Module 2 Regression (Ordinary least Squares (OLS) / Polynomial regression / Ridge regression (regularization) / MLE / MAP)

① training dataset $D=\{(\vec{x}_i, y_i)\}_{i=1}^{N}$, $\vec{x}_i \in R^d$ (input feature vector), $y_i \in R$ (output target value); unknown mapping $y=f(\vec{x})$, hypothesis set $h_{\vec{w}}(\vec{x})$, weight vector $\vec{w} \in R^d$ → find $\hat{y}=h_{\vec{w}^*}(\vec{x})$ best approx $y=f(\vec{x})$ (true value)

③ $h_{\vec{w}}(\vec{x}) = w_0 + w_1 x_1 + \cdots + w_d x_d$, $\vec{x} = (1, x_1, \cdots, x_d)^T$ (input 前加 1), $\vec{w} = (w_0, w_1, \cdots, w_d)^T$

$h_{\vec{w}}(\vec{x}) = \vec{x}^T\vec{w} = \sum_{j=0}^{d} w_j x_j$ linear regression problem

③ OLS: $\hat{y}_i = h_{\vec{w}}(\vec{x}_i) = \vec{x}_i^T\vec{w}$, we want $\hat{y}_i \approx y_i$

$$\begin{pmatrix}\hat{y}_1\\ \hat{y}_2\\ \vdots\\ \hat{y}_n\end{pmatrix} = \begin{pmatrix}1 & x_{11} & \cdots & x_{1d}\\ 1 & x_{21} & \cdots & x_{2d}\\ \vdots & \vdots & & \\ 1 & x_{n1} & \cdots & x_{nd}\end{pmatrix}\begin{pmatrix}w_0\\ w_1\\ \vdots\\ w_d\end{pmatrix} = X\vec{w}$$

(1st data $(1, x_{11}, \cdots, x_{1d})$; d: num of feature)

loss (cost) function (sum of squared error / SSE) (target value in training data)

$$L(\vec{w}) = \sum_{i=1}^{N}(\hat{y}_i - y_i)^2 = \sum_{i=1}^{N}(h_{\vec{w}}(\vec{x}_i) - y_i) = \sum_{i=1}^{N}(\vec{x}_i^T\vec{w} - y_i)^2 = \|X\vec{w} - \vec{y}\|_2^2$$

optimization $\vec{w}^* = \arg\min_{\vec{w}} \{L(\vec{w}) = \|X\vec{w} - \vec{y}\|_2^2\}$

$L(\vec{w}) = \|X\vec{w} - \vec{y}\|_2^2 = (X\vec{w} - \vec{y})^T(X\vec{w} - \vec{y}) = \vec{w}^TX^TX\vec{w} - 2\vec{w}^TX^T\vec{y} + \vec{y}^T\vec{y}$

$\nabla_{\vec{w}} L(\vec{w}) = 2X^TX\vec{w} - 2X^T\vec{y} = 0 \Rightarrow X^TX\vec{w} = X^T\vec{y} \Rightarrow \vec{w}^* = (X^TX)^TX^T\vec{y}$

L is convex ← $HL(\vec{w}) = \nabla^2 L(\vec{w}) = 2X^TX$, $\vec{w}^T(2X^TX)\vec{w} = 2(X\vec{w})^T(X\vec{w}) = 2\|X\vec{w}\|_2^2 \geq 0$

④ Nonlinear Feature Mapping $y=f(\vec{x})$ nonlinear

$\phi: R^L \to R^{d+1}$ $(d>L)$, $\vec{x}_i \in R^L$, $\phi(\vec{x}_i) \in R^{d+1}$, $\phi(\vec{x}_i)^T = (\phi_0(\vec{x}_i) \cdots \phi_d(\vec{x}_d))$ (basis function (polynomial, gaussian))

hypo function $h_{\vec{w}}(\vec{x}) = \sum_{j=0}^{d} w_j \phi_j(\vec{x}) = \phi(\vec{x})^T\vec{w}$

⑤ Polynomial Feature Mapping for 1 dimensional input

$x \to \phi(x) = (1\ x\ x^2 \cdots x^m)^T$, $h_{\vec{w}}(x) = w_0 + w_1 x + \cdots + w_m x^m = \phi(x)^T\vec{w}$ (input 可為 x)

Dataset $D=\{(x_i, y_i)\}_{i=1}^{N}$, $y=f(x)$, $\hat{y} = h_{\vec{w}}(x) = w_0 + w_1 x + \cdots + w_m x^m = \phi(x)^T\vec{w}$

$\vec{w} = (w_0, w_1, \cdots, w_m)^T$, $\phi(x)^T = (1\ x\ x^2, \cdots, x^m)$

$$\begin{pmatrix}\hat{y}_1\\ \vdots\\ \hat{y}_N\end{pmatrix} = \begin{pmatrix}1 & x_1 & x_1^2 & \cdots & x_1^m\\ 1 & x_2 & x_2^2 & \cdots & x_2^m\\ \vdots & \vdots & \vdots & & \vdots\\ 1 & x_N & x_N^2 & & x_N^m\end{pmatrix}\begin{pmatrix}w_0\\ w_1\\ \vdots\\ w_m\end{pmatrix} \quad \hat{\vec{y}} = X\vec{w}$$

(input x_i)

$L(\vec{w}) = \|X\vec{w} - \vec{y}\|_2^2 \longrightarrow w^* = (X^TX)^{-1}X^Ty$

⑥ Regularization – Ridge Regression (L_2) (Lasso Regression (L_1)) (validation)

$L(\vec{w}) = \|X\vec{w} - \vec{y}\|_2^2 + \lambda\|\vec{w}\|_2^2$, $\lambda>0$, hyperparameter, choose through

$= \vec{w}^TX^TX\vec{w} - 2\vec{w}^TX^T\vec{y} + \vec{y}^T\vec{y} + \lambda\vec{w}^T\vec{w}$

$\nabla_{\vec{w}} L(\vec{w}) = 2X^TX\vec{w} - 2X^T\vec{y} + 2\lambda\vec{w} = 0 \Rightarrow \vec{w}_{ridge} = (X^TX + \lambda I)^{-1}X^Ty$

$HL(\vec{w}) = \nabla_{\vec{w}}^2 L(\vec{w}) = 2X^TX + 2\lambda I$ positive definite, minimum unique

Regularization can be applied to overcome overfitting (overfitting)

⑦ Maximum Likelihood Estimate (MLE) Model

underlying model: $y=f(\vec{x})$, dataset $D=\{(\vec{x}_i, y_i)\}_{i=1}^{N}$

$y_i \in R$ is the observation of r.v. Y_i, i.e. $Y_i = h_{\vec{w}}(\vec{x}_i) + \varepsilon_i$ (r.v.)

assume $\varepsilon_i \sim N(0, \sigma^2)$, i.i.d. then $Y_i \sim N(h_{\vec{w}}(\vec{x}_i), \sigma^2)$

$L(\vec{w}, D) = P(Y_1=y_1, \cdots, Y_N=y_n \mid \vec{x}_1, \cdots, \vec{x}_n, \vec{w}) = \prod_{i=1}^{N} P(y_i \mid \vec{x}_i, \vec{w})$

$l(\vec{w}; X, y) = \ln(L(\vec{w}, D)) = \sum_{i=1}^{N} \ln(P(y_i \mid \vec{x}_i, \vec{w}))$

$P(Y=y_i \mid \vec{x}_i, \vec{w}) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{(y_i - h_{\vec{w}}(\vec{x}_i))^2}{\sigma^2}}$

$l(\vec{w}; X, \vec{y}) = -\sum_{i=1}^{N}\frac{(y_i - h_{\vec{w}}(\vec{x}_i))^2}{2\sigma^2} - N\ln(\sqrt{2\pi}\sigma)$

$\hat{\vec{w}}_{MLE} = \arg\max_{\vec{w}}\left(\sum_{i=1}^{N}\frac{(y_i - h_{\vec{w}}(\vec{x}_i))^2}{2\sigma^2} + N\ln(\sqrt{2\pi}\sigma)\right) = \arg\min_{\vec{w}}\left(\sum_{i=1}^{N}(y_i - h_{\vec{w}}(\vec{x}_i))^2\right)$

$(h_{\vec{w}}(\vec{x}) = \vec{x}_i^T\vec{w})$ $= \arg\min_{\vec{w}}\{\|X\vec{w} - \vec{y}\|_2^2\}$ (OLS problem)

⑧ Maximum a Posterior (MAP) method Module 1 P.2-63

MAP ← Ridge; MLE ← OLS ← Nonlinear, Polynomial

Module 3 Performance Evaluation and Model Selection (underfitting, overfitting / training, validation, test, cross-validation / performance metrics of regression / Bias – Variance trade-off)

① underfitting: the model used is too simple to catch the trend behind the given dataset; overfitting: the model is so powerful it not only catch the trend but also the noise in the dataset.

② Example: $\vec{w}_{ridge} = \arg\min_{\vec{w}}\{\|\phi\vec{w} - \vec{y}\|_2^2 + \lambda\|\vec{w}\|_2^2\} = (\phi^T\phi + \lambda I)^{-1}\phi^T\vec{y}$

$\vec{w}$ – model parameter, a parameter to be optimized

order of polynomial model m, regularization weight λ, hyperparameters

③ true error $R(\vec{w}) = E_{\vec{x},y}[L(h_{\vec{w}}(\vec{x}), y)]$, expected loss over the whole data distribution, $\vec{w}^* = \arg\min_{\vec{w}} R(\vec{w})$ impossible

training error (in-sample error) $\hat{R}_{Train}(\vec{w}) = \frac{1}{N}\sum_{i=1}^{N} L(h_{\vec{w}}(\vec{x}_i), y_i)$

test error (out-of-sample error) $\hat{R}_{test}(\vec{w}) = \frac{1}{M}\sum_{i=1}^{M} L(h_{\vec{w}}(\vec{x}_i), y_i)$

④ The purpose of validation set is to determine the best value of the hyperparameters (used for hyperparameter tuning) Then, the tuned model will be tested using the test dataset.

⑤ Cross-validation: K-fold cross-validation: P.14

1. Shuffle the training data, and partition it into K blocks
2. Train the model on all the data except block i, use block i to evaluate the model. 3. Average the K validation error

⑥ Training Data – optimize the model parameter

Validation Data – optimize the model's hyperparameter

Test Data – evaluate the optimized model

⑦ Goodness-of-fits (performance metrics of regression model)

Mean Squared Error (MSE) $= \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

Mean Absolute Error (MAE) $= \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$

R^2 score $= 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$ most of time $[0,1]$ → 1 better

⑧ Bias-Variance Decomposition P27, 31, (32) (38) 39

expected squared error $\epsilon(\vec{x}, h) = E[(h(\vec{x}, D) - Y)^2]$ (training data set)

fixed test point $\vec{x}$, two r.v. D, Y, where $Y = f(\vec{x}) + Z$

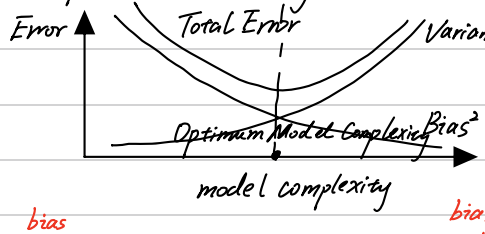
$\epsilon(\vec{x}, h) = E[(h(\vec{x}, D) - Y)^2] = (E[h(\vec{x}, D)] - f(\vec{x}))^2 + Var(h(\vec{x}, D)) + Var(Z)$

Bias Variance Noise

Bias-Variance tradeoff (underfitted – Good fit/Robust – overfitted)

Overfitting — model captures the noise of the data, fits the data too well, shows low bias but high variance (fit too well; change dataset, poor generalization to unseen data)

Underfitting — model can not capture the underlying trend of the data, does not fit the data well enough, shows low variance but high bias (模型过于简单, 数据被欠 ×fit well)



Training error reflect bias (bias) not variance. Test error reflects both. (bias + variance)

- Decreasing the bias ↓ will increase the variance ↑
- Decreasing the variance ↓ will increase the bias ↑

Use validation or cross validation to tune some hyperparameters of your algorithm to achieve the trade-off

Regularization to overcome overfitting, bias ↑ variance ↓

Example Module 3 P42-48

⑨ Reduce avoidable bias ← increase model size (complexity) / Reduce/eliminate regularization

Reduce variance — Decrease the model size (complexity) / add regularization / add more training data / feature selection to reduce number of input features / add early stop during training

---

Module 4 Perceptron and Logistic Regression (classification / performance metrics for classifiers / Perceptron model / Gradient Descent / Perceptron training algorithm (Batch training / sequential training) / Logistic Regression)

① Classification Module 4 P1-6

Decision regions, Decision boundaries

Discriminant function 判别函数

② Performance Metrics for Classifier

Confusion Matrix ← multi-class classifier case / binary classifier case

P15

Classifier		Actual class: Positive	Actual class: Negative
Predicted class	Positive	TP	FP
	Negative	FN	TN

TP – True Positive

TN – True Negative

FP – False Positive

FN – False Negative

$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$ (imbalance dataset, mislead)

Error rate $= 1 - Accuracy$ P15-24

$Precision = \frac{TP}{TP + FP}$ $Recall = \frac{TP}{TP + FN}$

$F_1 = \frac{2}{recall^{-1} + precision^{-1}} = \frac{2 \cdot recall \cdot precision}{precision + recall}$ F_β score

③ linearly separable, line/hyperplane

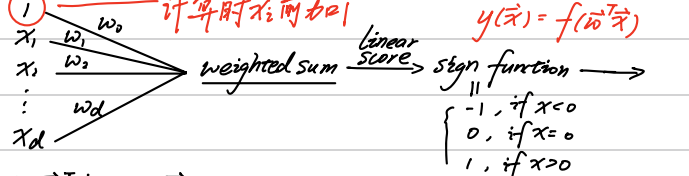
④ Perceptron Model

Binary linearly separable case: input $\vec{x}$, two classes C_1, C_2.

$y = 1$ for C_1, $y = -1$ for C_2. discriminant function $g(\vec{x}) = \vec{w}^T\vec{x}$

$\vec{w}^T\vec{x}_i > 0$, $\vec{x}_i$ in C_1 ; $\vec{w}^T\vec{x}_i \le 0$, $\vec{x}_i$ in C_2

$\vec{w}^T\vec{x}$ is called linear score of the input $\vec{x}$



$\Rightarrow \vec{w}^T\vec{x}_i > 0$, $\vec{x}_i$ in C_1 $(y_i = 1)$

$\vec{w}^T\vec{x}_i \le 0$, $\vec{x}_i$ in C_2 $(y_i = -1)$

normalized linear score

$\Rightarrow \vec{w}^T\vec{x}_i y \ge 0$. any $\vec{x}_i \in D$

Perceptron criterion: the error (loss) function

$$E_p(\vec{w}) = -\sum_{i \in M} \vec{w}^T\vec{x}_i y_i$$

M denote the set of all misclassified instance

try to minimize the quantity $-\vec{w}^T\vec{x}_i y_i$

Gradient Descent Procedure

$$\vec{w}^{k+1} = \vec{w}^k - \eta \nabla E_p(\vec{w}^k)$$

k: indexes the steps; η: step size (learning rate); $\nabla E_p(\vec{w})$: gradient vector

$$\nabla E_p(\vec{w}) = -\sum_{i \in M} \vec{x}_i y_i \Rightarrow \vec{w}^{k+1} = \vec{w}^k + \eta \sum_{i \in M} \vec{x}_i y_i$$

The Perceptron training Algorithm

Batch size (批量大小) — 每次梯度更新时使用样本的数量

epoch (训练周期) — one cycle through the entire training dataset

- Batch Gradient Descent 全批量梯度下降
  批量大小等于整个训练集大小
- Mini-batch Gradient Descent 小批量梯度下降
  批量大小介于 1 和训练集大小之间 (如 32, 64)
- Stochastic Gradient descent, SGD 随机梯度下降
  批量大小为 1, 速度快, 方向不准, 收敛路径曲折 shuffling the training set each epoch

① Batch Training Algorithm 批训练 P41

Batch size = N, 每次迭代中,批训练使用整个训练集的数据来计算误差并更新权重 所有数据遍历后 misclassified 的存在 M_k 中, $\vec{w} \leftarrow \vec{w} + \eta \sum_{i \in M_k} \vec{x}_i y_i$

② Sequential Training Algorithm 顺序训练 P42

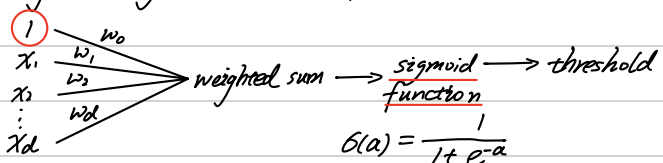
Batch size = 1, 每次迭代中使用单个样本来更新权重

$k \leftarrow 0$, $k \leftarrow k \bmod N$ (k/N的余数), if $\vec{x}_k$ misclassified, $\vec{w} \leftarrow \vec{w} + \eta \vec{x}_k y_k$

⑤ logic OR, XOR problem P43, 44

⑥ Sequential training Example P45-58

⑦ Logistic Regression (Classification) P70-92



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

$$E(\vec{w}) = -\ln p(\vec{y}|\vec{w}) = -\sum_{i=1}^{N}\{y_i \ln p_i + (1-y_i)\ln(1-p_i)\}$$

$$p_i = P(C_1|\vec{x}_i) = \sigma(\vec{w}^T\vec{x}_i) = \frac{1}{1+e^{-\vec{w}^T\vec{x}_i}}$$

$$\nabla_{\vec{w}} E(\vec{w}) = -\sum_{i=1}^{N}\left(\frac{y_i}{p_i} - \frac{1-y_i}{1-p_i}\right)p_i(1-p_i)\vec{x}_i = -\sum_{i=1}^{N}(y_i - p_i)\vec{x}_i$$

$$\vec{w}_{k+1} = \vec{w}_k - \eta \nabla_{\vec{w}} E(\vec{w}) = \vec{w}_k + \eta \sum_{i=1}^{N}(y_i - p_i)\vec{x}_i$$

where $p_i = \sigma(\vec{w}_k^T\vec{x}_i) = \frac{1}{1+e^{-\vec{w}_k^T\vec{x}_i}}$

- Regularized Logistic Regression $E(\vec{w}) = -\ln p(\vec{y}|\vec{w}) + \lambda\|\vec{w}\|_2^2$ P84
- Multiclass Logistic Regression

Module 5 SVM (margin/SVM/The lagrange mltiplier method for constrained optimization/The lagrange dual problem)

① Margin of hyperplane — measure the distance between the hyperplane and the closest data points from each class.

hyperplane $H: \{\vec{x}: \vec{w}^T\vec{x} + b = 0\}$, find the distance from a point $\vec{x}$ to hyperplane H. $\vec{x} = \vec{x}_p + r\frac{\vec{w}}{\|\vec{w}\|}$

r is the algebraic distance from $\vec{x}$ to hyperplane H

$$\vec{w}^T\vec{x} + b = \vec{w}^T\vec{x}_p + b + r\frac{\vec{w}^T\vec{w}}{\|\vec{w}\|} \Rightarrow r = \frac{\vec{w}^T\vec{x} + b}{\|\vec{w}\|}$$

hyperplane $H: \{\vec{x}: \vec{w}^T\vec{x} + b = 0\}$

algebraic distance of a point $\vec{x}_i$ to H is $r_i = \frac{\vec{w}^T\vec{x}_i + b}{\|\vec{w}\|}$

geometric distance of a point $\vec{x}_i$ to H is $\frac{y_i(\vec{w}^T\vec{x}_i + b)}{\|\vec{w}\|}$

The margin of hyperplane H is the geometric distance of the closet point in the data set to the hyperplane

$$\min_i \left\{\frac{y_i(\vec{w}^T\vec{x}_i + b)}{\|\vec{w}\|}\right\}$$

Those point $\vec{x}_i$ that satisfies $\vec{w}^T\phi(\vec{x}_i) + b = \pm 1$ are called support vectors

The maximum margin solution is found by solving the optimization problem

$$\arg\min_{\vec{w},b}\left\{\frac{1}{2}\|\vec{w}\|^2\right\}$$

subject to $y_i(\vec{w}^T\phi(\vec{x}_i) + b) \geq 1$, $i = 1, 2, \cdots, N$

② Lagrange Multiplier Method

$$\min_{x \in R^n} f(\vec{x}) \quad \text{subject to} \quad \begin{cases} C_i(\vec{x}) = 0, & i \in \mathcal{E} \\ C_j(\vec{x}) \geq 0, & j \in \mathcal{J} \end{cases}$$

$f(\vec{x})$: objective function; $C_i(\vec{x}) = 0$: equality constraints; $C_j(\vec{x}) \geq 0$: inequality constraint

Lagrangian function $\mathcal{L}(\vec{x}, \lambda_1) = f(\vec{x}) - \lambda_1 C_1(\vec{x})$

$$\nabla_{\vec{x}}\mathcal{L}(\vec{x}, \lambda_1) = \nabla f(\vec{x}) - \lambda_1 \nabla C_1(\vec{x})$$

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(x) \\ \frac{\partial f}{\partial x_n}(x) \end{pmatrix} \qquad \nabla_{\vec{x}}\mathcal{L}(\vec{x}, \lambda_1) = 0$$

(like $\nabla_{x_1}\mathcal{L}(\vec{x}, \lambda_1) = 0$, $\nabla_{x_2}\mathcal{L}(\vec{x}, \lambda_1) = 0$)

$\Rightarrow x_1, x_2$ use constraint $\Rightarrow \lambda \Rightarrow x$

Example P21 P27-29 P30-33

$$\min\{x_1 + x_2\}, \text{ s.t. } 2 - x_1^2 - x_2^2 \geq 0, \; x_2 \geq 0$$

$$\mathcal{L}(\vec{x}, \vec{\lambda}) = f(\vec{x}) - \lambda_1 C_1(\vec{x}) - \lambda_2 C_2(\vec{x})$$

$$= x_1 + x_2 - \lambda_1(2 - x_1^2 - x_2^2) - \lambda_2 x_2$$

$$\frac{\partial \mathcal{L}(\vec{x}, \vec{\lambda})}{\partial x_1} = 2\lambda_1 x_1 + 1 = 0 \Rightarrow x_1 = -\frac{1}{2\lambda_1}$$

$$\frac{\partial \mathcal{L}(\vec{x}, \vec{\lambda})}{\partial x_2} = 2\lambda_1 x_2 + 1 - \lambda_2 = 0 \Rightarrow x_2 = \frac{\lambda_2 - 1}{2\lambda_1}$$

λ_1 is strictly positive only when $C_1(\vec{x})$ is active, that is

$$C_1(\vec{x}) = 2 - x_1^2 - x_2^2 = 0$$

λ_2 is strictly positive only when $C_2(\vec{x})$ is active, that is

$C_2(\vec{x}) = x_2 = 0$ 代回 constraints

$$\Rightarrow 2 - x_1^2 = 0 \Rightarrow x_1^2 = 2 \Rightarrow \frac{1}{4\lambda_1^2} = 2 \Rightarrow \lambda_1 = \frac{1}{2\sqrt{2}}$$

$$\frac{\lambda_2 - 1}{2\lambda_1} = 0 \Rightarrow \lambda_2 = 1$$

$$\Rightarrow \vec{\lambda} = \left(\frac{1}{2\sqrt{2}}, 1\right)^T$$

$$\vec{x} = (x_1, x_2)^T = \left(-\frac{1}{2\lambda_1}, \frac{\lambda_2 - 1}{2\lambda_1}\right) = (-\sqrt{2}, 0)$$

③ Lagrange Dual Problem

$$\min_{\vec{x} \in R^n} f(\vec{x}) \text{ subject to } C_i(\vec{x}) \geq 0, \; i = 1, \cdots, m$$

$$\mathcal{L}(\vec{x}, \vec{\lambda}) = f(\vec{x}) - \sum_{i=1}^{m}\lambda_i C_i(\vec{x})$$

$$\nabla f(\vec{x}) = \lambda_1 \nabla C_1(\vec{x}) + \lambda_2 \nabla C_2(\vec{x}) + \cdots + \lambda_m \nabla C_m(\vec{x})$$

$$C_i(\vec{x}) \geq 0, \; i = 1, \cdots, m$$

$$\lambda_i C_i(\vec{x}) = 0 \quad i = 1, \cdots, m$$

Dual objective function $g(\vec{\lambda}) = \inf_{\vec{x}} \mathcal{L}(\vec{x}, \vec{\lambda})$

Dual problem $\max_{\vec{\lambda}} q(\vec{\lambda})$ subject to $\vec{\lambda} \geq 0$

Example P37-39

$\min_{x_1,x_2}\left(\frac{1}{2}(x_1^2+x_2^2)\right)$ s.t. $x_1 - 1 \geq 0$

$L(x_1,x_2,\lambda_1) = \frac{1}{2}(x_1^2+x_2^2) - \lambda_1(x_1-1)$

$\frac{\partial L}{\partial x_1} = x_1 - \lambda_1 = 0 \Rightarrow x_1 = \lambda_1$

$\frac{\partial L}{\partial x_2} = x_2 = 0 \Rightarrow x_2 = 0$

Dual objective function

$q(\lambda_1) = \frac{1}{2}(\lambda_1^2+0) - \lambda_1(\lambda_1-1) = -\frac{1}{2}\lambda_1^2 + \lambda_1$

Dual problem

$\max_{\lambda_1 \geq 0} \left(-\frac{1}{2}\lambda_1^2+\lambda_1\right)$

$\frac{d\left(-\frac{1}{2}\lambda_1^2+\lambda_1\right)}{d\lambda_1} = 0 \Rightarrow \lambda_1 = 1$

$\Rightarrow \vec{x}^* = (x_1\ x_2)^T = (\lambda_1\ 0)^T = (1\ 0)$

④ SVM

The maximum margin solution is found by solving

$\arg\min_{\vec{w},b}\left\{\frac{1}{2}\|\vec{w}\|^2\right\}$ subject to $y_i(\vec{w}^T\phi(\vec{x}_i)+b) \geq 1,\ i=1,\cdots,N$

hard margin P48-51

soft margin P52-59

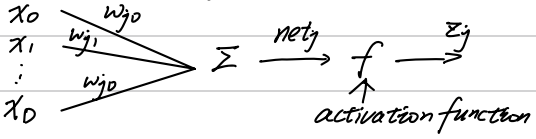
Kernels and Kernel Trick P60-69

Module 6 ANN (Multi-layer Perceptron (MLP) / forward propagation backpropagation / regularization)

① Multiple-layer Perceptron Example P9-13

② ANN

1) The hidden layer

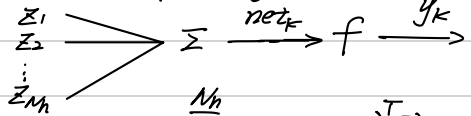


$net_j = \sum_{i=0}^{D} x_i w_{ji} = \vec{w}_j^T \vec{x}$

$\vec{w}_j = (w_{j0} \cdots w_{jD})^T$, $\vec{x} = (x_0, \cdots, x_D)^T$

$z_j = f(net_j)$

2) The output layer



$net_k = \sum_{j=1}^{N_h} z_j w_{kj} = \vec{w}_k^T \vec{z}$

$y_k = f(net_k)$

activation function P19

sigmoid $f(net) = \frac{1}{1+e^{-net}}$

tanh $f(net) = \frac{e^{net}-e^{-net}}{e^{net}+e^{-net}}$

③ Training problem

$E(\vec{w}) = \sum_{n=1}^{N} E_n(\vec{w}) = \frac{1}{2}\sum_{n=1}^{N} \|y(\vec{x}_n,\vec{w}) - \vec{t}_n\|^2$

Gradient descent + backpropagation

$\vec{w}^{\tau+1} = \vec{w}^{\tau} - \eta \nabla E(\vec{w}^{\tau})$

④ Backpropagation

At hidden layer, node j:

$net_j = \sum_{i=0}^{D} x_i w_{ji} = \vec{w}_j^T \vec{x}$, $z_j = f(net_j)$

$\begin{pmatrix} net_1 \\ \vdots \\ net_{N_h} \end{pmatrix} = \begin{pmatrix} w_{10} & w_{11} & \cdots & w_{1D} \\ \vdots & \vdots & & \vdots \\ w_{N_h 0} & w_{N_h 1} & \cdots & w_{N_h D} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_D \end{pmatrix}$

At output layer, node K

$net_k = \sum_{j=1}^{N_h} z_j w_{kj} = \vec{w}_k^T \vec{z}$, $y_k = f(net_k)$

$\begin{pmatrix} net_1 \\ \vdots \\ net_C \end{pmatrix} = \begin{pmatrix} w_{11} & \cdots & w_{1N_h} \\ \vdots & & \vdots \\ w_{C1} & \cdots & w_{CN_h} \end{pmatrix} \begin{pmatrix} z_1 \\ \vdots \\ z_{N_h} \end{pmatrix}$

instance $(\vec{x}_n, \vec{t}_n)$ from training data set, output $\vec{y}_n$

$\vec{t}_n = \begin{pmatrix} t_1 \\ \vdots \\ t_C \end{pmatrix}$ target value; $\vec{y}_n = \begin{pmatrix} y_1 \\ \vdots \\ y_C \end{pmatrix}$ predicted value

Define the training error

$E_n = \frac{1}{2}\sum_{k=1}^{C} (y_k - t_k)^2 = \frac{1}{2}\|\vec{y}_n - \vec{t}_n\|^2$

First, hidden-to-output weights w_{kj}

$\frac{\partial E_n}{\partial w_{kj}} = \frac{\partial E_n}{\partial y_k} \cdot \frac{\partial y_k}{\partial net_k} \cdot \frac{\partial net_k}{\partial w_{kj}} = (y_k - t_k) f'(net_k) z_j$

($\frac{\partial E_n}{\partial y_k} = y_k - t_k$, $\frac{\partial y_k}{\partial net_k} = f'(net_k)$, $\frac{\partial net_k}{\partial w_{kj}} = z_j$)

sensitivity of node k, $\delta_k = -\frac{\partial E_n}{\partial net_k} = (t_k - y_k) f'(net_k)$

$\frac{\partial E_n}{\partial w_{kj}} = -\delta_k z_j$

Second, input-to-hidden weights w_{ji}

$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial net_k} \cdot \frac{\partial net_k}{\partial z_j} \cdot \frac{\partial z_j}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ji}} = -\delta_j x_i$

sensitivity for hidden node j $\delta_j = \left(\sum_{k=1}^{C} w_{kj}\delta_k\right) f'(net)$

⑤ Improving Backpropagation training P36-42

1) Initializing weights 2) Activation function 3) Scaling input

4) Num of hidden nodes 5) Learning rate 6) Overtraining problem

⑥ Regularization in Neural Network

$\tilde{E}(\vec{w}) = E(\vec{w}) + \alpha\|\vec{w}\|_2^2$

Module 7 KNN and Naive Bayes Classifier (The k-Nearest Neighbor Method / Metrics (distance) / Curse of dimensionality / Naive Bayes Classifier)

① K-Nearest Neighbor (KNN) Module 7 P10 / Question P15

choose K — calculate distance — sort D — first K — $y = \overset{\text{众数}}{\text{mode}}()$

② Metrics used in KNN: $\vec{x}=(x_1,\cdots,x_m)^T$, $\vec{y}=(y_1,\cdots,y_m)^T$

Euclidean distance: $d(\vec{x},\vec{y}) = \left(\sum_{i=1}^{m}(x_i-y_i)^2\right)^{\frac{1}{2}}$ L_2 norm

Minkowsky distance: $d(\vec{x},\vec{y}) = \left(\sum_{i=1}^{m}|x_i-y_i|^p\right)^{\frac{1}{p}}$ L_p norm

Manhanton distance: $d(\vec{x},\vec{y}) = \sum_{i=1}^{m}|x_i-y_i|$ L_1 norm

Chebyshev distance: $d(\vec{x},\vec{y}) = \max_i |x_i-y_i|$ L_∞ norm

Hamming distance: $d_i = \begin{cases}1, & \text{if } x_i \neq y_i \\ 0, & \text{otherwise}\end{cases}$; $d(\vec{x},\vec{y}) = \sum_{i=1}^{m} d_i$

cosine similarity measure $sim(\vec{x},\vec{y})$; Correlation $corr(\vec{x},\vec{y})$ P14

③ Curse of dimensionality P17

N data points, d dimensional unit hypercube, a hypercube of edge l ($0<l<1$) contains K nearst neighbor of test set, fraction $r=\frac{K}{N}$

$l^d = r \Rightarrow l = r^{\frac{1}{d}}$, $d\to\infty, l\to 1$ close to entire unit hypercube

数据稀疏性 Data Sparsity

This breaks down the KNN assumptions, because in high dimensions, the points in K nearest neighbor are not particularly closer than any other data points in the training set.

Dimension reduction (PCA) may provide a solution to alleviate the problem.

④ Naive Bayes Classifier — generative approach P25-29

training dataset $D=\{(\vec{x}_i,y_i)\}_{i=1}^{N}$ belongs to class C_k if $y_i=k$

new instance $\vec{x}=(x_1\ x_2\ \cdots\ x_d)^T$, predict the class of this instance

Bayesian approach to assign the most probable target value

$$y_{MAP} = \arg\max_k P(C_k|x_1,\cdots,x_d) \quad \text{(attribute values)}$$
$$= \arg\max_k \frac{P(x_1,\cdots,x_d|C_k)P(C_k)}{P(x_1,\cdots,x_d)}$$
$$= \arg\max_k P(x_1,\cdots,x_d|C_k)P(C_k) \quad \propto$$

Assume the attributes are conditionally independent → Naive

$$P(x_1,x_2,\cdots,x_d|C_k) = P(x_1|C_k)\cdots P(x_d|C_k) = \prod_{j=1}^{d} P(x_j|C_k)$$

$$\Rightarrow y_{MAP} = \arg\max_k P(C_k)\prod_{j=1}^{d} P(x_j|C_k)$$

计算不同 K 的 $P(C_k)\prod_{j=1}^{d}P(x_j|C_k)$ 选择最大值的类 (计算 $P(C_k)$ 和 $\prod_{j=1}^{d}P(x_j|C_k)$ 数↓)

⑤ Laplace smoothing (additive smoothing) ($P(\text{yellow}|+)=0$) P30-33

to calculate conditional probability $P(x_i|C_k) = \frac{N_{x_i,C_k}+1}{N_{C_k}+d}$

N_{x_i,C_k} — 特征 x_i 在类别 C_k 训练集中出现的次数, d - dimensionality of the feature vector

N_{C_k} - 训练集中属于类别 C_k 的样本数量

Module 8 Decision Tree and Random Forest Model (Entropy / Information Gain / Overfitting / Ensemble learning)

① Entropy — measure of impurity of a collection of samples

(binary sets) $Entropy(S) = -P_{\oplus}\log_2 P_{\oplus} - P_{\ominus}\log_2 P_{\ominus}$ (0~1)

($0\log_2 0 = 0$) entropy↓ impurity↓, $P_{\oplus}=0.5$ entropy=1

(multi-class sets) $Entropy(S) = -\sum_{i=1}^{k} P_i \log_2 P_i$

② Information gain (expected reduction in entropy)

$$Gain(S,A) = Entropy(S) - \sum_{v\in Value(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

feature 样本总数

③ Overfitting in Decision Tree: Limit the depth and nodes; Post-prune the tree; use multiple trees to form a forest

④ Ensemble learning - Random Forest

Bagging (Bootstrap aggregating)

Random subset of features (attributes)

L_2 norm $\vec{x}=(x_1,\cdots,x_n)$, $\|\vec{x}\|_2 = \sqrt{x_1^2+x_2^2+\cdots+x_n^2}$

$\arg\min_{\vec{w}} L(\vec{w})$ 使 $L(\vec{w})$ 取得最小值时 $\vec{w}$ 的值

$\nabla_{\vec{x}}(\vec{x}^T\vec{b}) = \vec{b}$, $\nabla_{\vec{x}}(\vec{x}^TA\vec{x}) = 2A\vec{x}$ $\nabla_{\vec{x}}(A\vec{x}) = A$

$$\nabla f(x_1,\cdots,x_n) = \begin{pmatrix}\frac{\partial f}{\partial x_1}\\ \vdots \\ \frac{\partial f}{\partial x_n}\end{pmatrix}$$