$$D_{\text{KL}}(p||q) \ne D_{\text{KL}}(q||p)$$

$$D_{\text{KL}}(p||q) \ge 0$$

$D_{\text{KL}}(p||q) = 0$ if and only if $p(x) = q(x)$

Probability Distributions

Exponential Family: A probability distribution belongs to the exponential family if it can be written in the following canonical form:

$$f(x|\theta) = h(x) \exp\left\{\eta(\theta)^T T(x) - A(\theta)\right\}$$

where $\eta(\theta)$ is the natural parameter, $T(x)$ is the sufficient statistic, $A(\theta)$ is the log partition function, $h(x)$ is the base measure, and θ is the parameter.

- **Bernoulli distribution:**
  $p(x|\mu) = \mu^x(1 - \mu)^{1-x}$
- **Gaussian distribution:**
  $p(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x - \mu)^2\right\}$
- **Gamma distribution:**
  $p(x|a, b) = \frac{1}{\Gamma(a)} b^a x^{a-1} \exp(-bx)$
- **Beta distribution:**
  $p(x|a, b) = \frac{1}{B(a,b)} x^{a-1}(1 - x)^{b-1}$
- **Dirichlet distribution:**
  $p(\mathbf{x}|\alpha) = \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1)\cdots\Gamma(\alpha_K)} \prod_{k=1}^{K} x_k^{\alpha_k - 1}$

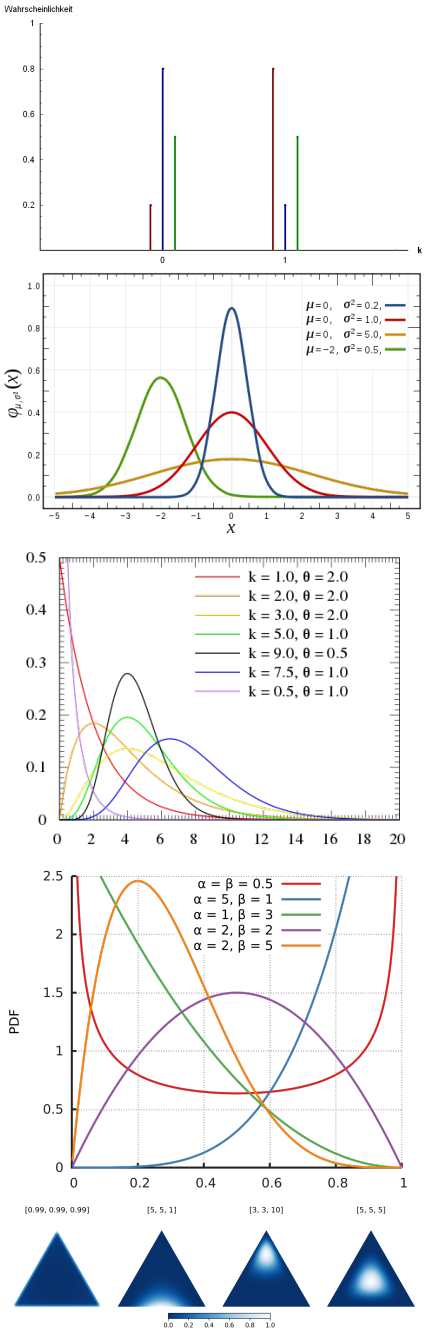

Figure 2: Distributions. (Source: Internet)

Density Estimation: Given a data set $\mathbf{X} = \{x_1, \ldots, x_N\}$, we want to estimate the underlying probability density function $p(x)$.

- **Frequentist approach:** Estimate the density function directly from the data. (e.g., likelihood maximization)
- **Bayesian approach:** Assume a prior distribution over the parameters of the density function. (e.g., MAP estimation)

Maximum Likelihood Estimation: Given a data set $\mathbf{X} = \{x_1, \ldots, x_N\}$ and a density function $p(x|\mathbf{w})$, the log likelihood function is

$$\ln p(\mathbf{X}|\mathbf{w}) = \sum_{n=1}^{N} \ln p(x_n|\mathbf{w})$$

Bayesian Estimation: Given a data set $\mathbf{X} = \{x_1, \ldots, x_N\}$ and a density function $p(x|\mathbf{w})$, the log posterior distribution is

$$\ln p(\mathbf{w}|\mathbf{X}) = \ln p(\mathbf{w}) + \sum_{n=1}^{N} \ln p(x_n|\mathbf{w})$$

Conjugate Priors: If the prior distribution $p(\mathbf{w})$ and the likelihood function $p(\mathbf{X}|\mathbf{w})$ are in the same family, then the posterior distribution $p(\mathbf{w}|\mathbf{X})$ will also be in the same family.

Mixtures of Gaussians: A mixture of Gaussians is a linear combination of K Gaussian distributions:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

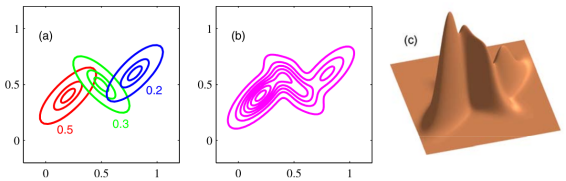

Figure 3: A mixture of 3 Gaussians in 2-D.

Linear Models for Regression

Linear Basis Function Models: A linear model is a linear combination of fixed nonlinear functions of the input variables:

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$$

where $\mathbf{w} = (w_0, \ldots, w_{M-1})^T$ is the weight vector, $\phi(\mathbf{x}) = (\phi_0(\mathbf{x}), \ldots, \phi_{M-1}(\mathbf{x}))^T$ is the feature vector, and $\phi_0(\mathbf{x}) = 1$.

Basis Functions:

- **Polynomial basis functions:** $\phi_j(x) = x^j$
- **Gaussian basis functions:** $\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$
- **Sigmoidal basis functions:** $\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right)$
- **Fourier basis functions:** $\phi_{2j}(x) = \cos(jx)$ and $\phi_{2j+1}(x) = \sin(jx)$

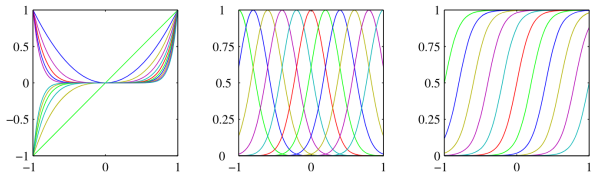

Figure 4: Basis functions.

Maximum Likelihood and Least Squares: Given a data set $\mathbf{X} = \{x_1, \ldots, x_N\}$ and a linear model $y(x, \mathbf{w}) = \mathbf{w}^T \phi(x)$, we assume tha the target variable t is given by

$$t = y(x, \mathbf{w}) + \epsilon$$

where ϵ is the noise term. The likelihood function is

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|\ \mathbf{w}, \beta^{-1}\mathbf{I})$$

where $\mathbf{t} = (t_1, \ldots, t_N)^T$, is the design matrix, β is the precision of the noise term, and $\mathbf{I}$ is the identity matrix.
Making the assumption that the data points are independently and identically distributed (i.i.d.), the log likelihood function is

$$\ln p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \ln \mathcal{N}(t_n|\mathbf{w}^T\phi(x_n), \beta^{-1})$$
$$= -\frac{\beta}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^T\phi(x_n)\}^2 + \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi)$$

Gratient of the log likelihood function:

$$\nabla_{\mathbf{w}} \ln p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \beta \sum_{n=1}^{N}\{t_n - \mathbf{w}^T\phi(x_n)\}\phi(x_n)$$

$$\nabla_{\beta} \ln p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \frac{N}{2\beta} - \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^T\phi(x_n)\}^2$$

Solution:

$$\mathbf{w}_{\mathrm{ML}} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{t}$$

$$\beta_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N}\{t_n - \mathbf{w}_{\mathrm{ML}}^T\phi(x_n)\}^2$$

Regularized Least Squares: The total error function is

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^T\phi(x_n)\}^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

where λ is the regularization coefficient. The solution is

$$\mathbf{w}_{\mathrm{MAP}} = (\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\mathbf{I})^{-1}\mathbf{\Phi}^T\mathbf{t}$$

Sequential Learning: Using the sequential learning approach, we can update the weight vector $\mathbf{w}$ incrementally. The update rule is

$$\mathbf{w}^{\mathrm{new}} = \mathbf{w}^{\mathrm{old}} + \eta\phi(x_n)\{t_n - \mathbf{w}^{\mathrm{old}}\phi(x_n)\}$$

where η is the learning rate.

Bias-Variance Decomposition: The expected prediction error of a model can be decomposed into three terms:

$$\begin{aligned}&\mathrm{E}[(t - y(x))^2]\\ &= \mathrm{E}[(t - \mathrm{E}[y(x)])^2] + \mathrm{E}[(\mathrm{E}[y(x)] - y(x))^2] + \mathrm{E}[(t - \mathrm{E}[t])^2]\\ &= \mathrm{Bias}^2 + \mathrm{Variance} + \mathrm{Noise}\end{aligned}$$

Bayesian Linear Regression: Bayesian linear regression is a probabilistic approach to linear regression. It can avoid overfitting problem of maximum likelihood estimation by introducing a prior distribution over the weight vector $\mathbf{w}$.

Given a data set $\mathbf{X} = \{x_1, \ldots, x_N\}$ and a linear model $y(x, \mathbf{w}) = \mathbf{w}^T\phi(x)$, we assume that the target variable t is given by

$$t = y(x, \mathbf{w}) + \epsilon$$

where ϵ is the noise term. The prior distribution over the weight vector $\mathbf{w}$ is

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$$

where α is the precision of the prior distribution.

The posterior distribution over the weight vector $\mathbf{w}$ is

$$p(\mathbf{w}|\mathbf{X}, \mathbf{t}, \alpha, \beta) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)$$

where

$$\mathbf{m}_N = \beta\mathbf{S}_N\mathbf{\Phi}^T\mathbf{t}$$
$$\mathbf{S}_N^{-1} = \alpha\mathbf{I} + \beta\mathbf{\Phi}^T\mathbf{\Phi}$$

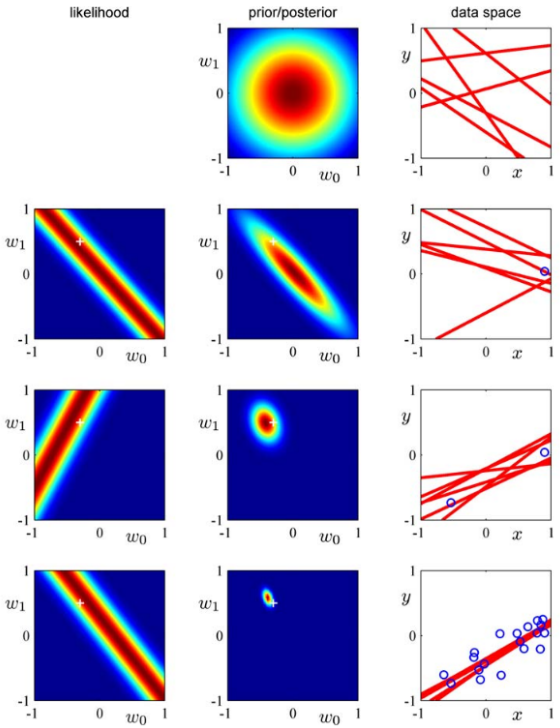

Figure 5: Bayesian linear regression.

Locally Weighted Regression: Locally weighted regression is a non-parametric method for regression. It assigns different weights to different data points based on their distance to the query point. The loss function is

$$J(\mathbf{w}) = \sum_{n=1}^{N} \exp\left(-\frac{(x - x_n)^2}{2\tau^2}\right)(t_n - \mathbf{w}^T\boldsymbol{\phi}(x_n))^2$$

where the weight vector $\mathbf{w}$ is

$$\mathbf{w} = (\mathbf{\Phi}^T\mathbf{W}\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{W}\mathbf{t}$$

and the weight matrix $\mathbf{W}$ is

$$\mathbf{W} = \text{diag}\left\{\exp\left(-\frac{(x - x_n)^2}{2\tau^2}\right)\right\}$$

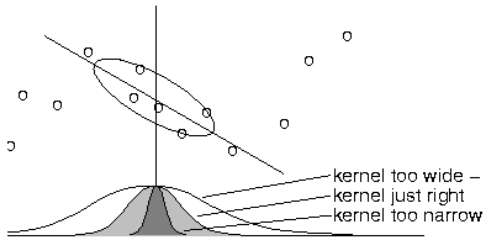

Figure 6: Locally weighted regression.

Classification

Discriminative vs. Generative:

- **Discriminative models:** Model $p(t|\mathbf{x})$ directly (logistic regression). Learn mappings from inputs to classes and focus on the decision boundary. (e.g., logistic regression, softmax regression, support vector machines)
  Objective: $\max L(\theta) = \max \prod_{i=1}^{N} p(t_i|x_i;\theta)$

- **Generative models:** Model $p(\mathbf{x}, t)$ and learn the Model $p(\mathbf{x}|t)$ and $p(t)$ separately. Use Bayes' theorem to compute the posterior probability $p(t|\mathbf{x})$. (e.g., Naive Bayes, Gaussian discriminant analysis)
  Objective: $\max L(\theta) = \max \prod_{i=1}^{N} p(x_i, t_i;\phi,\theta)$

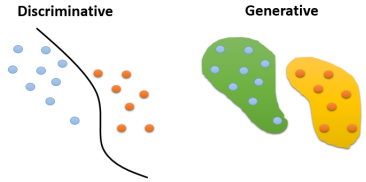

Figure 7: Discriminative vs. Generative models.

Discriminant Functions: Two-Class Case: Given a data set $\mathbf{X} = \{x_1, \ldots, x_N\}$ and a linear model $y(x) = \mathbf{w}^T\mathbf{x} + \omega_0$, we assume that the target variable t is given by

$$t = \begin{cases} +1, & \text{if } y(x, \mathbf{w}) \geq 0 \\ -1, & \text{otherwise} \end{cases}$$

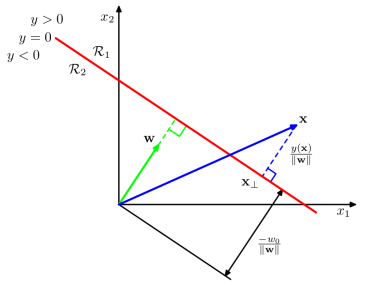

Figure 8: Linear classification.

Logistic Regression:

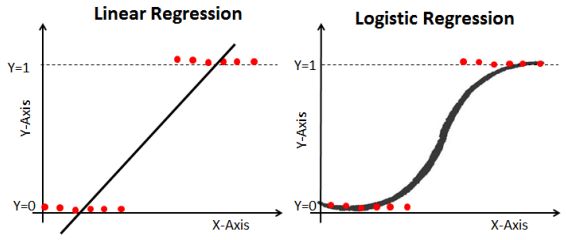

Figure 9: Linear classification.

Logistic regression is a probabilistic approach to classification. It models the posterior probability of the class label t given the input vector $\mathbf{x}$. The logistic sigmoid function is

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

The likelihood function is

$$p(t|\mathbf{x}, \mathbf{w}) = \sigma(t\mathbf{w}^T\mathbf{x})$$

The log likelihood function is

$$\ln p(\mathbf{t}|\mathbf{X}, \mathbf{w}) = \sum_{n=1}^{N} \ln \sigma(t_n\mathbf{w}^T\mathbf{x}_n)$$

The gradient of the log likelihood function is

$$\nabla_{\mathbf{w}} \ln p(\mathbf{t}|\mathbf{X}, \mathbf{w}) = \sum_{n=1}^{N}(t_n - \sigma(\mathbf{w}^T\mathbf{x}_n))\mathbf{x}_n$$

The solution is

$$\mathbf{w}_{\text{ML}} = \text{argmax}_{\mathbf{w}} \ln p(\mathbf{t}|\mathbf{X}, \mathbf{w})$$

Example: Two-Class Logistic Regression Given a data set $\mathbf{X} = \{x_1, \ldots, x_N\}$ and a linear model $y(x) = \mathbf{w}^T\mathbf{x} + \omega_0$, we assume that the target variable t is given by

$$t = \begin{cases} +1, & \text{if } y(x, \mathbf{w}) \geq 0 \\ -1, & \text{otherwise} \end{cases}$$

The log loss function is

$$\ell_k = \begin{cases} -\ln p_k & \text{if } t_k = +1 \\ -\ln(1-p_k), & \text{if } t_k = 0 \end{cases}$$

These two cases can be combined into a single expression:

$$\ell_k = -t_k \ln p_k - (1-t_k)\ln(1-p_k)$$

where $p_k = \sigma(\mathbf{w}^T\mathbf{x}_k)$. This expression is more formally known as the cross-entropy loss function.

Softmax Regression: Softmax regression is a generalization of logistic regression to multiple classes. The softmax function is

$$y_k = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

Cross-entropy loss function:

$$\ell = -\sum_k t_k \ln y_k$$

Gaussian Discriminant Analysis:
Gaussian discriminant analysis is a generative approach to classification. It models the class-conditional densities $p(\mathbf{x}|y)$ and the class priors $p(y)$, and then uses Bayes' theorem to compute the posterior probabilities $p(y|\mathbf{x})$. It assumes that the feature vectors from each class are drawn from a multivariate Gaussian distribution:

$$p(\mathbf{x}|y=k) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

The class priors are

$$p(y=k) = \phi_k$$

The posterior probabilities are

$$p(y=k|\mathbf{x}) = \frac{p(\mathbf{x}|y=k)p(y=k)}{\sum_j p(\mathbf{x}|y=j)p(y=j)}$$

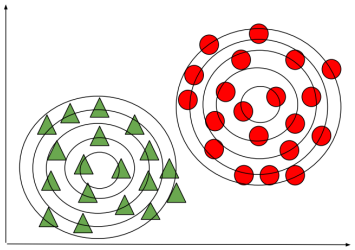

Figure 10: Generative Learning Algorithm (GDA).

Naive Bayes: Naive Bayes is a generative approach to classification. It assumes that the features are conditionally independent given the class label y:

$$p(x_1, x_2, \cdots, x_n|y) = p(\mathbf{x}|y) = \prod_{j=1}^{D} p(x_j|y)$$

Support Vector Machines: Support vector machines (SVMs) are a discriminative approach to classification. They find the hyperplane that maximizes the margin between the classes. The decision function is

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

The margin is

$$\text{margin} = \frac{1}{\|\mathbf{w}\|}$$

The optimization problem is

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{subject to} && t_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1 \end{aligned}$$

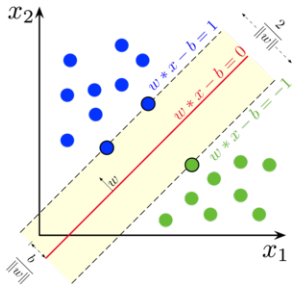

Figure 11: Support vector machines.

Kernel Methods: Kernel methods are a generalization of linear models to nonlinear models. They map the input vectors into a higher-dimensional feature space using a kernel function. The decision function is

$$y(\mathbf{x}) = \sum_{n=1}^{N} \alpha_n k(\mathbf{x}_n, \mathbf{x}) + b$$

Kernel trick: The kernel function $k(\mathbf{x}, \mathbf{x}')$ is a similarity function that computes the inner product of the feature vectors in the higher-dimensional space without explicitly computing the feature vectors. For example, the Gaussian kernel is

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right)$$

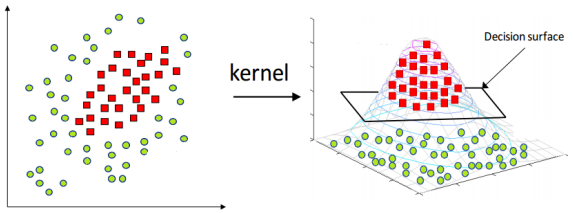

Figure 12: Kernel trick.

Decision Trees A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.

Ensemble Methods Ensemble methods combine multiple models to improve the predictive performance. The most common ensemble methods are

- **Bagging:** Train multiple models on different subsets of the data and average their predictions.
- **Boosting:** Train multiple models sequentially, where each model learns to correct the errors of the previous model.
- **Stacking:** Train multiple models and combine their predictions using another model (meta-learner).

Advice for Applying Machine Learning

Bias-Variance Tradeoff:

- *Bias*: The error due to bias is the difference between the average prediction of the model and the true value. $\text{bias}(\mathbf{x}) = \text{E}[\hat{f}(\mathbf{x})] - f(\mathbf{x})$
- *Variance*: The error due to variance is the variability of a model prediction for a given data point. $\text{var}(\mathbf{x}) = \text{E}[(\hat{f}(\mathbf{x}) - \text{E}[\hat{f}(\mathbf{x})])^2]$
- *High bias:* The model is too simple and underfits the data.
- *High variance:* The model is too complex and overfits the data.

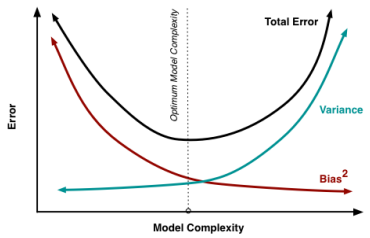

Figure 13: Bias-variance tradeoff.

Regularization: Regularization is a technique to prevent overfitting by adding a penalty term to the loss function. The total error function is

$$E(\mathbf{w}) = \text{loss}(\mathbf{w}) + \lambda\Omega(\mathbf{w})$$

where λ is the regularization coefficient and $\Omega(\mathbf{w})$ is the regularization term. The most common regularization terms are

- *L1 regularization:* $\Omega(\mathbf{w}) = \|\mathbf{w}\|_1$
- *L2 regularization:* $\Omega(\mathbf{w}) = \|\mathbf{w}\|_2^2$

Training, Validation, and Test Sets: The data set is divided into three subsets:

- *Training set:* Used to train the model.
- *Validation set:* Used to tune the hyperparameters of the model.
- *Test set:* Used to evaluate the performance of the model.

Cross-Validation: Cross-validation is a technique to evaluate the performance of a model. The data set is divided into K subsets. The model is trained on $K-1$ subsets and tested on the remaining subset. This process is repeated K times, each time using a different subset as the test set.

Model Selection: Model selection is the process of choosing the best model from a set of models. The most common methods are

- *Grid search:* Exhaustively search over a set of hyperparameters.
- *Random search:* Randomly sample hyperparameters from a distribution.
- *Bayesian optimization:* Use a probabilistic model to model the objective function and select the hyperparameters that maximize the expected improvement.

Empirical Risk Minimization: Empirical risk minimization is the process of minimizing the expected loss over the training set. A function that measures the error between the predicted output of the model and the actual target value. Common examples include the mean squared error for regression and cross-entropy loss for classification.

$$\text{ERM} = \frac{1}{N}\sum_{i=1}^{N}\ell(\hat{y}_i, y_i) \tag{1}$$

where ℓ is the loss function, $\hat{y}_i$ is the predicted output of the model, and y_i is the actual target value.

Unsupervised Learning

K-Means Clustering: K-means clustering is a partitioning method that divides the data set into K clusters. The objective is to minimize the sum of squared distances between the data points and their corresponding cluster centroids.

Given a data set $\mathbf{X} = \{x_1, \ldots, x_N\}$ and the number of clusters K, the K-means algorithm is

1. Initialize the cluster centroids $\mu_1, \ldots, \mu_K$.
2. Assign each data point to the nearest cluster centroid.
3. Update the cluster centroids by taking the mean of the data points in each cluster.
4. Repeat steps 2 and 3 until convergence.

It minimizes the within-cluster sum of squared errors:

$$\text{SSE} = \sum_{k=1}^{K}\sum_{n=1}^{N_k}\|x_n - \mu_k\|^2$$

where N_k is the number of data points in cluster k.

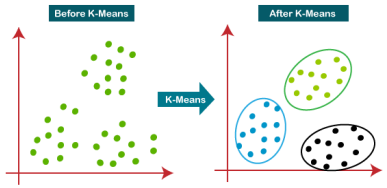

Figure 14: K-means clustering.

Density Estimation: Density estimation is the process of estimating the probability density function of the data. The most common methods are

- *Histogram:* Divide the data into bins and estimate the probability density function using the frequency of the data points in each bin.
- *Kernel density estimation:* Place a kernel function on each data point and sum the kernel functions to estimate the probability density function.
- *Gaussian mixture model:* Model the data as a mixture of Gaussian distributions and estimate the parameters of the Gaussian distributions.

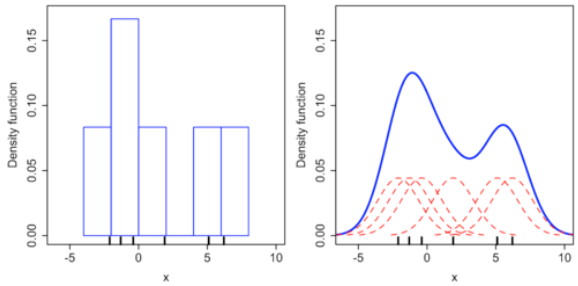

Figure 15: Comparison of histogram (left) and kernel density estimation (right).

Gaussian Mixture Model: A Gaussian mixture model (GMM) is a probabilistic model that represents the data as a mixture of K Gaussian distributions. The likelihood function is

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{k=1}^{K}\pi_k\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Expectation-Maximization Algorithm: The expectation-maximization (EM) algorithm is an iterative method to estimate the parameters of a probabilistic model with latent variables. The algorithm consists of two steps:

1. *Expectation step (E-step):* Compute the expected value of the latent variables given the observed data and the current estimate of the parameters.
2. *Maximization step (M-step):* Update the parameters of the model by maximizing the likelihood function given the observed data and the expected value of the latent variables.

Jensen's Inequality: Jensen's inequality states that the expected value of a convex function of a random variable is greater than or equal to the convex function of the expected value of the random variable.

$$f(\mathrm{E}[X]) \leq \mathrm{E}[f(X)]$$