

Ch1. ML Basics

1. Key concept & definitions in ML.

ML Definition:

make computer learn from data. (automation, extract knowledge, predict future)

ML Strategy:

use "data + model" strategy.

ML Pipeline:

model construction ~ training ~ evaluation.

ML ingredient:

Data ~ Experience.

Model ~ a function with some params need to be optimised.

(objective fun) Loss function ~ Judge how well the params are set.

Training Algorithm ~ optimise model params. use error function to judge model's performance.

2. Typical ML learning types:

1. ML learning tasks:

Supervised, Unsupervised, Reinforcement Learning.

Classification 分类 有监督

Regression 回归 (预测) 有监督

Clustering 聚类 无监督.

2. Supervised learning:

exist a "teacher" provide target output for each data pattern.

training example: < input data pattern, target output >

typical: classification, regression.

3. Unsupervised learning:

no explicit "teacher" providing expected output, learn from untagged data.

learn "hidden structure" from unlabelled data.

typical: clustering, generative modeling (estimate distribution)

unsupervised representation learning.

4. Reinforcement Learning.

exist a "teacher" provide reward & punishment.

5. Classification & Regression.

classification: 把... input data patterns ~ category memberships.

⇒ Binary classification

Multi-class classification

Multi-label classification

Structured classification

regression: 3rd thi). "estimate relationship among inputs and output vars".
input var ~ continuous output vars.

6. Clustering.

arrange objects into groups based on their similarity.

Ch 2. K-NN Classification

1. K-NN rule for classification.

* core idea: Assign datapoint to most common class in its k-nearest neighbor.

* pseudo code:

Testing point x_{te} .

for each training point x_{tr} :

measure distance $[x_{te}, x_{tr}]$.

sort distances.

pick k-nearest points.

assign x_{te} to the most common class.

* usage: binary classification.

2. K-NN rule for regression:

* core idea: predicted value of $x_{pr} = \text{average (sum of its k-nearest neighbors' value)}$

* pseudo code:

Testing point x_{te} .

for each training point x_{tr} :

measure distance $[x_{te}, x_{tr}]$.

sort distances.

pick k-nearest points: $x_{NN_1}, \dots, x_{NN_k}$.

output $y_{te} = \frac{1}{k} \cdot \sum_{i=1}^k y_{NN_i}$.

* usage: value prediction for discrete points, face prediction.

(compare Euclidean distance using image pixels)

3. Distance Calculation

1. Euclidean distance + Minkowski distance:

$$d(p, q) = \left[\sum_{i=1}^d |p_i - q_i|^k \right]^{\frac{1}{k}}$$

$\Rightarrow k=1$: Manhattan distance

$k=2$: Euclidean distance

$k \geq 3$: Minkowski distance.

2. Similarity Measuring:

cosine 矢量上是 similarity.

越大相似度越高。 { i. use inner product.

ii. use cosine (p, q) as the measurement of distance: $d(p, q) = 1 - \text{Cos}(p, q)$

$$\text{Cos}(p, q) = \frac{S_{\text{inner}}(p, q)}{\|p\|_2 \cdot \|q\|_2} = \frac{\vec{p} \cdot \vec{q}}{\|\vec{p}\| \cdot \|\vec{q}\|}$$



4. K-NN behavior

training sample size:

- { small → insufficient
- large → more information, cost time & space
- noisy → inaccurate

neighbor number K: hyper parameter.

- { small: may model noise, inaccurate. (sensitive to noise)

- | large: include too many irrelevant info, may negatively affect prediction. (inaccurate prediction)

Ch3. Machine Learning Experiments.

1. Classification Performance Measures

* accuracy & error

classification { accuracy rate: $\frac{\# \text{ correctly classified ones}}{\text{total}}$

error rate: $\frac{\# \text{ wrong classified ones}}{\text{total}}$

can't be used solely as evaluation standard: unreliable for assessing unbalanced data.

* Confusion Matrix

confusion matrix (error matrix):

TP	FP
FN	TN

* Important: convert from large confusion matrix with 3×3 to 2×2 confusion matrix for specific class.

\hat{y}_1 : samples in predicted class

\hat{y}_0 : sample in actual class.

* if \hat{y}_1 binary model has F_1 , precision, sensitivity, ... etc. 等指标为同类。

* TP, FP, TN, FN

TP ~ true positive FP ~ false positive

FN ~ false negative TN ~ true negative.

Actual class			
Predicted class	Cat	Dog	Rabbit
Actual class			
Cat	5	2	0
Dog	3	3	2
Rabbit	0	1	11

confusion matrix for class "cat"		
Actual class		
Predicted class	Cat	Non-cat
Actual class		
Cat	5 True Positives	2 False Positives
Non-cat	3 False Negatives	17 True Negatives

* Precision & Recall (sensitivity, true positive rate, TPR) assess ability to classify positive samples.

Precision = $\frac{TP}{TP + FP}$ accuracy on positive

recall = $\frac{TP}{TP + FN}$ how many Ps retrieved from all positive samples

* Specificity & FPR assess ability to classify negative samples.

specificity = $\frac{TN}{TN + FP}$ how many \hat{y}_0 s retrieved (互补 True Negative Rate)

FPR (false positive rate): $1 - \text{specificity} = \frac{FP}{TN + FP}$.

* F_1 Score:

$$F_1 \text{ score: } \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

2. Regression Performance Measures

1. RMSE 根均方误差. $\sqrt{\frac{1}{mn} \sum_{i=1}^m \sum_{j=1}^n (y_{ij} - \hat{y}_{ij})^2}$ (multi)

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

y_{ij} : 理想值

2. MAE 均绝对误差. $\frac{1}{mn} \sum_{i=1}^m \sum_{j=1}^n |y_{ij} - \hat{y}_{ij}|$ (multi)

$\frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$ (single)

\hat{y}_{ij} : 测量值

\bar{y} : y 均值

$$3. MAPE \text{ 均绝对值误差率. } \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m \frac{|y_{ij} - \hat{y}_{ij}|}{|y_{ij}|} \text{ (multi) } \quad \frac{1}{n} \sum_{i=1}^n \frac{|y_i - \hat{y}_i|}{|y_i|} \text{ (single)}$$

$$4. R^2: 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}. \text{ FOR SINGLE output ONLY!}$$

3. Concepts:

1. sample error, true error, bias error, variance error

errors * 样本误差: error computed by a performance metric using a set of data samples.

sample error: always can be calculated:

{ infinite samples: 样本误差 converge to 真实误差.

insufficient samples: 样本误差无法作为真实误差的有效估计.

error * 真实误差: classifier \approx 误分类概率.

regression \approx 预测值和真实值差的期望.

* Bias: the amount that prediction differs from target value

provide optimistically biased estimate of hypothesis over future samples

\Rightarrow Hold out: use new set of test examples independent from training samples.

Bias Error: $E[(y - E[f])^2]$: how much averaged prediction is close to true value.

* Variance: accuracy of new set of test samples vary from true accuracy.

\Rightarrow Increase set of test samples: smaller set of test samples may cause high variance.

Variance Error: $E[(f - E[f])^2]$: how much prediction varies among different realizations of model.

2. Under-fitting & Over-fitting.

* Bias - Variance Decomposition:

$$E[(y - f(x))^2] = E[(f(x) - \hat{y})^2] + E[(y - \hat{y})^2], \hat{y} = E[f(x)].$$

bias error variance error

variance error: 预测结果的对训练数据依赖.

bias error: 预测结果整体和真实值的偏差程度.

* Over-fitting: low bias error, high variance error

\Rightarrow over-complex model, sensitive to fluctuations in training set.

* Under-fitting: high bias error, low variance error.

\Rightarrow excessively simple model.

4. Confidence interval & hypothesis comparison (Z-test)

1. 置信区间 confidence interval

Confidence interval tells you 置信区间.

With p probability, the true error lies in the interval of

Confidence level: p	50%	68%	80%	90%	95%	98%	99%
Constant:							

$\text{error}_D \in [\text{error}_S - a, \text{error}_S + a]$. Interval around the sample error.

Constant: z_p	0.67	1.00	1.28	1.64	1.96	2.33	2.58
--------------------	------	------	------	------	------	------	------

given p , compute a :

$$a = z_p \cdot \sqrt{\frac{\text{error}(1-\text{error})}{n}} \quad \text{error}_D \in [\text{error}_S - a, \text{error}_S + a].$$

* confidence interval is an approximate

* work well for over 30 samples ($n > 30$) and sample error not too close to 0 or 1.

2. Hypothesis test.

Z-test: 适用 ≥ 30 samples.

given: classifier A $\sim \text{error}_S(A)$ find: $\text{error}_S(A) - \text{error}_S(B) > 0$.

classifier B $\sim \text{error}_S(B)$ wonder: $P(\text{error}_S(A) > \text{error}_S(B))$.

ii. 两个 classifier 的 errors 可能不来自 sample set!

Perform Z-test:

$$z_p = \frac{d}{\sigma} : \left\{ \begin{array}{l} d = |\text{error}_S(A) - \text{error}_S(B)| \\ \sigma = \sqrt{\frac{\text{error}_S(A)(1-\text{error}_S(A))}{n_1} + \frac{\text{error}_S(B)(1-\text{error}_S(B))}{n_2}} \end{array} \right.$$

find the closest P in table

Confidence level: p	50%	68%	80%	90%	95%	98%	99%
Constant: z_p	0.67	1.00	1.28	1.64	1.96	2.33	2.58

$$\Rightarrow C = 1 - \frac{1-p}{2}, \text{ finally.}$$

5. Data splitting strategies & usage

Solution:

1. Holdout: dataset \Rightarrow test set + training set (not suitable for small datasets, error rate estimate may be misleading)
2. Random Subsampling (randomly select k samples for testing, rest for training, repeat k times)
3. K-fold Cross Validation + leave-one-out. (divide entire set into K portions)
use $K-1$ portion for training, remaining one for testing
3. Bootstrap (有放回取样) (randomly pick M (instances) for training, rest for testing)

ML - Experiment:

1. Model Training

ii: 1. training-testing set switching. (from high bias error)

2. Model Evaluation

ii: 2. 选用合适的 data splitting strategy.

3. Model Selection.

Ch4. ML Models

1. Basic ML in gradients

i) model : a function that provide answer / solution to given task T.

ii) objective function: $O(\theta)$ compute performance P of given task T. 目标函数 = 损失函数.
(loss function)

iii) optimisation : $\min_{\theta} O(\theta)$.
(training process)

2. Probabilistic & Deterministic approaches for classification & regression

* classification { deterministic thresholding: $\hat{y} = \begin{cases} \text{class 1} & f(x) \geq t \\ \text{class 2} & f(x) < t \end{cases}$ t: treat as hyper parameter. commonly: $t=0$.
probabilistic Probability Inference: $\hat{y} = \arg\max_k p(\text{class } k | x)$.

model posterior class probability $p(\text{class } k | x)$:

1. construct posterior function directly.

logistic regression classifier

2. construct likelihood $p(x | \text{class } k)$ and prior $p(\text{class } k)$ functions,
then calculate posterior function:

naive Bayes

$$p(\text{class } k | x) = \frac{p(x | \text{class } k) \cdot p(\text{class } k)}{p(x)}$$

* regression { deterministic find mapping function

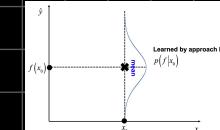
probabilistic model posterior output probability $p(f | x)$.

model posterior output probability $p(f | x)$: f: 输出函数.

\Rightarrow Estimate output using condition mean: $\hat{y} = \mathbb{E}_f[f | x]$.

在每一个 x 上, 输出函数(值)输出一个
概率分布, 其 mean 作为模型在该
 x 上的预测值.

1. model $p(f | x)$ directly.



2. construct likelihood $p(x | f)$ and prior $p(f)$. then
calculate posterior: $p(f | x) = \frac{p(x | f) \cdot p(f)}{p(x)} = \frac{p(x | f) \cdot p(f)}{\int p(x | f) d f}$

3. Principle & Usage of linear model (single output + multiple output):

Linear model - single output: $\hat{y} = f(x_1, \dots, x_d)$.

single data point \vec{x} (单向量): $\hat{y} = \vec{w}^T \cdot \vec{x}$. $\boxed{1} \cdot \boxed{1} = \boxed{0}$

multiple datapoints \vec{X} (-多个): $\vec{Y} = \vec{X} \cdot \vec{w}$. $\boxed{3} \cdot \boxed{1} = \boxed{3}$ Y: -1 1 3 1

Linear model - multi output: $[\hat{y}_1, \dots, \hat{y}_d] = f(x_1, \dots, x_d)$.

single datapoint \vec{x} (单向量): $\hat{y} = \vec{w}^T \cdot \vec{x}$. $\boxed{3} \cdot \boxed{1} = \boxed{3}$

multiple datapoints \vec{X} (-多个): $\hat{Y} = \vec{X} \cdot \vec{w}$ $\boxed{3} \cdot \boxed{3} = \boxed{9}$ Y: -1 1 3 1

1. Linear model for classification:

- { non-probabilistic : separating hyperplane
- probabilistic : logistic regression.

non-probabilistic: decision boundary separation

超平面. 2D~线. 3D~面. 2D~面. 3D~面. 超平面.

Binary Classification: Construct a single-output linear model, then construct the discriminant function by thresholding ($t=0$).

$$f = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_d x_d$$

$$\hat{y} = \begin{cases} +1, & f \geq 0, \\ -1, & f < 0. \end{cases}$$

- +1 indicates the input is from class 1.
- 1 indicates the input is from class 2.

Multiclass Classification: Construct a multi-output linear model, then construct the discriminant function for each class.

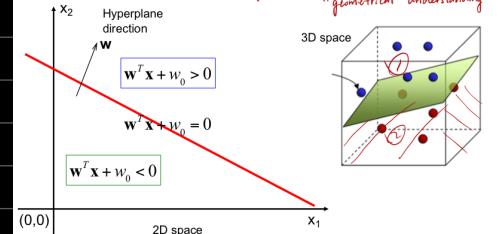
$$f_i = w_{0i} + w_{1i} x_1 + w_{2i} x_2 + \dots + w_{di} x_d, \quad i = 1, 2, \dots, C$$

$$\hat{y}_i = \begin{cases} +1, & f_i \geq 0, \\ -1, & f_i < 0. \end{cases}$$

- +1 indicates the input is from class i.
- 1 indicates the input is not from class i.

Setting a linear function to zero results in a **hyperplane** that partitions the space into two parts. I.e., classes.

"geometrical understanding"

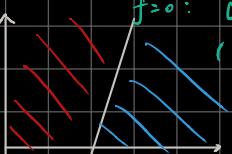


e.g.

$$f = 3 + 5x_1 - x_2$$

$$\hat{y} = \begin{cases} 1, & f \geq 0 \\ -1, & f < 0. \end{cases}$$

$f=0$: classification boundary (decision boundary).



create line to separate datapoints into 2 groups.

probabilistic inference for classification: logistic regression

classification using probabilistic inference

compute class posterior using sigmoid (binary) / softmax (multi-class) from linear model

Logistic Regression

=> Binary: sigmoid function

Binary Classification:

- Use a single-output linear model.

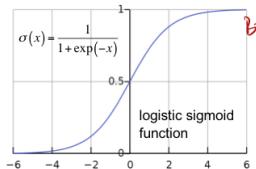
$$f = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_d x_d = \mathbf{w}^T \tilde{\mathbf{x}}$$

bias

One way: MLE
Need to find the optimal \mathbf{w} using the training data.

- Use the **logistic sigmoid function** to convert a real-valued number to probability.

$$\sigma(\mathbf{w}^T \tilde{\mathbf{x}})$$



$$\text{Binary: } p(c_1 | \mathbf{x}) = \sigma(f) = \frac{1}{1 + \exp(-\mathbf{w}^T \tilde{\mathbf{x}})}$$

$$p(c_2 | \mathbf{x}) = 1 - \sigma(f) = \frac{1}{1 + \exp(\mathbf{w}^T \tilde{\mathbf{x}})}$$

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

=> multinomial: softmax function

Multi-class Classification:

- Use a multi-output linear model.

$$f_1 = w_{01} + w_{11} x_1 + w_{21} x_2 + \dots + w_{d1} x_d = \mathbf{w}_1^T \tilde{\mathbf{x}}$$

$$f_2 = w_{02} + w_{12} x_1 + w_{22} x_2 + \dots + w_{d2} x_d = \mathbf{w}_2^T \tilde{\mathbf{x}}$$

⋮

$$f_c = w_{0c} + w_{1c} x_1 + w_{2c} x_2 + \dots + w_{dc} x_d = \mathbf{w}_c^T \tilde{\mathbf{x}}$$

Need to find the optimal \mathbf{w}_i using the training data.

$$p(c_f | \mathbf{x}) = \frac{e^{w_f^T \tilde{\mathbf{x}}}}{e^{w_1^T \tilde{\mathbf{x}}} + e^{w_2^T \tilde{\mathbf{x}}} + \dots + e^{w_c^T \tilde{\mathbf{x}}}}$$

$$= \frac{e^{w_f^T \tilde{\mathbf{x}}}}{\sum_{j=1}^c e^{w_j^T \tilde{\mathbf{x}}}}$$

线性组合转为正数

归一化 [0, 1]

- Use the softmax function to convert outputs to probabilities.

$$P(c_k | \mathbf{x}) = \frac{\exp(\mathbf{w}_k^T \tilde{\mathbf{x}})}{\sum_{j=1}^c \exp(\mathbf{w}_j^T \tilde{\mathbf{x}})}, k = 1, 2, \dots, c$$

$$\text{softmax}(a_1, a_2, \dots, a_k) = \left[\frac{e^{a_1}}{\sum_{i=1}^k e^{a_i}}, \frac{e^{a_2}}{\sum_{i=1}^k e^{a_i}}, \dots, \frac{e^{a_k}}{\sum_{i=1}^k e^{a_i}} \right].$$

$k = 2$ ref: softmax is logistic sigmoid function.

3. Linear model for regression. (only care about single output)

non-probabilistic:

$$f = w_0 + w_1 x_1 + \dots + w_d x_d = \mathbf{w}^T \tilde{\mathbf{x}}. \quad \square$$

\Rightarrow directly use output as predicted value.

probabilistic:

need to estimate $p(f | x)$.

\Rightarrow assume gaussian distribution: $p(f | x) = N(\mathbf{w}^T \tilde{\mathbf{x}}, \sigma^2)$

then $E_f | f(x)$.

\Rightarrow 等价于: $y = \mathbf{w}^T \tilde{\mathbf{x}} + \epsilon$. ϵ is gaussian noise: $\epsilon \sim N(0, \sigma^2)$.

4. Principle & Use of linear basis function model

linear basis function model: 非线性可分的低维空间 $\xrightarrow{\varphi}$ 线性可分的高维空间

directly formulate many non-linear functions (basis function): $\{\varphi_i(x)\}_{i=1}^n$
they work as "feature extractor".

$$\hat{y} = w_0 + w_1 \varphi_1(x) + w_2 \varphi_2(x) + \dots + w_n \varphi_n(x) = \mathbf{w}^T \varphi(x).$$

5. Core idea of kernel method + inner product in kernel space.

kernel method: 用核函数直接计算用 φ 映射而成的高维空间中向量内积。

Kernel Name	Expression $K(\mathbf{x}, \mathbf{y})$	Comments
Linear	$\mathbf{x}^T \mathbf{y}$	No parameter
Polynomial	$(\mathbf{x}^T \mathbf{y} + 1)^p$	p is a user defined parameter
Gaussian, also called radial basis function (RBF)	$\exp(-\ \mathbf{x} - \mathbf{y}\ _2^2 / (2\sigma^2))$	σ is the user defined width parameter
Hyperbolic tangent	$\tanh(c \mathbf{x}^T \mathbf{y} + \beta)$	c & β are user defined parameters

$$\varphi(x_i)^T \cdot \varphi(x_j) = K(x_i, x_j).$$

$$\text{that: } f(x) = w_0 + w_1 x_1 + \dots + w_n x_n.$$

\downarrow 核函数所处的高维空间

$$f(x) = w_0 + w_1 \varphi(x_1) + \dots + w_n \varphi(x_n).$$

(w_0 是 bias)

不知道 $\varphi(x)$, 但知道 weight: w_1, \dots, w_n 可表示新空间 φ 的一组基 $\varphi(x_1), \dots, \varphi(x_n)$ 的线性组合。

\Rightarrow 从 \mathbb{R}^d 到 \mathbb{R}^m 的函数可用核函数表示。

$$\begin{aligned}
 f(\mathbf{x}) &= \mathbf{w}^T \phi(\mathbf{x}) + w_0 \\
 &= \left(\sum_{i=1}^n a_i \phi(\mathbf{x}_i) \right)^T \phi(\mathbf{x}) + w_0 \\
 &= \sum_{i=1}^n a_i \phi^T(\mathbf{x}_i) \phi(\mathbf{x}) + w_0 \\
 &= \sum_{i=1}^n a_i k(\mathbf{x}_i, \mathbf{x}) + w_0
 \end{aligned}$$

- Training becomes to find the best combination coefficients \mathbf{a} .
- $$\mathbf{a} = [a_1, a_2, \dots, a_n]^T$$
- This looks similar to a linear basis function model, where f is a linear combination of many nonlinear basis functions instead.

Ch 5. Loss function.

1. calculation of SSE, MSE, Hinge loss & cross entropy loss.

Recall RMSE root mean square error

$$\text{RMSE} = \sqrt{\frac{1}{mn} \sum_{i=1}^n \sum_{j=1}^m (y_{ij} - \hat{y}_{ij})^2}$$

simplified

SSE: sum-of-squares error loss

$$\text{SSE} = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^c (y_{ij} - \hat{y}_{ij})^2$$

Training Data: $D_{tr} = \{\mathbf{x}_i, y_i\}_{i=1}^N$

Feature vector: $\mathbf{x}_i \in R^d$

Target output: $\mathbf{y}_i \in R^c$ where $\mathbf{y}_i = [y_{i1}, y_{i2}, \dots, y_{ic}]$

MSE: mean-square-error.

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

y_i / y_{ij} : true value

\hat{y}_i / \hat{y}_{ij} : predicted value.

Hinge loss:

$$O(\theta) = \sum_{i=1}^n \max(0, 1 - y_i \cdot f(\theta, \mathbf{x}_i))$$

true label predicted label

↓ ↓

Hinge loss classification (① Hinge loss is objective fun)

Hinge loss assesses classification error.

Given +1/-1 label coding, hinge loss over N training samples is

$$O(\theta) = \sum_{i=1}^N \max(0, 1 - y_i f(\theta, \mathbf{x}_i))$$

class label $y_i \in \{-1, +1\}$

your prediction model: $f(\theta, \mathbf{x}_i)$

Let f be a linear model and add a regularisation term to the loss. This results in support vector machine.

$$\min_{(\mathbf{w}, w_0)} C \sum_{i=1}^N \max(0, 1 - y_i (\mathbf{w}^T \mathbf{x}_i + w_0)) + \frac{1}{2} \mathbf{w}^T \mathbf{w}$$

regularisation parameter (hyper-parameter)

hinge loss

regularisation term

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}$$

Cross Entropy loss:

Cross entropy loss computed over N training samples is

$$O = - \sum_{i=1}^N \left[y_i \log_b(p(c_1 | \mathbf{x}_i)) + (1 - y_i) \log_b(p(c_2 | \mathbf{x}_i)) \right]$$

does it really count?
You can use natural log ($\ln, b=e$) or log base 2 ($b=2$).

Cross entropy measures distance between probability distributions.

Its discrete version can be used to examine the distance between the predicted class probabilities (posterior) and the true probabilities.

$$H(p, q) = -[p(1)\log(q(1)) + p(0)\log(q(0))] \quad (\text{two classes})$$

$$H(p, q) = -[p(1)\log(q(1)) + p(2)\log(q(2)) + \dots + p(c)\log(q(c))] \quad (\text{multiple classes})$$

Cross entropy loss computed over N training samples is

$$O = - \sum_{i=1}^N \sum_{k=1}^c y_{ik} \log_b(p(c_k | \mathbf{x}_i))$$

for each sample,

Cross Entropy Loss

• Multi-class classification $y_{ik}=1$ means that the i-th sample belongs to class k, $y_{ik}=0$ otherwise.

$$\circ 1\text{-of-}K \text{ label coding scheme: } \mathbf{Y} = \begin{bmatrix} y_{11} & y_{12} & \dots & y_{1c} \\ y_{21} & y_{22} & \dots & y_{2c} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N1} & y_{N2} & \dots & y_{Nc} \end{bmatrix}, \text{ where } y_{ik} \in \{0, 1\}$$

• Cross entropy loss computed over N training samples is

$$O = - \sum_{i=1}^N \sum_{k=1}^c y_{ik} \log_b(p(c_k | \mathbf{x}_i))$$

②: cross entropy loss is encoding with one-hot

2. Basic idea of (log) likelihood, MLE (maximum likelihood estimator):

i. like kin: likelihood, log likelihood.

- Likelihood: Given the observed data, it is the conditional probability assumed for the observed data given some parameter values.

假设: 假设选择的模型是当前的数据选择下, 该数据集和训练集中结果一致的可能性最大。

$$\text{Likelihood}(\theta | \text{data}) = p(\text{data} | \theta)$$

- Log likelihood: Take the natural logarithm of the likelihood.

自然对数

Likelihood: $p(x | f)$, $p(x | \text{class})$.

③: if posterior probability using prior, Likelihood + Bayes Thm.

2. MLE (maximum likelihood estimator):

A model can be trained by **maximising** the likelihood (or log likelihood) function of the training samples.

Assume independence between samples.

- **Maximum likelihood estimator (MLE):**

$$\max_{\theta} = \prod_{i=1}^N p(\mathbf{x}_i, y_i | \theta)$$

↑ ideal prediction results
↓ datapoint one param selection

- **Log likelihood maximisation:**

$$\max_{\theta} = \sum_{i=1}^N \log p(\mathbf{x}_i, y_i | \theta)$$

Likelihood of N training samples:

$$L = \prod_{i=1}^N N(y_i | \mathbf{w}^T \tilde{\mathbf{x}}_i, \sigma^2)$$

Assume a Gaussian distribution for likelihood estimation.

$$p(\text{data}_i | \theta) = p(y_i | \theta) = N(y_i | \mathbf{w}^T \tilde{\mathbf{x}}_i, \sigma^2)$$

Log-likelihood:

$$O = \ln(L) = \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta \left[\frac{1}{2} \sum_{i=1}^N (y_i - \mathbf{w}^T \tilde{\mathbf{x}}_i)^2 \right]$$

where $\beta^{-1} = \sigma^2$

(essentially SSE \Rightarrow MLE)

Sum-of-squares error function!

In this case, an MLE is equivalent to a sum-of-squares error minimizer.

3. Concept of Regularisation & Formation of regularisation term:

- A regularisation term can be added to the error function. For instance, in the single-output case, we have

$$\min O_\lambda(\mathbf{w}) = \text{sum of squares error} + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 = \frac{1}{2} \sum_{i=1}^N (y_i - \mathbf{w}^T \tilde{\mathbf{x}}_i)^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w}$$

λ is a positive real-valued number set by the user.

$$\mathbf{w}^T \mathbf{w} = \sum_{j=1}^{d+1} w_j^2$$

- Other type of regulariser:

$$\min O_\lambda(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (y_i - \mathbf{w}^T \tilde{\mathbf{x}}_i)^2 + \frac{\lambda}{2} \sum_{j=1}^{d+1} |w_j|^q$$

Here q is a positive integer set by the user.

- The case of $q=1$ (l_1 -regularisation) for regression is known as **lasso**.
- The case of $q=2$ (l_2 -regularisation) for regression is known as **ridge regression**.

- Regularisation prevents the model from over-fitting to training data.
- When λ is too large, it will lead to under-fitting though.

$$\text{lasso, } l_1\text{-regularisation: } \frac{\lambda}{2} \sum_{j=1}^{d+1} |w_j| = \frac{\lambda}{2} \|\mathbf{w}\|_1$$

$$\text{ridge, } l_2\text{-regularisation: } \frac{\lambda}{2} \sum_{j=1}^{d+1} |w_j|^2 = \frac{\lambda}{2} \mathbf{w}^T \mathbf{w} = \frac{\lambda}{2} \|\mathbf{w}\|_2^2$$

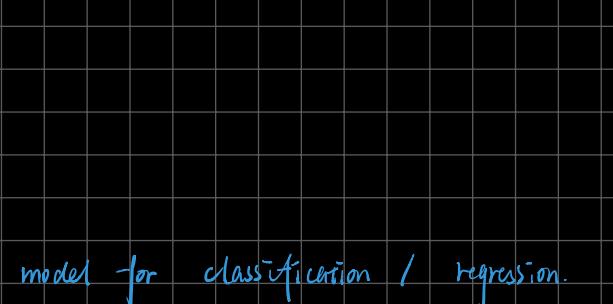
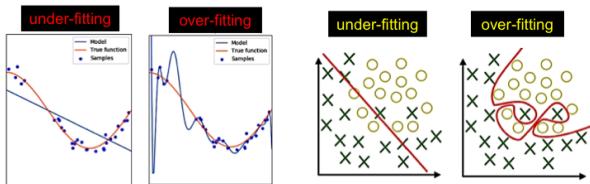
$$\text{general: } l_k\text{-regularisation: } \frac{\lambda}{2} \sum_{j=1}^{d+1} |w_j|^k.$$

作用: 防止模型过拟合。过大则欠拟合。

- Over-fitting: Fit too closely to a particular set of data (e.g., training data), and may therefore fail to fit new data.

- Under-fitting: Cannot capture the underlying trend of the training data.

- Prevent over-fitting, e.g., $O_\lambda(\mathbf{w})$ gives less emphasis to the sum of squares error of the training data.



4. Principle & Usage of (regularised) least square model for classification / regression.

linear least squares (LLS): Train a model by minimising sum-of-squares error.

$$\left\{ \begin{array}{l} \text{single: } \min O(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (y_i - \mathbf{w}^T \tilde{\mathbf{x}}_i)^2. \\ \text{multiple: } \min O(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^{d+1} (y_{ij} - w_j \tilde{x}_{ij})^2. \end{array} \right.$$

+ regularisation term (l_2 -regularisation)

\Rightarrow

- A regularisation term can be added to the error function. For instance, in the single-output case, we have

$$\min O_\lambda(\mathbf{w}) = \text{sum of squares error} + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 = \frac{1}{2} \sum_{i=1}^N (y_i - \mathbf{w}^T \tilde{\mathbf{x}}_i)^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w}$$

$$\mathbf{w}^T \mathbf{w} = \sum_{j=1}^{d+1} w_j^2$$

- Other type of regulariser:

$$\min O_\lambda(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (y_i - \mathbf{w}^T \tilde{\mathbf{x}}_i)^2 + \frac{\lambda}{2} \sum_{j=1}^{d+1} |w_j|^q$$

Here q is a positive integer set by the user.

- The case of $q=1$ (l_1 -regularisation) for regression is known as **lasso**.

- The case of $q=2$ (l_2 -regularisation) for regression is known as **ridge regression**.

Likelihood of An Individual Sample

Consider the i -th training sample (\mathbf{x}_i, y_i)

- **Binary classification:**

$$p(\mathbf{x}_i, y_i | \theta) = \theta(\mathbf{x}_i)^{y_i} (1 - \theta(\mathbf{x}_i))^{1-y_i}$$

- **Multi-class classification:** Assume the class label follows categorical (multinomial) distribution (generalise Bernoulli distribution to more than two options):

$$p(\mathbf{x}_i, y_i | \theta) = \prod_{k=1}^c \theta_k(\mathbf{x}_i)^{y_{ik}}$$

Given N training samples and assume sample independence.

- **Binary classification:**

$$L = \prod_{i=1}^N p(\mathbf{x}_i, y_i | \theta) = \prod_{i=1}^N \theta(\mathbf{x}_i)^{y_i} (1 - \theta(\mathbf{x}_i))^{1-y_i}$$

- **Multi-class classification:**

$$L = \prod_{i=1}^N p(\mathbf{x}_i, y_i | \theta) = \prod_{i=1}^N \prod_{k=1}^c \theta_k(\mathbf{x}_i)^{y_{ik}}$$

- **Binary classification:** Assume the class label follows Bernoulli distribution.

$$p(y|\theta) = \theta^y (1-\theta)^{1-y} = \begin{cases} \theta, & \text{if } y = 1, \\ 1-\theta, & \text{if } y = 0. \end{cases}$$

- **Multi-class classification:** Assume the class label follows categorical (multinomial) distribution.

$$p(\mathbf{y}|\theta_1, \theta_2, \dots, \theta_c) = \prod_{k=1}^c \theta_k^{y_k}$$

1-of-K coding scheme:

$$\mathbf{y} = [y_1, y_2, \dots, y_k]^T$$

$y_k = 1$ if the sample is from class k

$y_k = 0$ otherwise

Assumption:

1. **Binary \Rightarrow** class label follow Bernoulli distribution
2. **Multi-class \Rightarrow** class labels follow multinomial distribution

为一个基于 likelihood 的损失函数。

Negative Log likelihood Loss

$$L = \prod_{i=1}^N \prod_{k=1}^c \theta_k(\mathbf{x}_i)^{y_{ik}}.$$

- Classification loss: negative log likelihood.
- Negative log likelihood loss is equal to the cross-entropy loss, if

$$\theta(\mathbf{x}) = p(c_1 | \mathbf{x})$$

$$\theta_k(\mathbf{x}) = p(c_k | \mathbf{x})$$

Ch 6: Training & Optimisation

1. Principle & Usage of normal equation

损失函数: $O(\theta_0, \dots, \theta_m)$.

损失函数优化: $\min O(\theta_0, \dots, \theta_m) \Rightarrow$ 目标: $\frac{\partial O}{\partial \theta_0} = \frac{\partial O}{\partial \theta_1} = \dots = \frac{\partial O}{\partial \theta_m} = 0$.

Example. 使用 SSE (sum-of-square error) 作为损失函数.

$$\begin{array}{l} \text{single} \\ \text{input} \end{array} \left| \begin{array}{l} 1. \text{ linear model} \quad \hat{y}_i = \mathbf{w}^T \cdot \tilde{\mathbf{x}}_i \\ 2. \text{ loss function} \quad O(\mathbf{w}) = \frac{1}{2} \| \tilde{\mathbf{x}} \cdot \mathbf{w} - \mathbf{y} \|_2^2. \quad (\text{sum of square error loss / objective function}) \end{array} \right.$$

$$\begin{array}{l} \text{multiple} \\ \text{input} \end{array} \left| \begin{array}{l} 1. \hat{y}_i = \mathbf{w}^T \tilde{\mathbf{x}}_i \\ 2. O(\mathbf{w}) = \frac{1}{2} \| \tilde{\mathbf{x}} \cdot \mathbf{w} - \mathbf{y} \|_2^2. \end{array} \right.$$

normal equation 由 linear least square 得到:

linear least square 损失函数核心部分 " $\tilde{\mathbf{x}} \cdot \mathbf{w} - \mathbf{y}$ " 或 " $\tilde{\mathbf{x}} \cdot \mathbf{w} - \mathbf{y}$ ". ($\tilde{\mathbf{x}}$: expanded feature matrix).

\Rightarrow 损失函数核心: 使得 $\tilde{\mathbf{x}} \cdot \mathbf{w} - \mathbf{y}$ 为 0 的 \mathbf{w} .

$$\mathbf{x} \in \mathbb{R}^d$$

\Rightarrow 利用 normal equation (datapoint 数多于 feature dim, 即 $n > d$, 即 $N > d$)

normal equation's principle:

$$O(\mathbf{w}) = \frac{1}{2} \| \tilde{\mathbf{x}} \cdot \mathbf{w} - \mathbf{y} \|_2^2.$$

目标: 使得 $\nabla O(\mathbf{w}) = 0$: $\mathbf{w} = (\tilde{\mathbf{x}}^T \tilde{\mathbf{x}})^{-1} \tilde{\mathbf{x}}^T \mathbf{y}$.

$$\nabla O(\mathbf{w}) = \tilde{\mathbf{x}}^T \tilde{\mathbf{x}} \mathbf{w} - \tilde{\mathbf{x}}^T \mathbf{y}.$$

normal equation for least square problem.

normal equation 原理式:

1. for linear least square model:

$$N > d: \mathbf{w} = (\tilde{\mathbf{x}}^T \tilde{\mathbf{x}})^{-1} \tilde{\mathbf{x}}^T \mathbf{y} \quad (\text{single output})$$

over-determined

$$\mathbf{w} = (\tilde{\mathbf{x}}^T \tilde{\mathbf{x}})^{-1} \tilde{\mathbf{x}}^T \mathbf{y} \quad (\text{multiple output}).$$

$$N < d: \mathbf{w} = \tilde{\mathbf{x}}^T (\tilde{\mathbf{x}} \tilde{\mathbf{x}}^T)^{-1} \mathbf{y} \quad (\text{single output})$$

under-determined

$$\mathbf{w} = \tilde{\mathbf{x}}^T (\tilde{\mathbf{x}} \tilde{\mathbf{x}}^T)^{-1} \mathbf{y} \quad (\text{multiple output}).$$

2. for L_2 -regularisation linear least square model: (只看 $N > d$ 情形)

$$\mathbf{w} = (\tilde{\mathbf{x}}^T \tilde{\mathbf{x}} + \lambda \mathbf{I})^{-1} \tilde{\mathbf{x}}^T \mathbf{y}.$$

λ 为超参数: control balance between data-dependent error function and regularisation term.

用途: 依 normal equation 构造 optimal 权重的计算式并求解.

The quantity $(\tilde{\mathbf{x}}^T \tilde{\mathbf{x}})^{-1} \tilde{\mathbf{x}}^T$ and $\tilde{\mathbf{x}}^T (\tilde{\mathbf{x}} \tilde{\mathbf{x}}^T)^{-1}$ are called Moore-Penrose inverse (or called pseudoinverse) of $\tilde{\mathbf{x}}$.

称 $(\tilde{\mathbf{x}}^T \tilde{\mathbf{x}})^{-1} \tilde{\mathbf{x}}$ 和 $\tilde{\mathbf{x}}^T (\tilde{\mathbf{x}} \tilde{\mathbf{x}}^T)^{-1}$ 为 pseudo matrix.

适用于任何 \mathbf{x} . 即使 \mathbf{x} 为 singular 时.

- Controls the balance between the data dependent error function and the regularisation term.
 - To distinguish it from those model parameters to be optimised during the learning process (e.g., \mathbf{W}, \mathbf{b}), λ is referred to as a hyperparameter.
- Training, validation and testing
- Training set: A set of samples used for learning, that is to optimise the model parameters.
 - Validation set: A set of samples used to select (tune) the hyperparameters.
 - Test set: A set of samples used only to assess the performance of a fully-specified classifier (trained with selected hyperparameters).

λ 的选择:

用 cross-validation 取最优.

注意 train set, validation set, test set.

2. Derivation of least square model's solution

计算方法：1. 直接从所得的 objective function 中取每个权重值的对应偏导。让所有偏导数 $\nabla O(w_0), \nabla O(w_1), \dots, \nabla O(w_n)$ 全为0，计算出相应的权重值。

2. 利用 normal equation 计算 $w = (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T Y$. (或 $w = \tilde{X}^T (\tilde{X} \tilde{X}^T)^{-1} Y$).

3. Principle of Gradient Descent

梯度下降法：目标仍为最小化 loss function. (objective function).

原理：损失函数视为一个关于权重 w 的函数。

随机猜一个 initial weight. 由 $O(w)$ 关于权重各分量的偏导（梯度）决定 w 各分量的优化方向。最终目标仍为：让 $\nabla O(w) = 0$ ，最小化 $O(w)$.

分量的优化方式： $w^{(n+1)} = w^{(n)} - \eta \cdot \nabla O(w^{(n)})$.

学习率 learning rate: 越高 敏. η 小 \Rightarrow “学的慢”。 η 大 \Rightarrow “学过头”。

梯度下降法变体：

Recall the error function computed using all the training samples: $O(w) = \sum_{i=1}^N O_i(w)$

- Gradient descent (GD) computes the gradient of the error function using all the training samples.
 $w^{(t+1)} = w^{(t)} - \eta \nabla O(w^{(t)}) = w^{(t)} - \eta \sum_{i=1}^N \nabla O_i(w^{(t)})$
Update using all the N training samples.
- Stochastic gradient descent (SGD) estimates the gradient of the error function using one training sample.
 $w^{(t+1)} = w^{(t)} - \eta \nabla O_i(w^{(t)})$ pick 1
Update using only one training sample.
- Mini-batch gradient descent (MBGD) estimates the gradient of the error function using a small set of training samples.
 $w^{(t+1)} = w^{(t)} - \eta \sum_{n \in I} \nabla O_i(w^{(t)})$ I denotes a small set of training samples.
Update using a subset of training samples.

GD: 所有全算。

SGD: 随机算一个。

Mini-Batch: 算一小块。

4. Derive weight updating equation for least squares model:

构造损失函数 $O(w) \sim$ 对 $O(w)$ 关于 w 每个分量求偏导。

Ch. 7. Artificial Neural Network

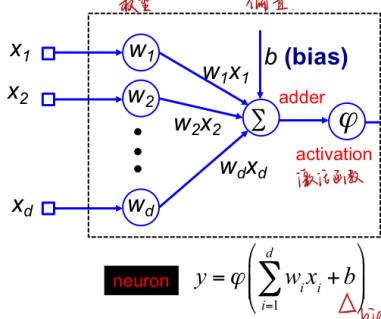
1. Single neuron model (adder & activation) + single layer perceptron.

人工神经网络：对生物神经元的模拟。

1. Single neuron model :

adder:

- Multiple inputs $[x_1, x_2, \dots, x_d]$ and one output y .



Given d input, a neuron is modeled by $d+1$ parameters.

Basic elements of a typical neuron include:

- A set of **synapses or connections**. Each of these is characterised by a weight (strength).
- An **adder** for summing the input signals, weighted by the respective synapses.
- An **activation function**, which squashes the permissible amplitude range of the output signal.

上层连接

adder (和权值线性组合).

激活函数.

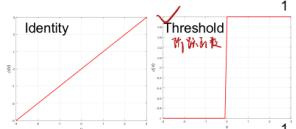
of params: $n+1$. # of weights: n .

Input: $[x_1, \dots, x_d]$. Output: $y = \varphi(w_1x_1 + \dots + w_dx_d + b)$. 类似于多输入单输出的线性模型。

常用激活函数:

activation:

- Identify function: $\varphi(v) = v$



- Threshold function:

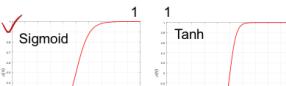
$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ -1 & \text{if } v < 0 \end{cases}$$

注：对 regression problem: 线性输出，用 Identity (linear) activation function.

- Sigmoid function ("S"-shaped curve):

$$\varphi(v) = \frac{1}{1 + \exp(-v)} \in (0, 1) \text{ or}$$

$$\varphi(v) = \tanh(v) = \frac{\exp(2v) - 1}{\exp(2v) + 1} \in (-1, 1)$$



- Rectified linear unit (ReLU):

$$\varphi(v) = \begin{cases} v & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases}$$



注：对 binary classification: 模型模型。

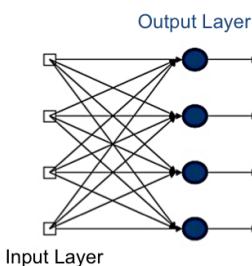
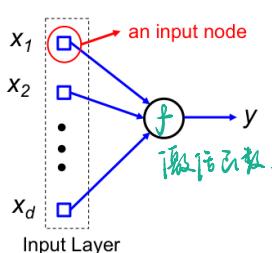
用 sigmoid activation function 获得选择概率。

对 computer vision : ReLU 常用。

2. Single Layer Perceptron:

一个输入数位数，即 input layer 和一个输出数位数，即 output layer.

A single layer perceptron has one input layer and one output layer.



An SLP is similar to a multi-input multi-output linear model.

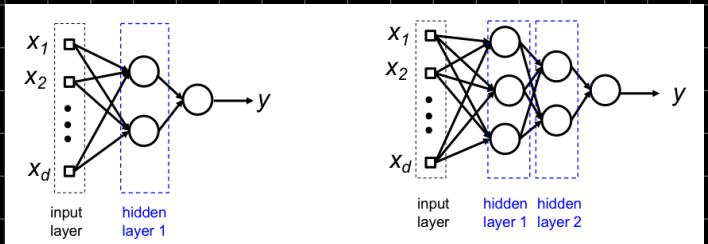
注：输出数位数：

类似于 multi-input, multi-output
对线性模型。

3. multi-layer perceptron + feed-forward neural network

network calculations given weights & inputs, calculate # of parameters given layer # and layer size

Hidden layer: 从 input layer 到 output layer 间的隐藏层。



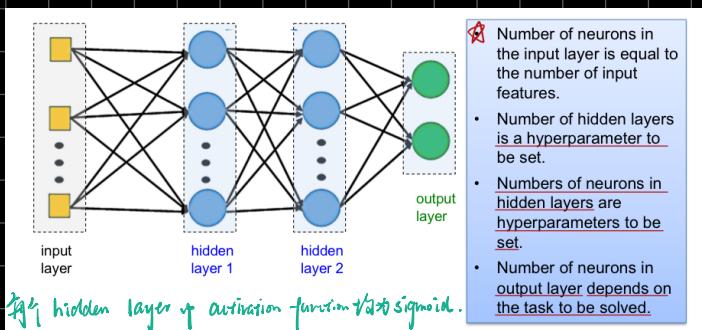
formulate complex non-linear functions.

each hidden layer find a partial solution to the problem, combine in next layer.

Multi-layer Perceptron (MLP).

多层感知机，又称前馈神经网络 (feed-forward artificial neural network).

含至多三层节点：输入 ~ 隐 ~ 藏 ~ 输出.



input 层节点数 = 输入 feature 数

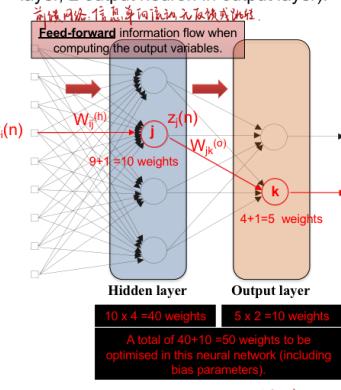
(节点数)

层数和每层结构：hyper parameter.

output 层节点数：看具体问题而定.

网络节点上值的计算：

- An MLP example with one hidden layer consisting of 4 hidden neurons. It takes 9 input features and returns 2 output variables (9 input neurons in input layer, 2 output neuron in output layer).



- Output of the j-th neuron in the hidden layer ($j=1,2,3,4$), for the n-th training sample:

$$z_j(n) = \varphi \left(\sum_{i=1}^9 w_{ij}^{(h)} x_i(n) + b_j^{(h)} \right)$$

- Output of the k-th neuron in the output layer ($k=1,2$), for the n-training sample:

$$y_k(n) = \varphi \left(\sum_{j=1}^4 w_{jk}^{(o)} z_j(n) + b_k^{(o)} \right)$$

计算值 ~ 将线性组合 $w_1x_1 + \dots + w_nx_n + b$
代入激活函数 $\varphi(x)$.

参数计算 ~ 每层每个节点参数量为 $n+1$.
(Tip: # of weights + # of bias)

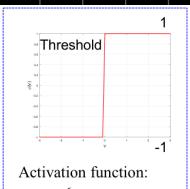
3. Principle & Usage of Perceptron Algorithm

Perceptron Algorithm:

- When the activation function is set as the threshold function, the model is still linear, and it is known as the perceptron of Rosenblatt.

single neuron model,
a psychogist
$$\text{output} = \begin{cases} 1 & \text{if } \mathbf{w}^T \mathbf{x} \geq 0 \\ -1 & \text{if } \mathbf{w}^T \mathbf{x} < 0 \end{cases}$$

for binary classification



使用 threshold function 作为激活函数的人工神经元.

Application: Binary classification.

updating weight:

1. 只用误分类的 Sample!

2. 更新规则:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \eta \cdot y_i \tilde{\mathbf{x}}_i.$$

目标: 最小化 perceptron criterion:

- Perceptron criterion assesses classification error. (另一个目标函数)

$$O(\mathbf{w}) = - \sum_{i \in \text{Misclassified Set}} y_i (\mathbf{w}^T \tilde{\mathbf{x}}_i)$$

If a sample is correctly classified, applies an error penalty of zero; if incorrectly classified, applies an error penalty of the following quantity:

$$|y_i (\mathbf{w}^T \tilde{\mathbf{x}}_i)| = -y_i (\mathbf{w}^T \tilde{\mathbf{x}}_i)$$

正确分类=0误差
错误分类=-y_i·w^T·x̃_i

亦可用 SGD 优化. ($O(\mathbf{w}) = -y_i \mathbf{w}^T \tilde{\mathbf{x}}_i$, $\nabla O(\mathbf{w}) = -y_i \tilde{\mathbf{x}}_i$)

- Stochastic gradient descent (SGD) is used to minimise the perceptron criterion.

- Calculate the error using one misclassified sample:

$$O_i(\mathbf{w}) = -y_i \mathbf{w}^T \tilde{\mathbf{x}}_i$$

- Calculate its gradient: $\frac{\partial O_i(\mathbf{w})}{\partial \mathbf{w}} = -y_i \tilde{\mathbf{x}}_i$

- SGD update: $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \eta y_i \tilde{\mathbf{x}}_i$ y_i : true label

\Rightarrow SGD 实际上 cover 了 perceptron algorithm.

4. Training Method for neural network:

{ Hebbian Learning Rule "用进废退"
Gradient Based Training

Hebbian Learning Rule: "ancient training rule"

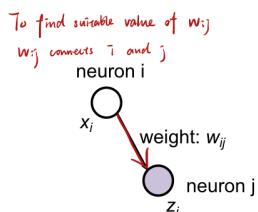
- Start with random value: $w_{ij} = \text{random}$ (...)
- each iteration update a bit: $w_{ij} \leftarrow w_{ij} + \Delta w_{ij}$.

- The weight change is formulated as

$$\Delta w_{ij} = F(x_i, z_j)$$

simplest form

- The simplest form of the function F think: "activated same $\Rightarrow \Delta w_{ij} > 0$ "
 $\Delta w_{ij} = \eta x_i z_j$ problem: what is η ?
 η : activated asymptotically $\Rightarrow \Delta w_{ij} \rightarrow 0$.



For this neuron:
(pre-synaptic signal: x_i)
(post synaptic signal: z_j)

$$\begin{aligned} w_{ij}^{(n+1)} &= w_{ij}^{(n)} + \Delta w_{ij}^{(n+1)} \\ \Delta w_{ij} &= \eta \cdot x_i \cdot z_j \end{aligned}$$

神经元 i 当前值 神经元 j 当前值.

can be unstable.

- Can be unstable.

\Rightarrow usually used in biological-inspired neural network

Gradient-based Training: (most commonly used)

A commonly used approach:

- Treat $z = \phi(\mathbf{x}, \mathbf{W}_{NN})$ as the new features, to be used as the input of a linear predictor.
- A loss function is minimised to find the optimal neural network weights \mathbf{W}_{NN} together with the weights of the linear predictor \mathbf{W}_P .
- Training (optimisation) methods: stochastic gradient descent, mini-batch gradient descent.
- More accurate and stable than Hebbian learning rules in practice.

类似地会连接到 sigmoid 层 (linear predictor).

5. Basic idea of Back Propagation.

误差反向传播法：实际上就是链式求导法则，从后往前利用各层节点值的计算函数，算出每一个节点的偏导（更新方向）。

- It uses the **chain rule** to iteratively compute the gradient for each layer.

Given $z(y(x))$, chain rule: $\frac{dz}{dx} = \frac{dz}{dy} \times \frac{dy}{dx}$ 链式法则：计算复合函数偏导数

- It can be viewed as a process of calculating the error contribution of each neuron after processing a batch of training data.

计算例子：见 Exercise 例题。

- After choosing a loss function $\text{Loss}(\mathbf{W}_{NN})$, a regularisation term, e.g., the sum of the all the squared weights, is added to the final optimisation objective function for training:

$$O(\mathbf{W}_{NN}) = \text{loss}(\mathbf{W}_{NN}) + \lambda \frac{1}{2} \|\mathbf{W}_{NN}\|_2^2$$

regularisation term

prevents over-fitting: when training data isn't sufficient

comes from optimizing loss function

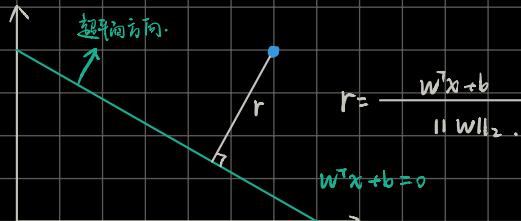
prevents it from fitting too much

- This is called L_2 -regularisation. The regularisation term plays the same role as in the regularised least squares model.
- The regularisation parameter $\lambda > 0$ is a hyper-parameter.

Ch 8. Support Vector Machine

1. Key concepts & basic idea of SVM: (separation margin, support vector, slack variables).

1. 背景知识: 超平面和点到超平面.

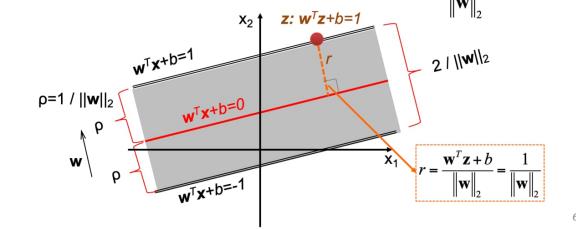


2. SVM: 用平行两个超平面: $w^T x + b = \pm 1$.

- We focus on two parallel hyperplanes:

$$\begin{cases} w^T x + b = 1 \\ w^T x + b = -1 \end{cases}$$

- Geometrically, distance between these two planes is $\frac{2}{\|w\|_2}$



目标: 对线性可分的数据点, 通过最小化 weight, 最大化 hard separation margin.

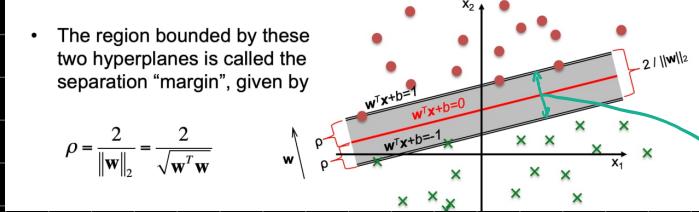
- Given two parallel hyperplanes below, we separate two classes of data points by preventing the data points from falling into the margin:

$$\begin{cases} w^T x + b \geq 1, & \text{if } y = 1, \\ w^T x + b \leq -1, & \text{if } y = -1. \end{cases} \quad \xrightarrow{\text{equivalent expression}} \quad y(w^T x + b) \geq 1$$

- The region bounded by these two hyperplanes is called the "margin", given by

$$\rho = \frac{2}{\|w\|_2} = \frac{2}{\sqrt{w^T w}}$$

"find an optimal hyperplane to separate the two classes of datapoints with widest margin".



Separation Margin

Hard - Margin SVM:

find optimal hyperplane to fully separate two class of datapoints with widest margin.

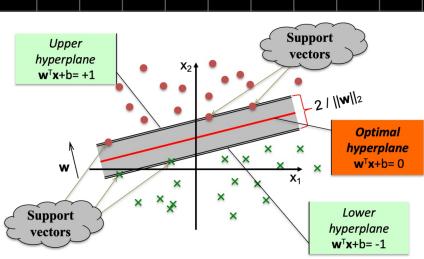
- Hard-margin SVM finds an optimal hyperplane to fully separate the two classes of data points with the widest margin, by solving the following constrained optimisation problem:

$$\min_{w,b} \frac{1}{2} w^T w$$

s.t.

$$y_i(w^T x_i + b) \geq 1 \quad \forall i \in \{1, \dots, N\}$$

Stopping training samples from falling into the margin.



Support Vector:

$$y_i (w^T x_i + b) = 1 \quad (\text{for hard margin SVM})$$

Capable of solving linear separable data patterns.

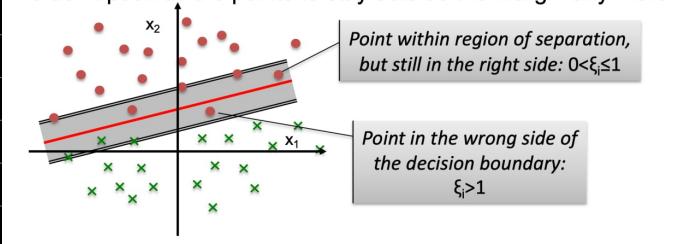
Soft - margin SVM

introduced slack variable $\xi_i \geq 0$, $i=1, 2, \dots, N$, in order to relax hard - margin SVM's constraint.

$$\begin{cases} w^T x_i + b \geq 1 - \xi_i, & \text{if } y_i = 1, \\ w^T x_i + b \leq -(1 - \xi_i), & \text{if } y_i = -1. \end{cases} \quad \xrightarrow{\text{equivalent expression}} \quad y_i (w^T x_i + b) \geq 1 - \xi_i$$

ξ_i : 表示第*i*个数据点, 而定, 衡量其超过 separation margin 的程度.

We don't push all the points to stay outside the margin any more.



Kernel SVM: 应用核函数将 SVM dual problem 映射到另一空间上，在那里问题线性可分。

The SVM dual problem in the original space:

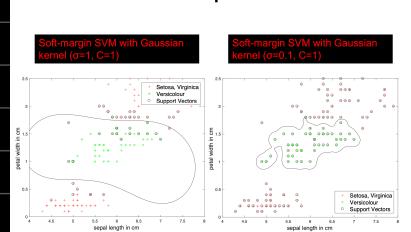
$$\begin{aligned} \text{Dual problem} \\ \max_{\lambda \in \mathbb{R}^N} & \sum_{i=1}^N \lambda_i - \frac{1}{2} \sum_{i,j=1}^N \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{s.t.} & \sum_{i=1}^N \lambda_i = 0 \\ & 0 \leq \lambda_i \leq C \end{aligned}$$

Move from the original space to a kernel induced space.

Kernel SVM with the modified dual problem:

$$\begin{aligned} \text{Dual problem} \\ \max_{\lambda \in \mathbb{R}^N} & \sum_{i=1}^N \lambda_i - \frac{1}{2} \sum_{i,j=1}^N \lambda_i \lambda_j y_i y_j \mathbf{\Phi}(\mathbf{x}_i)^T \mathbf{\Phi}(\mathbf{x}_j) \\ \text{s.t.} & \sum_{i=1}^N \lambda_i y_i = 0 \\ & 0 \leq \lambda_i \leq C \end{aligned}$$

Iris Classification Example:



适用数据非线性的情况。

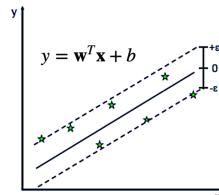
3. Support Vector Regression:

One simple way to formulate a linear SVM regressor:

$$\min_{(\mathbf{w}, b) \in \mathbb{R}^{d+1}} \frac{1}{2} \|\mathbf{w}\|_2^2$$

$$\text{subject to } |\mathbf{w}^T \mathbf{x}_i + b - y_i| \leq \epsilon, i = 1, 2, \dots, N$$

The parameter ϵ is a hyperparameter.



目标：让超平面 $w^T x + b = 0$ 尽可能多地接近现有 datapoints.

It can be extended to fit nonlinear data patterns by using kernel trick.

4. SVM Training:

Hard-Margin SVM

Hard-margin SVM training: the process of solving the following constrained optimisation problem:

$$\begin{aligned} \min_{\mathbf{w}, b} & \frac{1}{2} \mathbf{w}^T \mathbf{w} \\ \text{s.t.} & y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \quad \forall i \in \{1, \dots, N\} \end{aligned}$$

How to derive the dual form can be found in the SVM notes as optional reading materials.

The above problem is solved by solving a dual problem as shown below.

$$\begin{aligned} \text{Dual problem} \\ \max_{\lambda \in \mathbb{R}^N} & \sum_{i=1}^N \lambda_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{s.t.} & \sum_{i=1}^N \lambda_i y_i = 0 \\ & \lambda_i \geq 0 \end{aligned}$$

Soft-margin SVM

In addition to maximising the margin as before, we need to keep all slacks ξ_i as small as possible to minimise the classification errors. The modified SVM optimisation problem becomes:

$$\begin{aligned} \min_{\substack{(\mathbf{w}, b), \xi \\ [\lambda]_{\geq 0}}} & \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^N \xi_i \\ \text{s.t.} & y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \quad \forall i \in \{1, \dots, N\} \end{aligned}$$

$C > 0$ is a user defined parameter, which controls the regularisation. This is the trade-off between complexity and nonseparable patterns.

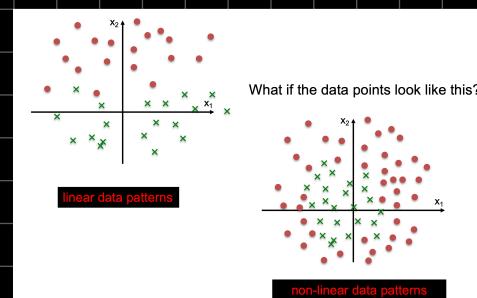
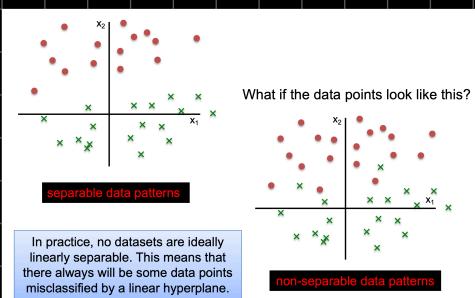
The above constrained optimisation problem can be converted to a QP problem.

$$\begin{aligned} \text{Dual problem} \\ \max_{\lambda \in \mathbb{R}^N} & \sum_{i=1}^N \lambda_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{s.t.} & \sum_{i=1}^N \lambda_i y_i = 0 \\ & 0 \leq \lambda_i \leq C \end{aligned}$$

Soft-margin SVM

同时最小化松弛变量。

5. Linear / non-linear, separable / non-separable data patterns



3. Usage of different types of SVM.



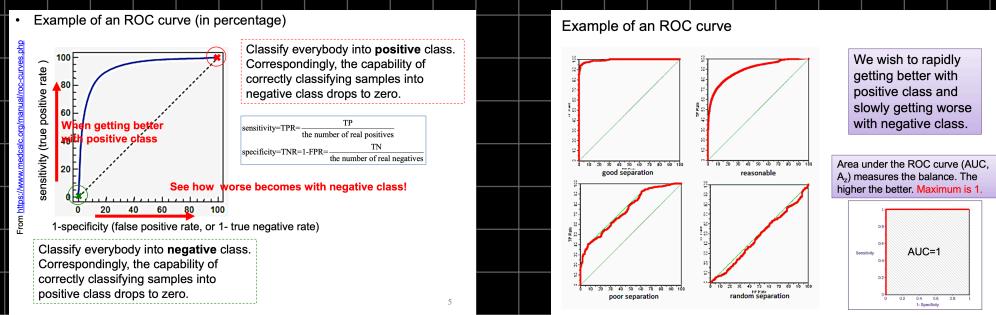
4. Pros & Cons of different machine learning models

1. ROC (Receiver Operating Characteristic) curve

x 轴: FPR ($1 - \text{specificity}$, $\frac{FP}{FP + TN}$)

y 轴: TPR (recall rate, sensitivity, $\frac{TP}{TP + FN}$)

用途: observe performance change over a varying threshold. (threshold is binary classifier's threshold).



2. different ML models

k-NN

Pros (+):

- A simple intuitive method with no training.
- Can be used for both classification and regression.
- Can handle both linear and nonlinear data patterns.
- Easy to implement for multi-class classification.
- There are only two things to decide, which can be relatively easy.
 - What hyper-parameter value k to use?
 - What distance measure to use?

Cons (-):

- Slow with large-scale data.
 - The computational complexity of distance calculation suffers from curse of dimensionality.
 - The computational complexity of neighbour search suffers from training data size.
- High memory cost (needs to store all the training data)
- Not good at dealing with imbalanced data.
- Sensitive to outliers.

THE UNIVERSITY OF MANCHESTER 1824
The University of Manchester

(Regularised) Linear Least Squares

1824
The University of Manchester

Pros (+):

- The most widely used modelling method.
- It is what most people mean when they say they have used "regression", "linear regression" or "least squares" to fit a model to their data.
- Can be used for both regression and classification.
- Makes efficient use of the data. Good results can be obtained with relatively small data.
- Easy to explain and understand.
- Low computational cost: fast training and prediction.
- Low memory.
- In the least squares case, there is no hyper-parameter to set. In the regularised case, there are two things to decide: (1) the form of regularisation term (e.g., L_1 , L_2), and regularisation parameter.

not very expressive

- Cons (-):
- Limited model expressive power, cannot deal with nonlinear data patterns.
 - Sensitive to outliers.
 - When being used in classification, the performance is sometimes sensitive to how the target output is set, and it does not offer probabilistic interpretation.

Logistic Regression

The University of

Pros (+):

- A simple and effective method for classification.
- Low computational cost: fast training and prediction.
- Low memory.
- Offer nice probabilistic interpretation.
- It does not assume specific distribution, e.g., Gaussian, in the model.
- No need to set hyper-parameter.

Cons (-):

- It is a linear classifier and cannot handle nonlinear data patterns.
- Not very good with multi-class classification.
- It is for classification only.

Artificial Neural Network

IS24
The University of Manchester

Pros (+):

- Capable of learning and modelling non-linear and complex relationships.
- It can generalise, supported by advanced training strategies.
- Does not impose restrictions and assumptions to the input.
- Learned information is all stored in network weights. 可迁移性高.
- Good choice for black-box modelling.

Cons (-):

- Need to determine a "good" network architecture, which is sometimes not easy.
- Long training time for large-scale data.
- Difficult to interpret the learned model.
- The solution is a local optimal.

Support Vector Machines

IS24
The University of Manchester

Pros (+):

- Usually provide competitive performance. It is a good choice when we have no idea on the data.
- Unlike ANN, it is not solved for local optima. globalness
- Scale relatively well to high dimensional data.
- It can generalise.
- Can handle both linear and nonlinear data patterns. (kernel SUM & linear SUM)

Cons (-):

- Need to choose a "good" kernel and select hyper-parameters, which are not easy.
- Long training time for large-scale data.
- Difficult to interpret the learned model.

3. No free lunch Theorem:

There is no context-independent or problem-independent reason to favour one method over another. You **always** need to

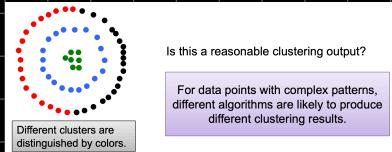
- Understand the data/task/problem to be processed/solved first!
- Experiment with different machine learning models!

Ch 9. Cluster Analysis

1. Definition, key concepts & tasks in clustering

Definition: 将相似的物体聚类在同一组内，不相似的留在其他组。 (unsupervised learning)

- Key Tasks:
1. define an appropriate distance (similarity) measure.
 2. identify a cluster number.
 3. specify clustering algorithm to group objects into sensible clusters.
 4. evaluate whether the clustering result is good or not.



- Often which distance (similarity) measure to use and what cluster number to adopt is treated as a **model selection** process, where cluster number is a hyper-parameter.
- There are some clustering algorithms that attempt to decide a cluster number for you, but they require to set other type of hyper-parameters.

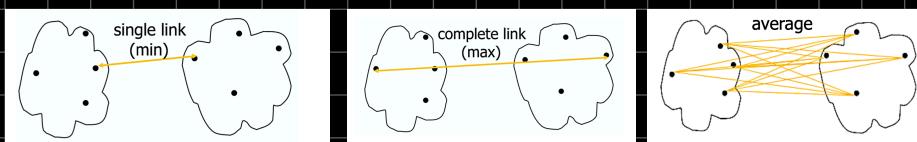
2. Distance calculation between clusters: (single-link, complete-link, average-link)

measure the distance (similarity) between 2 data clusters:

* single link: 两个 cluster 间点距离最小值. $\min_{p \in \text{cluster}_1, q \in \text{cluster}_2} d(x_p, x_q)$

* complete link: 两个 cluster 间点距离最大值. $\max_{p \in \text{cluster}_1, q \in \text{cluster}_2} d(x_p, x_q)$

* average link: 两个 cluster 间点距离平均值. $\frac{1}{n_1 n_2} \sum_{p \in \text{cluster}_1, q \in \text{cluster}_2} d(x_p, x_q)$



3. Principle & usage of K-means algorithm

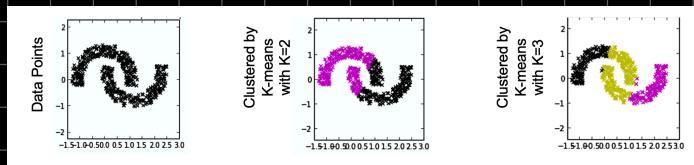
usage: group a set of datapoints to k clusters.

Steps:

- Pre-determine the **cluster number** k .
- Choose your **distance (or similarity) measure**.
- Initialise by **random** selecting k cluster center points.
- Iterate:
 - Step 1: **Calculate distances (or similarities)** between the data points and the cluster center points.
 - Step 2: Find the **nearest** cluster center to each data point, and assign the data point to that cluster.
 - Step 3: **Calculate the new cluster center for each cluster**, by **averaging** its member points. *update cluster center*.
 - Step 4: Go back to Step 1), stop when there is no change in the membership of each cluster.

Issues:

1. sensitive to initial cluster center points.
2. k need to be specified in advance.
3. can't handle noisy data & outliers.
4. not suitable for complex data patterns.



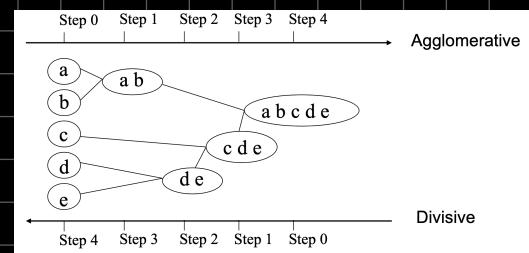
4. Basic idea of hierarchical clustering.

Idea: 1. group objects into tree of clusters \Rightarrow nested partitions layer by layer.

2. partition dataset sequentially, no need to know cluster number.

typical strategies:

- Agglomerative (bottom-up strategy).
 - o Start from treating each data point as a cluster (atomic cluster).
 - o Then **merge** the atomic clusters into larger and larger clusters.
- Divisive (top-down strategy).
 - o Start from treating all data points as one single cluster.
 - o Then **divide** this cluster into smaller and smaller clusters.



5. Principle & usage of agglomerative algorithm:

principle: 从底向上, 每次 iteration 依 distance matrix 中距离最小的两个 cluster 合并。
初始时每个 datapoint 视为一个 cluster. 最后 cluster 变成只有一个。

- Implemented in three steps:
 - Step 1: Calculate the distance matrix D between all the data points.
 - Step 2: Set each data point as a cluster. The between-cluster matrix M is equal to D.
 - Step 3: Repeat the following until the cluster cluster becomes one,
 - Merge the two closest clusters.
 - Update the between-cluster distance matrix M.
- Major drawbacks:
 - Sensitive to cluster distance measures and noise/outliers.
 - Less efficient: $O(N^2 \log N)$, where N is the number of total data points.

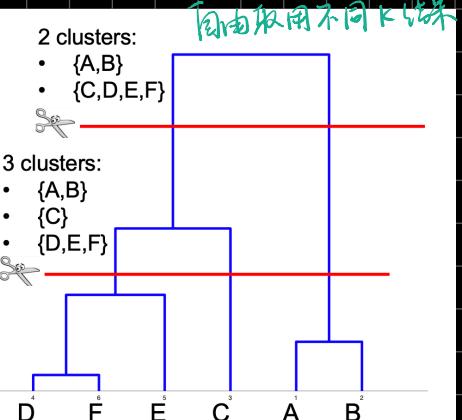
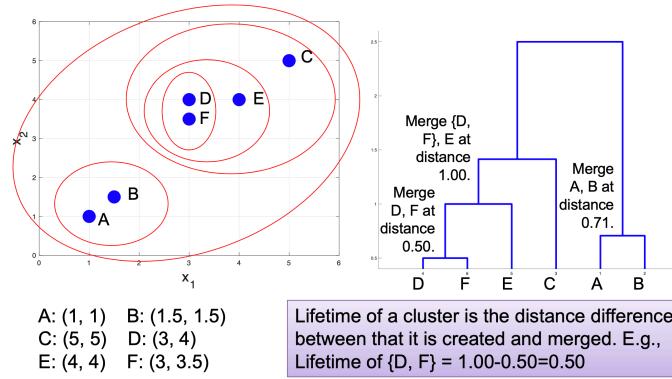
一开始算的就是点间 distance.
可用 Euclid.

可用 single link, complete link,
average link.

注: 某个^{时间} cluster 的“生命周期”:

the distance difference between it's created & merged.

Example: Apply the agglomerate algorithm to the 6 data points below.



6. Typical internal & external indexes for cluster validation

目标: validate if the clustering make sense.

* internal criteria (indexes): 不用外部信息 (如现有的分类结果), 只基于“单次”评价.

* external criteria (indexes): 利用外部信息, validate against ground truth.

Typical Internal Indexes : F-ratio index

- Normally, a good clustering result should have small within-cluster variance while large between-cluster variance (K is cluster number).

- Within-cluster variance (SSW): $SSW = \sum_{i=1}^K \sum_{p \in \text{cluster}_i} d^2(x_p, c_i)$

- Between-cluster variance (SSB): $SSB = \sum_{i=1}^K n_i d^2(c_i, c)$

- F-ratio index: $F = K \frac{SSW}{SSB}$

$$\begin{aligned} n_i &: \text{number of data points in cluster } i \\ c_i &= \frac{1}{n_i} \sum_{p \in \text{cluster}_i} x_p : \text{cluster center} \\ c &= \frac{1}{N} \sum_{p=1}^N x_p : \text{whole data center} \end{aligned}$$

- The above measures can be used to select cluster number.

聚类内点 variance 越小越好

不同聚类中心, 治全样本的中心, 越大越好。

$$F = K \cdot \frac{SSW}{SSB} . \text{ 可用于进行选择合适的 } K.$$

F-ratio

Typical External Validation : Rand Index

validate against a set of ground truth class labels

- It is computed by considering all pairs of data points by looking into agreement and disagreement against the "ground truth".
- It is defined based on the following agreement/disagreement table:

Obviously, a better match has larger a and d, while smaller b and c.

pointwise relationship		Pairs of data points from the same class	Pairs of data points from different classes
Pairs of data points from the same cluster	a = # agreement		b = # disagreement
Pairs of data points from different clusters	c = # disagreement		d = # agreement

- Rand index is computed by $\text{Rand} = \frac{a+d}{a+b+c+d}$

- Assume there are K clusters and C classes, rand index can also be computed from the following contingency table:

C classes

$$\begin{bmatrix} n_{11} & n_{12} & \dots & n_{1C} \\ n_{21} & n_{22} & \dots & n_{2C} \\ \vdots & \vdots & \ddots & \vdots \\ n_{K1} & n_{K2} & \dots & n_{KC} \end{bmatrix}$$

n_{ij} is the number of data points in both cluster i and class j .
 N is the total number of data points.

$$\begin{aligned} a &= \frac{1}{2} \sum_{i=1}^K \sum_{j=1}^C n_{ij} (n_{ij} - 1) \\ b &= \frac{1}{2} \left(\sum_{i=1}^K \left(\sum_{j=1}^C n_{ij} \right)^2 - \sum_{i=1}^K \sum_{j=1}^C n_{ij}^2 \right) \\ c &= \frac{1}{2} \left(\sum_{i=1}^K \left(\sum_{j=1}^C n_{ij} \right)^2 - \sum_{i=1}^K \sum_{j=1}^C n_{ij}^2 \right) \\ d &= \frac{1}{2} \left(N^2 + \sum_{i=1}^K \sum_{j=1}^C n_{ij}^2 - \sum_{i=1}^K \left(\sum_{j=1}^C n_{ij} \right)^2 - \sum_{i=1}^K \left(\sum_{j=1}^C n_{ij} \right)^2 \right) \end{aligned}$$

知道即可, 不考.

Ch 10. Deep learning

1. Basic Deep learning techniques

Definition: deep learning ~ techniques for learning using neural networks.
also considered a kind of representation (feature) learning techniques. e.g.: AlexNet

Advanced neural network architecture:

- CNN: 卷积神经网络.

used to automatically learn good feature vector for image from its pixels.

- RNN: 循环神经网络.

useful for learning from sequential data.

Attention Mechanism:

注意力机制: 权重和 (weight sum) 为 mathematical diagram, 为 weight functions 传递信息.

用途: help identify contributing parts in neural network to decision making.

解释 Neural network 为 Black box 为. make it easier to be interpreted.

- **Black box decision:** A decision supported by model parameters and mathematical operations that are "hard" to understand.



Why are these two images similar?

- **Attention mechanism:** A type of effective mathematical diagrams based on weighted sum, which uses weight functions to locate salient information.
- Embed an attention mechanism to a neural network would help identify the contributing parts to final decision making.

Training tips for Deep neural network:

1. Pre-train: (ResNet, VGGNet trained on ImageNet data)

use a pre-trained neural network if data share similar structure to existing training data that has been used to train some well-known neural networks.

2. Batch Normalisation:

remove mean & scale by standard deviation for hidden layers' output. output-mean std.

3. Regularisation: 此处有手写。

Typical regularisation techniques:

- Parameter norm penalties, remember $O(\mathbf{W}_{NN}) = \text{loss}(\mathbf{W}_{NN}) + \lambda \frac{1}{2} \|\mathbf{W}_{NN}\|_2^2$?
- Dataset augmentation by creating fake data and adding to the training set.
- Add small perturbation (noise) to network weights.
- Early stopping by treating the number of training iterations as a hyper-parameter and finding the best one by using validation data.
- Sparse representation by forcing hidden representation to have more zeros.
- Ensemble methods by training multiple models separately at the same time, and letting them vote the final output.
- Dropout by randomly removing a percentage of neurons during training.

for regularisation 为

for 1范数正则化

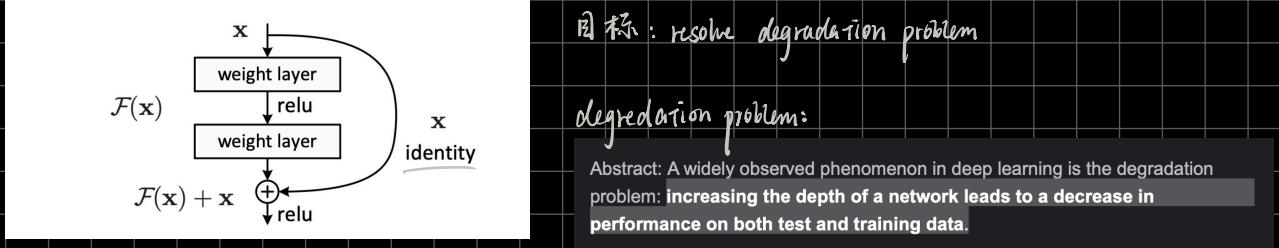
梯度反向传播

梯度下降法 (类似梯度反向操作)

Ensemble Method

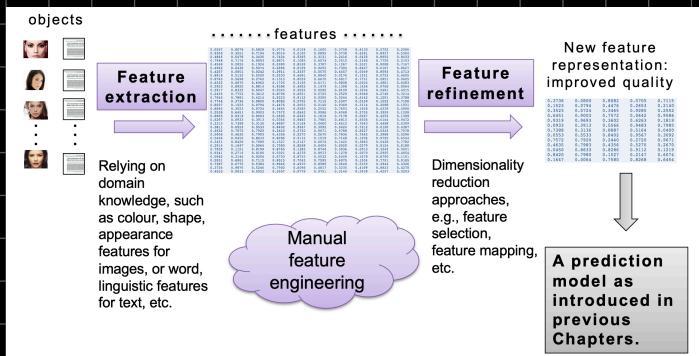
Drop out (类似于“丢弃”)

4. Skip connection. (in ResNet)



2. Differences between deep learning & traditional machine learning strategies

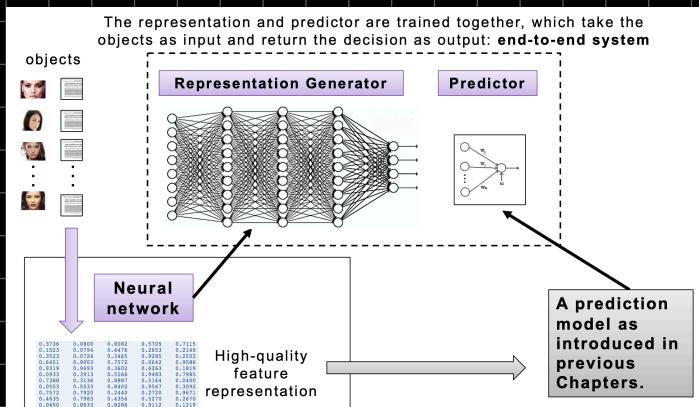
traditional ML: use **manual feature engineering**:
feature extraction ~ feature refinement ~ prediction model.



deep learning: use **neural network** to directly provide high-quality feature representation

① neural network ~~does~~ **trains** feature extraction & feature refinement.

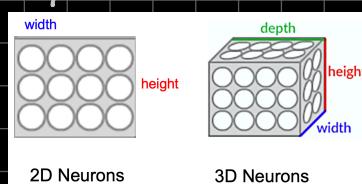
representation and predictor are trained together, take obj as input, return decision as output \Rightarrow end-to-end system.



3. Convolutional Network (architecture, convolutional, pooling, fully-connected layers)

main recipe:

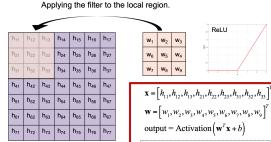
1. 2D / 3D neurons.



2. Convolution Layers 卷积层

receptive field

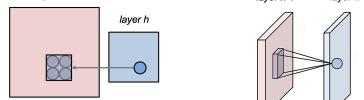
- The operation for applying the filter to a local region of neurons:



卷积核中有权重

仅有 weight sharing

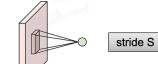
- Local connectivity:** Each neuron inside a layer is connected to only a small region of the previous layer, called a receptive field.



- The output of a neuron is a number computed from the output of those neurons from the corresponding local region and the weights of the convolutional filter: $y = \text{Activation}(w'x + b)$
- Weight sharing:** One same filter of the size of the local region slides over all spatial locations (in different words, slides over all the neurons in the layer).

- Let's look at the case of 3D neurons. **同时包含深度**.

Green cube is a convolutional filter: $F_1(\text{width}) \times F_2(\text{height}) \times d(\text{depth})$

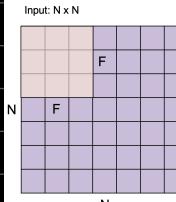


Red cube is one layer of 3D neurons: $N_1(\text{width}) \times N_2(\text{height}) \times d(\text{depth})$

- The resulting activation map based on one convolutional filter is 2D of size:

$$\left(\frac{N_1 - F_1 + 1}{S} + 1 \right) \times \left(\frac{N_2 - F_2 + 1}{S} + 1 \right)$$

Slide the filter over all spatial locations.



"滑动"的步长.

- The sliding process results in an activation map of size:

$$\left(\frac{N - F + 1}{S} + 1 \right) \times \left(\frac{N - F + 1}{S} + 1 \right)$$

- You will need to set appropriate filter size and stride size so that $(N - F)/S$ is an integer.

注：一般公用不止一个光标点

- 1 map 的大小：

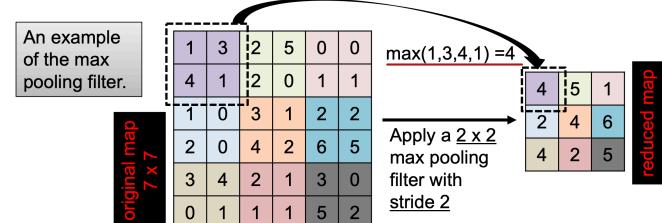
$$(\frac{N_1 - F_1}{S} + 1) \times (\frac{N_2 - F_2}{S} + 1).$$

3. Pooling Layer 池化层

take convolution layer's output: activation map as input, then reduce the size of each activation map. ("down sampling")

池化层 (pooling filter) 用法和卷积核 (convolutional layer) 用法基本一样。

- The pooling layer takes the activation maps returned by a convolutional layer as the input, and reduces the size of each activation map separately.
- It can be viewed as an operation of down-sampling. **降采样**
- A pooling filter slides over all the spatial locations in an activation map in the same way as a 2D convolutional filter.

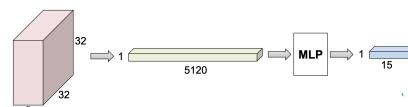


4. Fully connected Layer 全连接层

- The output of a convolutional (or pooling) layer is a set of activation maps, stored in a 3D array.
- These values in the 3D array can be moved to a 1D array. For instance:

$$32 \times 32 \times 5 \text{ array} \rightarrow 5120 \times 1 \text{ array}$$

- The fully connected layers are equivalent to a multilayer perceptron (MLP) taking the generated 1D array as the input.

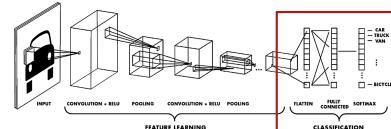


接收卷积层 or 池化层的输出:
a set of activation map. 在 3D matrix 里。

拍扁铺平. 将 1D array 为输出:

5. CNN architecture:

- A CNN is a sequence of convolutional (and pooling) layers, followed by fully connected layers in the end.



- Its unique architecture (2,3D neurons, local connectivity, etc.) makes it suitable for processing 2,3D data with typical local patterns, particularly images ($n \times n \times 3$ input where 3 corresponds to the R, G and B channels).

a sequence of convolutional / pooling layers,

接接着一个全连接层。

因其是 Image.