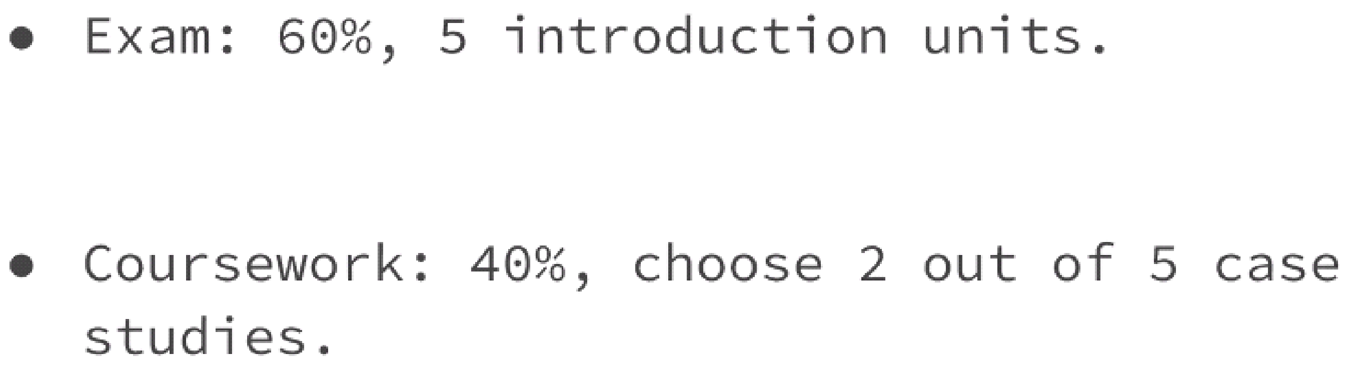
Introduction

23 September 2021 16:10



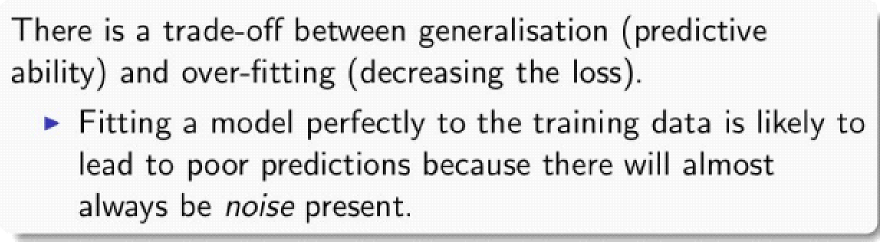
- Exam: 60%, 5 introduction units.

- Coursework: 40%, choose 2 out of 5 case studies.

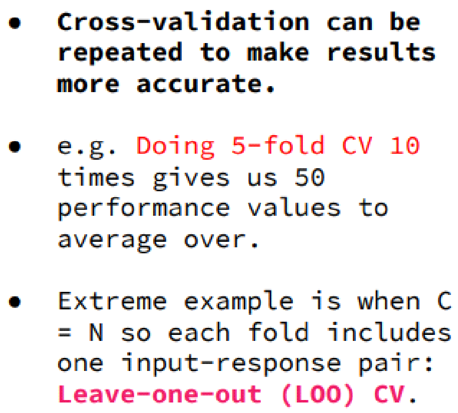
Content

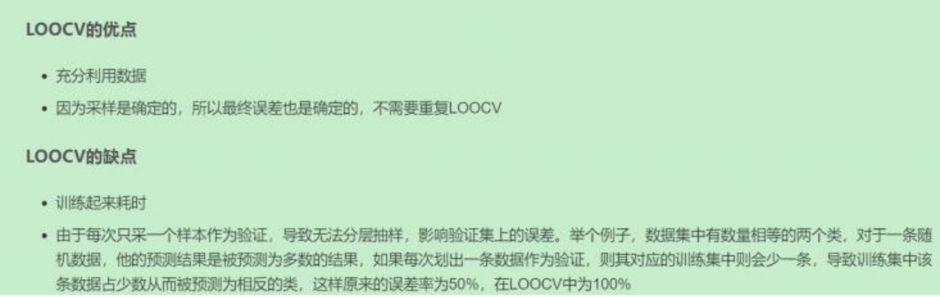
06 December 2021 13:15

Regression

1. Part1
    - a. Key definitions
        - i. Attributes and targets
        - ii. Variables
        - iii. Model
        - iv. Data
        - v. Loss and squared loss
    - b. Linear regression
        - i. $$\widehat{\mathbf{w}} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{t}$$
2. Part2
    - a. Polynomial Regression
        - i. It is still linear regression
        - ii. Loss always decreases as the model is made more complex, that's why loss on the training data cannot be used to choose models
    - b. Generalization and over-fitting
        - i. 

          There is a trade-off between generalisation (predictive ability) and over-fitting (decreasing the loss).

          - ▶ Fitting a model perfectly to the training data is likely to lead to poor predictions because there will almost always be *noise* present.
    - c. How to choose right model complexity? --cross validation
        - i. 

          - **Cross-validation can be repeated to make results more accurate.**
          - e.g. Doing 5-fold CV 10 times gives us 50 performance values to average over.
          - Extreme example is when C = N so each fold includes one input-response pair: **Leave-one-out (LOO) CV.**
        - ii. Leave-one-out-CV(LOOCV)
            - 1) https://blog.csdn.net/dpengwang/article/details/84934197
            - 2) 

              LOOCV的优点

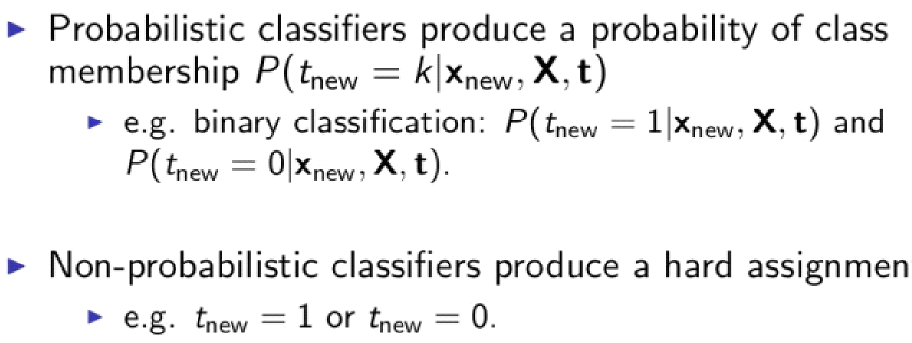
              - 充分利用数据
              - 因为采样是确定的，所以最终误差也是确定的，不需要重复LOOCV

              LOOCV的缺点

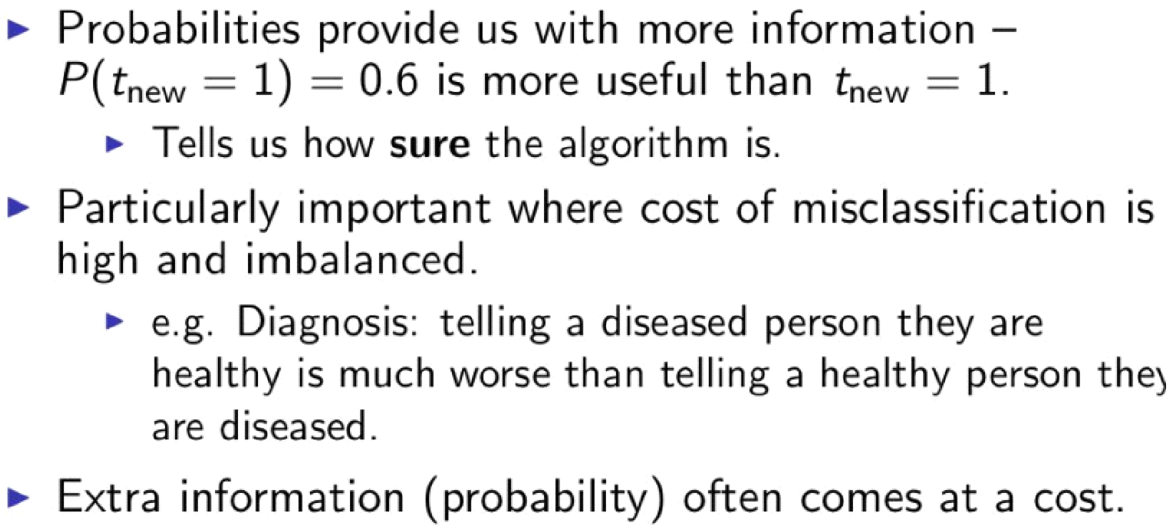
              - 训练起来耗时
              - 由于每次只采一个样本作为验证，导致无法分层抽样，影响验证集上的误差。举个例子，数据集中有数量相等的两个类，对于一条随机数据，他的预测结果是被预测为多数的结果，如果每次划出一条数据作为验证，则其对应的训练集中则会少一条，导致训练集中该条数据占少数从而被预测为相反的类，这样原来的误差率为50%，在LOOCV中为100%
    - d. Common functions
        - i. Polynomial
        - ii. Rbf
        - iii. sigmoid
3. Part3
    - a. Regularization
        - i. Motivation: we do not want the parameters to be too big in absolute value
        - ii. Comparison of linear, ridge and lasso

    b. Lasso
    c. Ridge
    d. Loss at different order

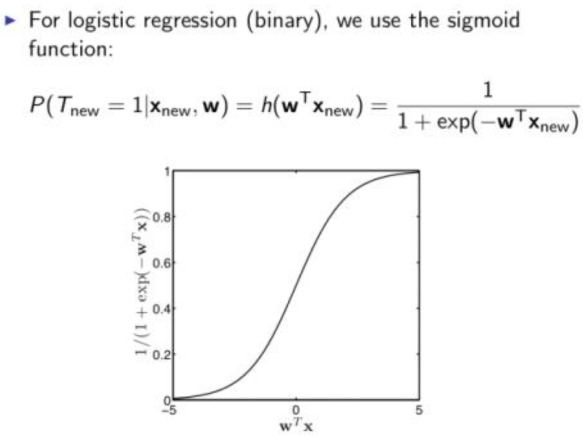
Classification

1. Part 1
    a. Probabilistic classifiers v.s. Non-probabilistic classifiers
        i. 

- Probabilistic classifiers produce a probability of class membership $P(t_{\text{new}} = k|\mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$
    - e.g. binary classification: $P(t_{\text{new}} = 1|\mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$ and $P(t_{\text{new}} = 0|\mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$.
- Non-probabilistic classifiers produce a hard assignment
    - e.g. $t_{\text{new}} = 1$ or $t_{\text{new}} = 0$.

        ii. 

- Probabilities provide us with more information – $P(t_{\text{new}} = 1) = 0.6$ is more useful than $t_{\text{new}} = 1$.
    - Tells us how **sure** the algorithm is.
- Particularly important where cost of misclassification is high and imbalanced.
    - e.g. Diagnosis: telling a diseased person they are healthy is much worse than telling a healthy person they are diseased.
- Extra information (probability) often comes at a cost.

    b. KNN
        i. Non-probabilistic
        ii. As k increases, small classes will disappear
        iii. Cannot work when data size differs a lot
        iv. Choosing K: CV
2. Part 2
    a. Logistic regression
        i. Cannot use linear regression because we want probability.
    b. Statistics
        i. Discrete and continuous random variables
        ii. Joint probabilities and densities
        iii. Dependence and independence
        iv. Conditioning
    c. Sigmoid function
        i. 

- For logistic regression (binary), we use the sigmoid function:

$$P(T_{\text{new}} = 1|\mathbf{x}_{\text{new}}, \mathbf{w}) = h(\mathbf{w}^{\mathsf{T}}\mathbf{x}_{\text{new}}) = \frac{1}{1 + \exp(-\mathbf{w}^{\mathsf{T}}\mathbf{x}_{\text{new}})}$$

ii.

$$\text{if} \quad t_n = 1, \qquad p(t_n = 1|\mathbf{x}_n, \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x}_n)}$$

$$\text{if} \quad t_n = 0, \qquad p(t_n = 0|\mathbf{x}_n, \mathbf{w}) = 1 - p(t_n = 1|\mathbf{x}_n, \mathbf{w})$$

d. Likelihood and loss function for one data point and all data

e. Performance evaluations

  i. 0/1loss: proportion of times classifier is wrong

    1) 离散化的mean square

  ii. Confusion matrices

  iii. Sensitivity

  iv. Specificity

  v. ROC, need a threshold

3. Part3

a. SVM

  i. It finds the decision boundary that maximizes the margin

  ii. $t_n(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b) \geq 1$

  iii. Optimization method

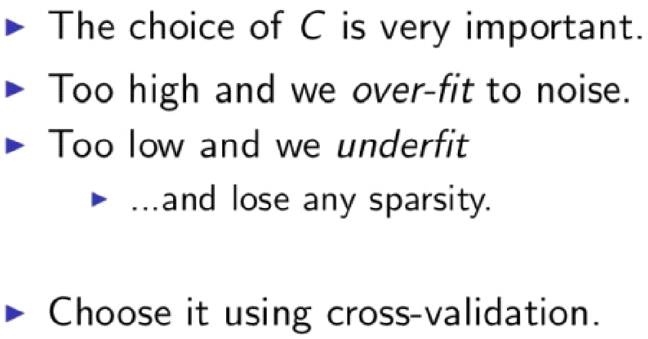
  iv. Draw the boundary line

  v. Support vectors

  vi. Hard margin and soft margin

  vii. 确定C的范围[unknown]

    1)



- The choice of *C* is very important.
- Too high and we *over-fit* to noise.
- Too low and we *underfit*
  - ...and lose any sparsity.
- Choose it using cross-validation.

    2) https://www.cnblogs.com/ceason/articles/12288603.html

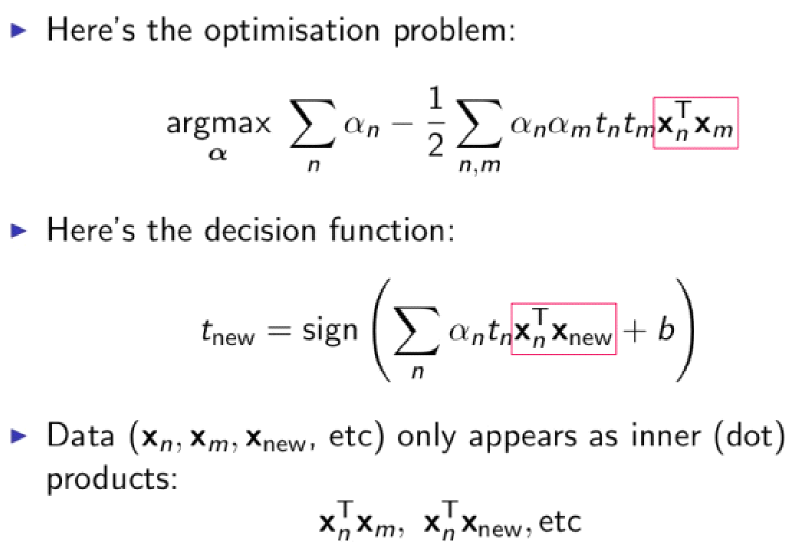
    3) C越大，相当于惩罚松弛变量，希望松弛变量接近0，即对误分类的惩罚增大，趋向于对训练集全分对的情况，这样对训练集测试时准确率很高，但泛化能力弱。C值小，对误分类的惩罚减小，允许容错，将他们当成噪声点，泛化能力较强。

    4) https://blog.csdn.net/transformed/article/details/90437821?utm_medium=distribute.pc_aggpage_search_result.none-task-blog-2~aggregatepage~first_rank_ecpm_v1~rank_v31_ecpm-2-90437821.pc_agg_new_rank&utm_term=sklearn%E4%B8%ADsvc%E5%90%84%E6%A0%B8%E5%87%BD%E6%95%B0%E4%B8%ADgamma%E5%90%AB%E4%B9%89&spm=1000.2123.3001.4430

    5) gamma:核函数系数，只对'rbf','poly','sigmoid'起作用. 默认为'auto'，此时值为样本特征数的倒数，即1/n_features.

  viii. SVM usually applies on small dataset

b. Inner product



i.

- Here's the optimisation problem:

$$\underset{\boldsymbol{\alpha}}{\text{argmax}} \sum_n \alpha_n - \frac{1}{2}\sum_{n,m} \alpha_n\alpha_m t_n t_m \boxed{\mathbf{x}_n^\mathsf{T}\mathbf{x}_m}$$

- Here's the decision function:

$$t_{\text{new}} = \text{sign}\left(\sum_n \alpha_n t_n \boxed{\mathbf{x}_n^\mathsf{T}\mathbf{x}_{\text{new}}} + b\right)$$

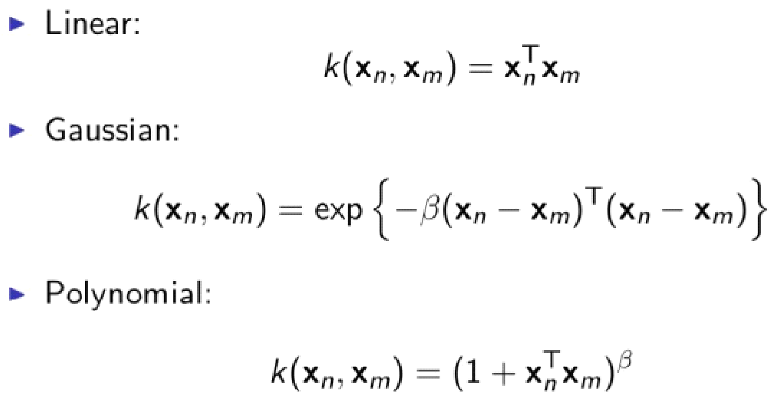
- Data ($\mathbf{x}_n, \mathbf{x}_m, \mathbf{x}_{\text{new}}$, etc) only appears as inner (dot) products:

$$\mathbf{x}_n^\mathsf{T}\mathbf{x}_m, \ \mathbf{x}_n^\mathsf{T}\mathbf{x}_{\text{new}}, \text{etc}$$

ii. Transform data into another domain

iii. 由于是内积, 只要保证最后数值结果不变就不影响最后的结果, 因此可以把数据映射到另一个空间以便更好地分割

c. Kernel



i.

- Linear:

$$k(\mathbf{x}_n, \mathbf{x}_m) = \mathbf{x}_n^\mathsf{T}\mathbf{x}_m$$

- Gaussian:

$$k(\mathbf{x}_n, \mathbf{x}_m) = \exp\left\{-\beta(\mathbf{x}_n - \mathbf{x}_m)^\mathsf{T}(\mathbf{x}_n - \mathbf{x}_m)\right\}$$

- Polynomial:

$$k(\mathbf{x}_n, \mathbf{x}_m) = (1 + \mathbf{x}_n^\mathsf{T}\mathbf{x}_m)^\beta$$

ii. Gaussian kernel

d. Choosing kernel function, parameters and C

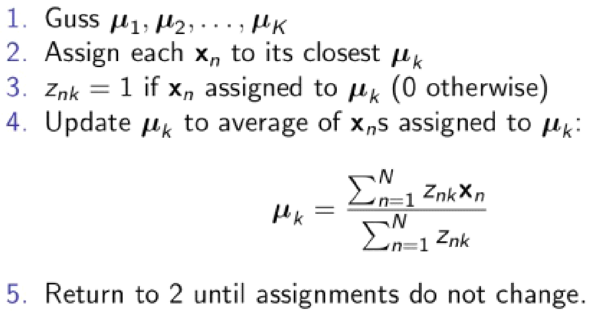
Clustering

1. Part 1

a. Unsupervised learning

b. Clustering

i. By an iterative algorithm



ii.

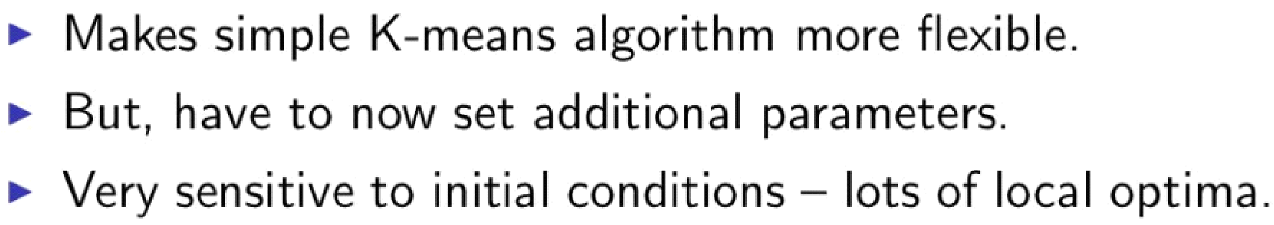
1. Guss $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$
2. Assign each $\mathbf{x}_n$ to its closest $\boldsymbol{\mu}_k$
3. $z_{nk} = 1$ if $\mathbf{x}_n$ assigned to $\boldsymbol{\mu}_k$ (0 otherwise)
4. Update $\boldsymbol{\mu}_k$ to average of $\mathbf{x}_n$s assigned to $\boldsymbol{\mu}_k$:

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^N z_{nk}\mathbf{x}_n}{\sum_{n=1}^N z_{nk}}$$

5. Return to 2 until assignments do not change.

c. k-mean

d. Kernel k-means



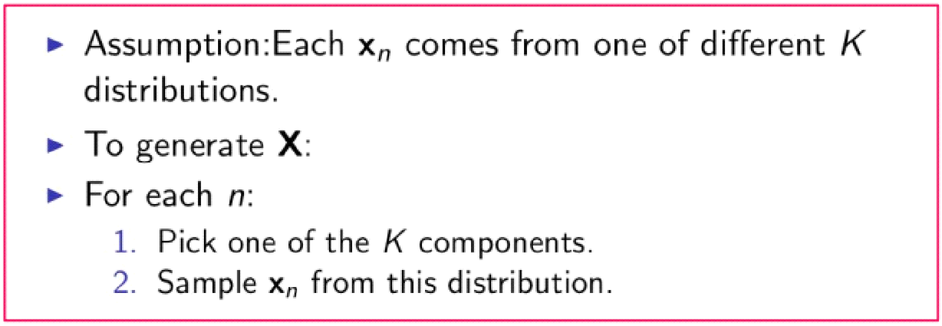
i.

- Makes simple K-means algorithm more flexible.
- But, have to now set additional parameters.
- Very sensitive to initial conditions – lots of local optima.

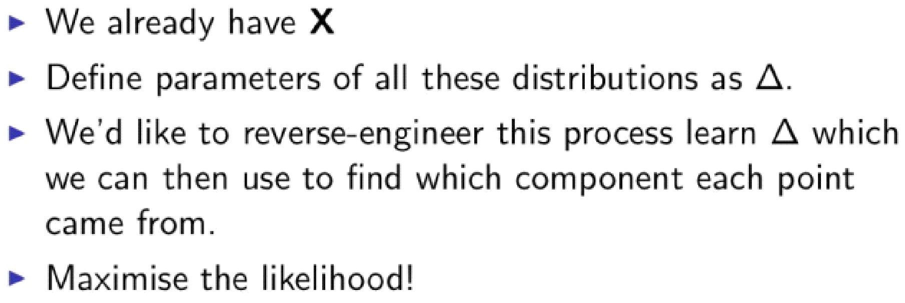
2. Part 2

a. Mixture models

i.



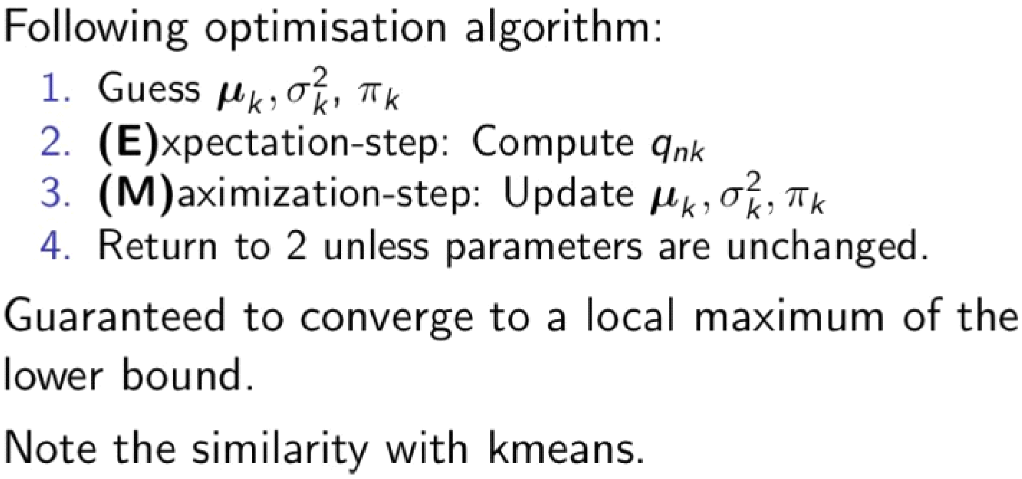
- Assumption:Each $\mathbf{x}_n$ comes from one of different K distributions.
- To generate $\mathbf{X}$:
- For each n:
  1. Pick one of the K components.
  2. Sample $\mathbf{x}_n$ from this distribution.



- We already have $\mathbf{X}$
- Define parameters of all these distributions as Δ.
- We'd like to reverse-engineer this process learn Δ which we can then use to find which component each point came from.
- Maximise the likelihood!

b. Expectation-Maximization (EM) algorithm

i.



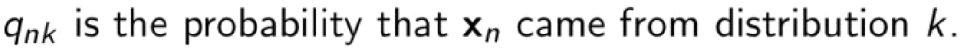
Following optimisation algorithm:

1. Guess $\boldsymbol{\mu}_k, \sigma_k^2, \pi_k$
2. **(E)**xpectation-step: Compute q_{nk}
3. **(M)**aximization-step: Update $\boldsymbol{\mu}_k, \sigma_k^2, \pi_k$
4. Return to 2 unless parameters are unchanged.

Guaranteed to converge to a local maximum of the lower bound.

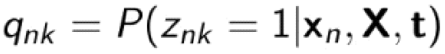
Note the similarity with kmeans.

ii. Qnk: points assigned to the cluster with the highest qnk value



q_{nk} is the probability that $\mathbf{x}_n$ came from distribution k.

1)

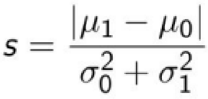


$$q_{nk} = P(z_{nk} = 1 | \mathbf{x}_n, \mathbf{X}, \mathbf{t})$$

iii. Likelihood always increases as σ decreases

Feature selection and projection

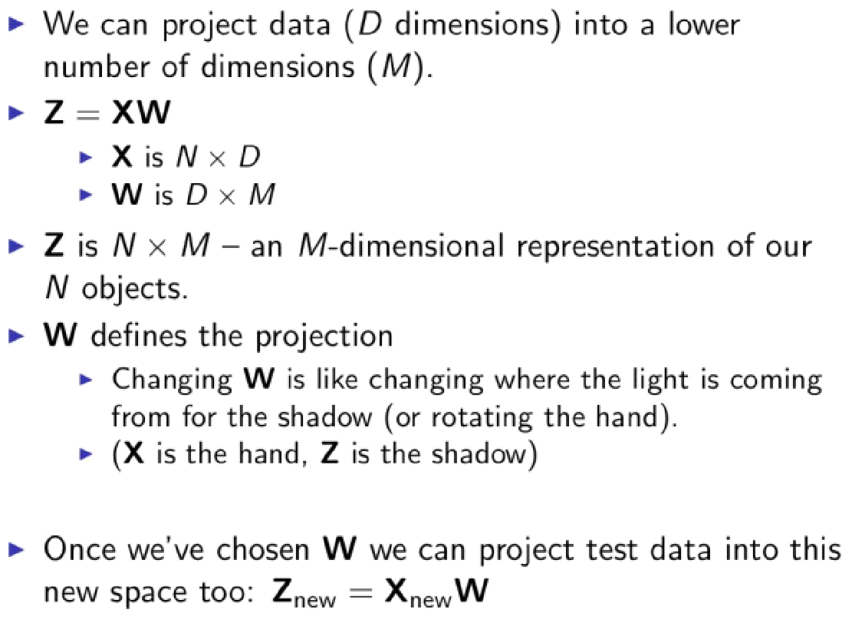
1. Motivation: too many features
2. Schemes: subset and combining
3. Subset
   a. Find the subset with the highest S[unknown]
   b.



$$s = \frac{|\mu_1 - \mu_0|}{\sigma_0^2 + \sigma_1^2}$$

   c. Must only compute S according to training data, otherwise biased
4. Combining/projection

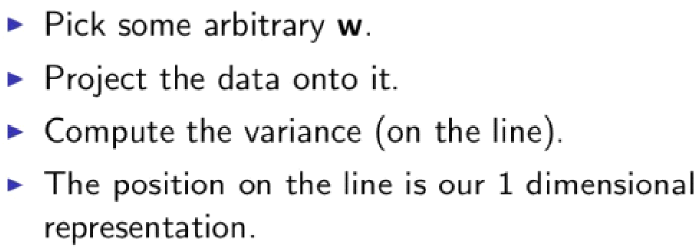
a.



- We can project data (D dimensions) into a lower number of dimensions (M).
- $\mathbf{Z} = \mathbf{XW}$
  - $\mathbf{X}$ is $N \times D$
  - $\mathbf{W}$ is $D \times M$
- $\mathbf{Z}$ is $N \times M$ – an M-dimensional representation of our N objects.
- $\mathbf{W}$ defines the projection
  - Changing $\mathbf{W}$ is like changing where the light is coming from for the shadow (or rotating the hand).
  - ($\mathbf{X}$ is the hand, $\mathbf{Z}$ is the shadow)
- Once we've chosen $\mathbf{W}$ we can project test data into this new space too: $\mathbf{Z}_{\text{new}} = \mathbf{X}_{\text{new}}\mathbf{W}$

b. PCA

i.



- Pick some arbitrary $\mathbf{w}$.
- Project the data onto it.
- Compute the variance (on the line).
- The position on the line is our 1 dimensional representation.

i. Search for best w

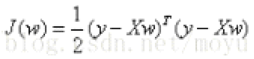
Regression

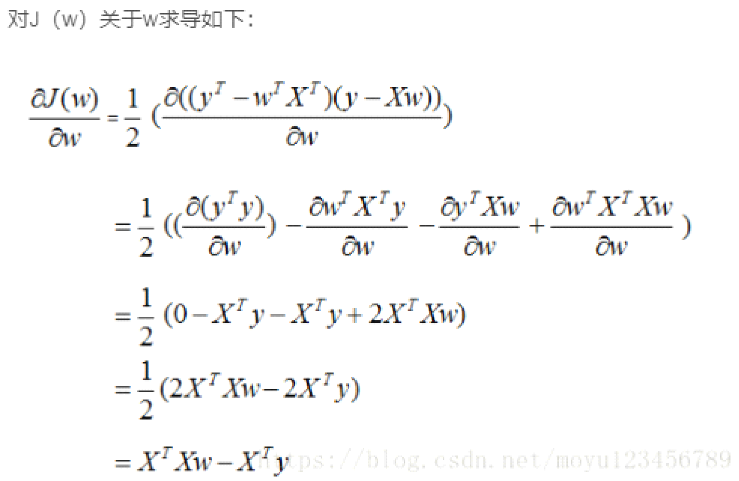
02 October 2021 23:24

Linear regression

1. https://blog.csdn.net/u014028027/article/details/72667733
2. https://blog.csdn.net/iqdutao/article/details/109402570?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163871676316780271539154%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163871676316780271539154&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-109402570.first_rank_v2_pc_rank_v29&utm_term=%E7%BA%BF%E6%80%A7%E5%9B%9E%E5%BD%92&spm=1018.2226.3001.4187
3. 主要分类
   a. 线性回归(Linear Regression)
   b. 逻辑回归（Logistic regressions）
   c. 多项式回归(Polynomial Regression)
   d. 逐步回归(Step Regression)
   e. 岭回归(Ridge Regression)
   f. 套索回归(Lasso Regression)

Loss

1. Least Square loss(最小二乘法)
   a. 它通过最小化误差的平方和寻找数据的最佳函数匹配。利用最小二乘法可以简便地求得未知的数据，并使得这些求得的数据与实际数据之间误差的平方和为最小
   b. 最小二乘法数学表示
   c. $$L(w_0, w_1) = \frac{1}{N}\sum_{n=1}^{N}(t_n - w_0 - w_1 x_n)^2$$
   d. 令L函数偏导为0，求出L的最小值
   e. 求偏导之后，w0和w1如下
   f. $$w_1 = \frac{\overline{xt} - \bar{x}\bar{t}}{\overline{xx} - \bar{x}^2}$$
      $$w_0 = \bar{t} - w_1\bar{x}$$
   g. 矩阵表示
      a. 最小二乘法矩阵表示
         i. 
            $$J(w) = \frac{1}{2}(y - Xw)^T(y - Xw)$$

      b. 求导
         i. 
            对J（w）关于w求导如下：
            $$\frac{\partial J(w)}{\partial w} = \frac{1}{2}(\frac{\partial((y^T - w^T X^T)(y - Xw))}{\partial w})$$
            $$= \frac{1}{2}((\frac{\partial(y^T y)}{\partial w}) - \frac{\partial w^T X^T y}{\partial w} - \frac{\partial y^T Xw}{\partial w} + \frac{\partial w^T X^T Xw}{\partial w})$$
            $$= \frac{1}{2}(0 - X^T y - X^T y + 2X^T Xw)$$
            $$= \frac{1}{2}(2X^T Xw - 2X^T y)$$
            $$= X^T Xw - X^T y$$

      c. 导数为零

    i. $X^T Xw - X^T y = 0 \Rightarrow X^T Xw = X^T y \Rightarrow w = (X^T X)^{-1} X^T y$

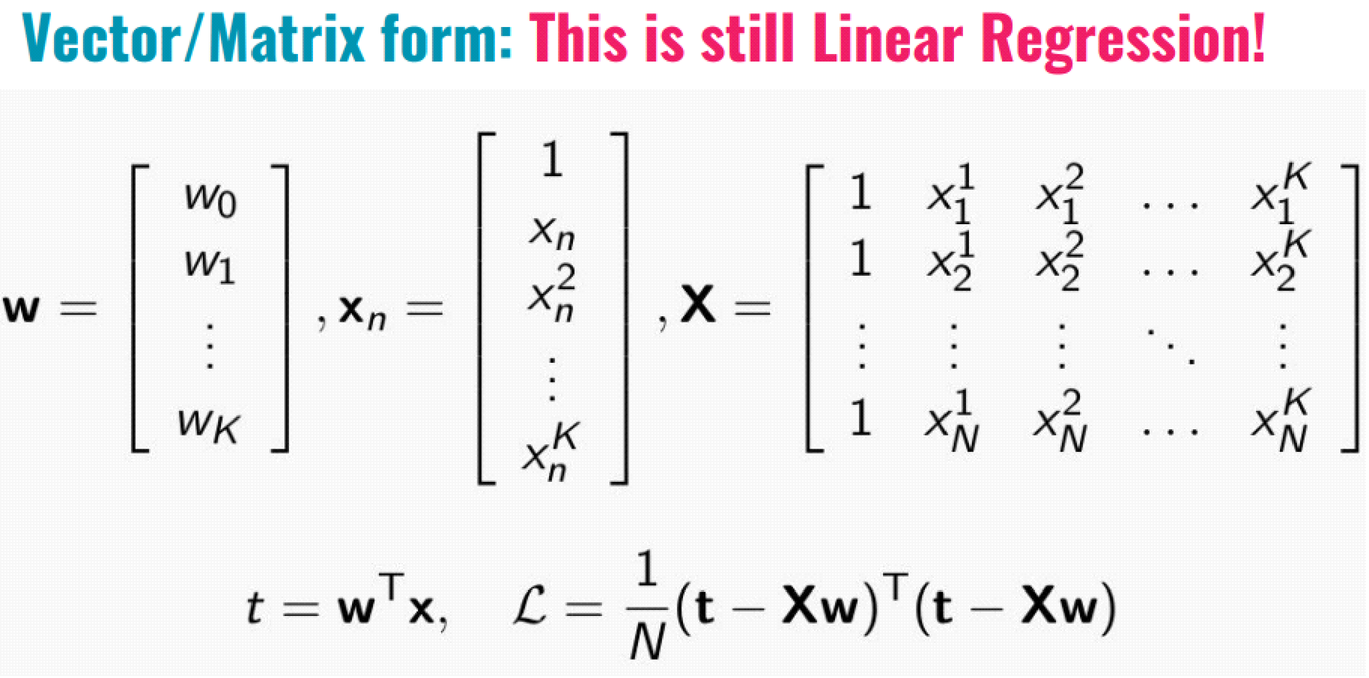
2. MMSE(最小均方差)与LSL(最小二乘)的差别
    a. https://blog.csdn.net/qq_43162058/article/details/106911540
    b. https://www.zhihu.com/question/27200164
    c.

	MMSE	LSL
作用	求已知模型的最佳参数	求未知模型的参数
前提条件	已知模型, 均值	未知模型, 需要用回归方法预测
底层原理	极大似然估计	贝叶斯决策

3. 用最大似然函数做代价函数的优点
    a. https://blog.csdn.net/weixin_34073886/article/details/112706581

Polynomial regression

1.

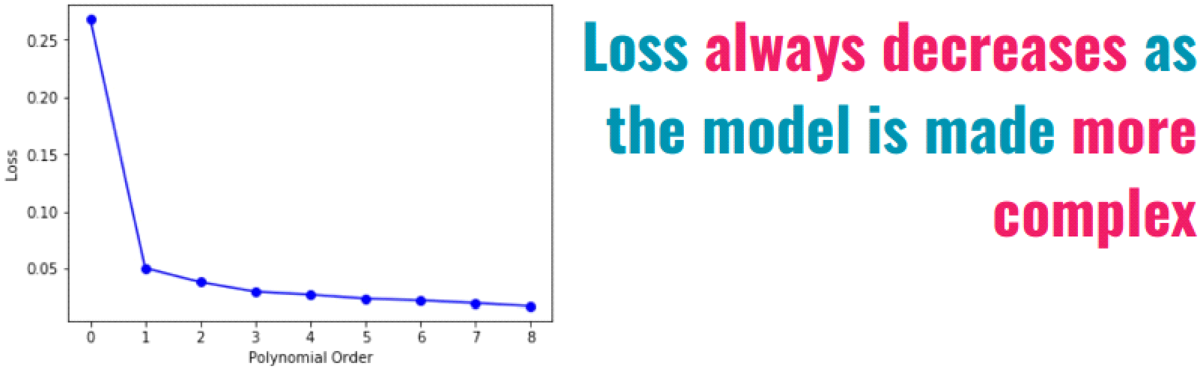


Vector/Matrix form: This is still Linear Regression!

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_K \end{bmatrix}, \mathbf{x}_n = \begin{bmatrix} 1 \\ x_n \\ x_n^2 \\ \vdots \\ x_n^K \end{bmatrix}, \mathbf{X} = \begin{bmatrix} 1 & x_1^1 & x_1^2 & \dots & x_1^K \\ 1 & x_2^1 & x_2^2 & \dots & x_2^K \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N^1 & x_N^2 & \dots & x_N^K \end{bmatrix}$$

$$t = \mathbf{w}^\mathsf{T}\mathbf{x}, \quad \mathcal{L} = \frac{1}{N}(\mathbf{t} - \mathbf{X}\mathbf{w})^\mathsf{T}(\mathbf{t} - \mathbf{X}\mathbf{w})$$

2.



Loss always decreases as the model is made more complex

cross-validation（cv）

1. 验证模型是准确程度的方法
    a. 将数据集平均分为n份，其中n-1份是训练集（training set），剩下的一份是验证集(validation set)，可以最后选择loss最小的那一种作为最终结果
    b. 缺点：相当于进行了n-1次训练，训练时间n倍。不适合数据集较大的模型
2. K-Fold方法

   a. 其中k代表数据集分成k份

Ridge & lasso

1. 详见lab2

Classification

19 October 2021 23:42

KNN

1. Knn实现
2. K-fold选择最优参数
3. pcolormesh画图

Logistics regression

1. 实现
2. 损失函数
3. 损失函数正则化
4. Sensitivity
5. Specificity
6. ROC&AUC

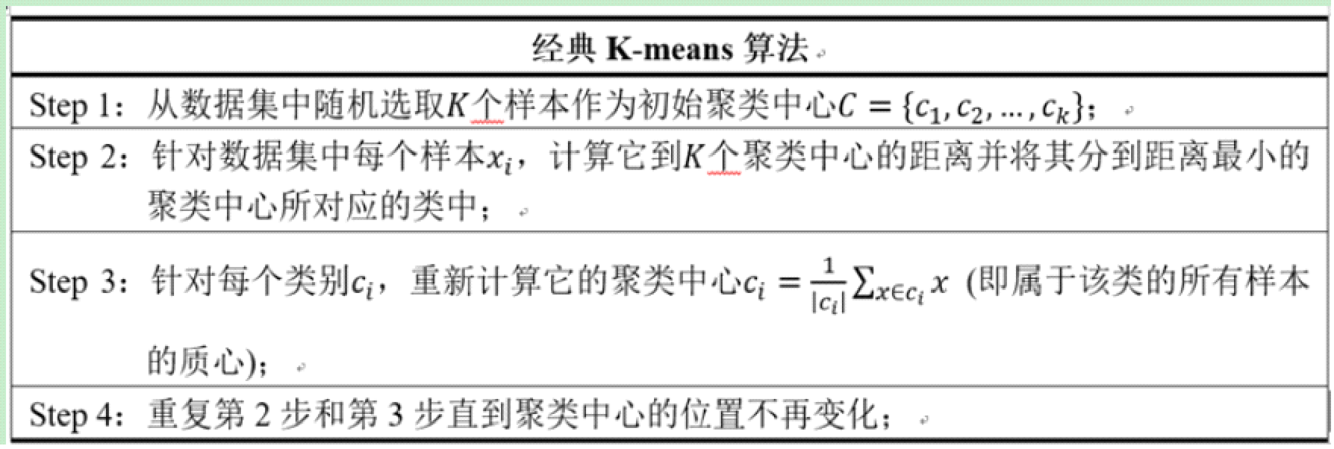
SVM

Clustering

05 December 2021 16:03

k-means

1. 预备阅读
    a. K-means聚类算法的三种改进(K-means++,ISODATA,Kernel K-means)介绍与对比
        i. https://www.cnblogs.com/yixuan-xu/p/6272208.html
    b. 详细解释
        i. https://scikit-learn.org/stable/modules/clustering.html
2. k-means具体流程
    a.

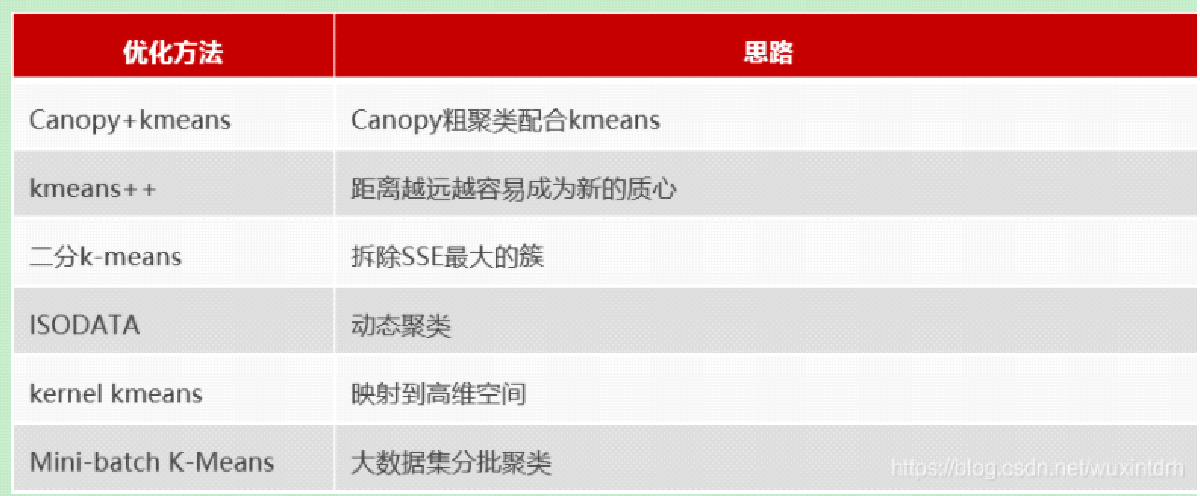


经典 K-means 算法
Step 1：从数据集中随机选取K个样本作为初始聚类中心$C=\{c_1,c_2,\dots,c_k\}$；
Step 2：针对数据集中每个样本x_i，计算它到K个聚类中心的距离并将其分到距离最小的聚类中心所对应的类中；
Step 3：针对每个类别c_i，重新计算它的聚类中心$c_i=\frac{1}{\lvert c_i\rvert}\sum_{x\in c_i}x$ (即属于该类的所有样本的质心)；
Step 4：重复第 2 步和第 3 步直到聚类中心的位置不再变化；

3. 介绍
    a. https://zhuanlan.zhihu.com/p/78798251
    b. K-means 是我们最常用的基于欧式距离的聚类算法，其认为两个目标的距离越近，相似度越大
    c. 数据接近高斯分布的时候效果最好
    d. K值选取很重要, 需要手动选取, 有一些选取方法

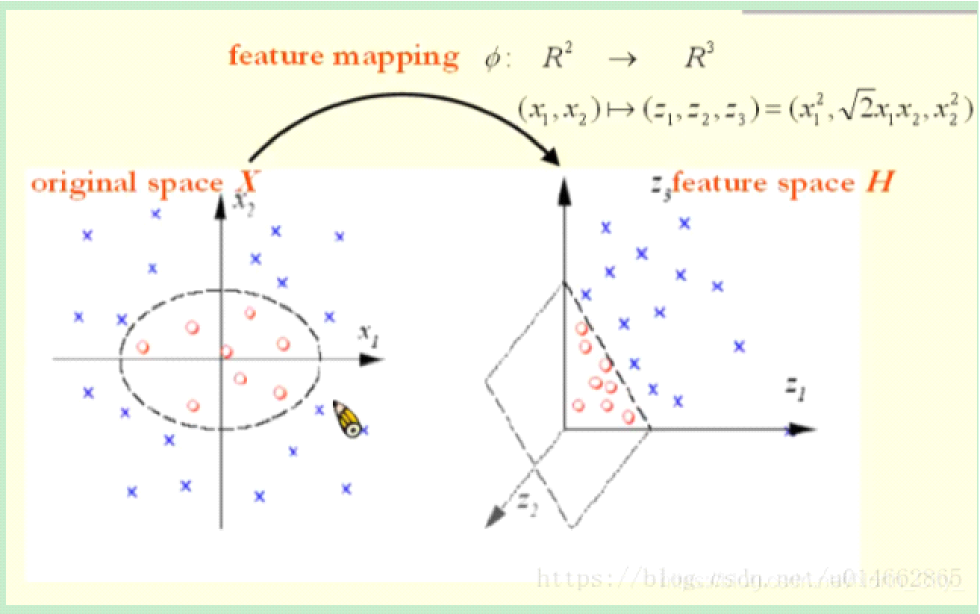
Kernel k-means

1. 预备阅读
    a. k-means的优化
        i. https://chbxw.blog.csdn.net/article/details/105425314
    b.



优化方法	思路
Canopy+kmeans	Canopy粗聚类配合kmeans
kmeans++	距离越远越容易成为新的质心
二分k-means	拆除SSE最大的簇
ISODATA	动态聚类
kernel kmeans	映射到高维空间
Mini-batch K-Means	大数据集分批聚类

2. 介绍
    a. Kernel k-means是对k-means的一种映射, 将其从不好线性分割的二维平面映射到容易线性分割的三维空间中, 以获得更好的分割效果

b. 
feature mapping $\phi: R^2 \rightarrow R^3$
$(x_1, x_2) \mapsto (z_1, z_2, z_3) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$
original space X
x_2
x_1
z_3 feature space H
z_1
z_2
https://blog.csdn.net/u014662865

3. Kernel
    a. 核函数
        i. https://www.cnblogs.com/infinite-h/p/10723853.html

Mixture models

1. 预备阅读
    a. 贝叶斯分类器, 贝叶斯决策论
        i. https://zhuanlan.zhihu.com/p/42991859
    b. 整体介绍
        i. https://blog.csdn.net/lin_limin/article/details/81048411
2. Gaussian mixture model, GMM
    a. 对于一个分布单一高斯模型拟合的效果并不好, 可能模型是由几个高斯分布线性组合来拟合分布
    b. 为了求线性组合的参数, 可以用极大似然估计和EM估计两种方法
    c. GMM的核心就是确认其参数(μ, σ, π)
    d. μ与σ为高斯分布的参数, π为不同高斯分布线性组合时候的系数
3. Maximum Likehood Estimate, MLE(极大似然估计)
    a. 适用情况:
        i. 已知数据分布模型(即所有数据源于同一个分布)和采样数据, 求数据分布模型的参数
        ii. 例: 已知都来源于(μ1, σ1)确定的高斯分布和采样数据, 求每一个高斯模型对应的参数π
    b. 在对数似然函数中没办法通过求导的方式求解, 因此提出了EM
4. EM
    a. 适用情况
        i. 已知各个类具体的分布模型和采样数据, 求采样数据来源于哪一个模型以及各个模型的参数
        ii. 例: 已知都来源于不同高斯分布, 求线性组合每一项对应的pi和(μn, σn)
        iii. https://zhuanlan.zhihu.com/p/78311644
        iv. https://zhuanlan.zhihu.com/p/61127050
        v. https://zhuanlan.zhihu.com/p/383732110
    b. 步骤
        i. E-Step 主要通过观察数据和现有模型来估计参数，然后用这个估计的参数值来计算似然函数的期望值
        ii. M-Step 是寻找似然函数最大化时对应的参数。由于算法会保证在每次迭代之后似然函数都会增加，所以函数最终会收敛。

Projection

05 December 2021 20:13

Projection

1. Sometimes too many features, we have to reduce the number of features
2. 2 general schemes: subset and combination

Subset

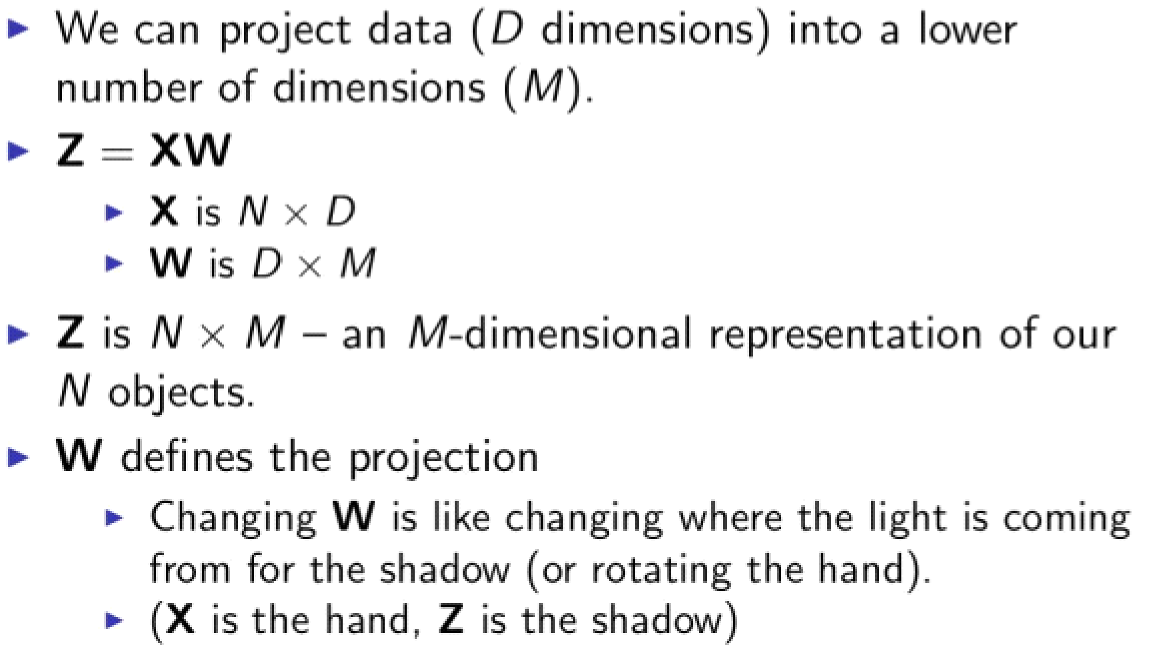
1. 预备阅读
    a. 特征子集选择: 最优子集选择, 向前选择, 向后选择
        i. https://blog.csdn.net/jesseyule/article/details/95157045?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163870786616780255288897%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=163870786616780255288897&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_v2~hot_rank-5-95157045.pc_v2_rank_blog_default&utm_term=%E5%AD%90%E9%9B%86&spm=1018.2226.3001.4450
        ii. https://blog.csdn.net/u012303532/article/details/42550499?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163870659116780255288382%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=163870659116780255288382&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_v2~hot_rank-3-42550499.pc_v2_rank_blog_default&utm_term=subset&spm=1018.2226.3001.4450
2. S值计算
    a. 选取一个feature, 求出μ和σ, 分成两类
    b.

$$s = \frac{|\mu_1 - \mu_0|}{\sigma_0^2 + \sigma_1^2}$$

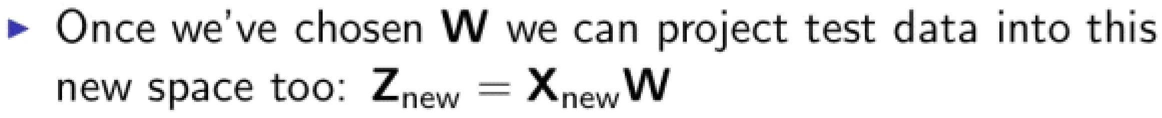
    c. s越大效果越好
    d. 只能用训练集求s值, 测试集不能使用

Combination

1. Make new features by combination
2. Clustering
    a. See clustering
3. Projection

a. 

- We can project data (D dimensions) into a lower number of dimensions (M).
- $\mathbf{Z} = \mathbf{XW}$
  - $\mathbf{X}$ is $N \times D$
  - $\mathbf{W}$ is $D \times M$
- $\mathbf{Z}$ is $N \times M$ – an M-dimensional representation of our N objects.
- $\mathbf{W}$ defines the projection
  - Changing $\mathbf{W}$ is like changing where the light is coming from for the shadow (or rotating the hand).
  - ($\mathbf{X}$ is the hand, $\mathbf{Z}$ is the shadow)



- Once we've chosen $\mathbf{W}$ we can project test data into this new space too: $\mathbf{Z}_{\text{new}} = \mathbf{X}_{\text{new}}\mathbf{W}$

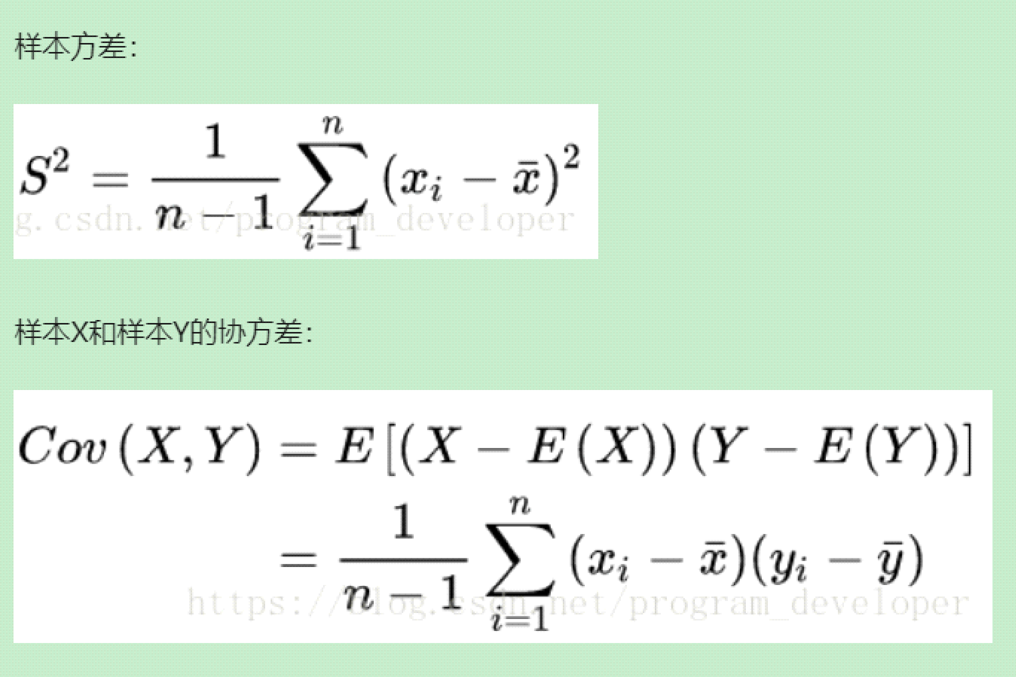
4. Choosing W

   a. 方差与协方差

      i. 协方差为正时，说明X和Y是正相关关系；协方差为负时，说明X和Y是负相关关系；协方差为0时，说明X和Y是相互独立。

      ii. 方差是在一维情况下的协方差

      iii. 方差和协方差在采用无偏估计的时候除数为(n-1)

      iv. 

样本方差：

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

样本X和样本Y的协方差：

$$\begin{aligned} Cov(X,Y) &= E\left[(X - E(X))(Y - E(Y))\right] \\ &= \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) \end{aligned}$$

      v. https://blog.csdn.net/program_developer/article/details/80632779?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163871073716780271913646%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=163871073716780271913646&biz_id=0

&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_v2~hot_rank-2-80632779.pc_v2_rank_blog_default&utm_term=pca&spm=1018.2226.3001.4450

b. PCA

   i. PCA(Principal Component Analysis)，即主成分分析方法，是一种使用最广泛的数据降维算法。PCA的主要思想是将n维特征映射到k维上，这k维是全新的正交特征也被称为主成分，是在原有n维特征的基础上重新构造出来的k维特征。PCA的工作就是从原始的空间中顺序地找一组相互正交的坐标轴，新的坐标轴的选择与数据本身是密切相关的。其中，第一个新坐标轴选择是原始数据中方差最大的方向，第二个新坐标轴选取是与第一个坐标轴正交的平面中使得方差最大的，第三个轴是与第1,2个轴正交的平面中方差最大的。依次类推，可以得到n个这样的坐标轴。通过这种方式获得的新的坐标轴，我们发现，大部分方差都包含在前面k个坐标轴中，后面的坐标轴所含的方差几乎为0。于是，我们可以忽略余下的坐标轴，只保留前面k个含有绝大部分方差的坐标轴。事实上，这相当于只保留包含绝大部分方差的维度特征，而忽略包含方差几乎为0的特征维度，实现对数据特征的降维处理。

   ii. sklearn中pca用法

      1) https://blog.csdn.net/qq_20135597/article/details/95247381

      2) 其中最主要的参数n为要保留的参数个数

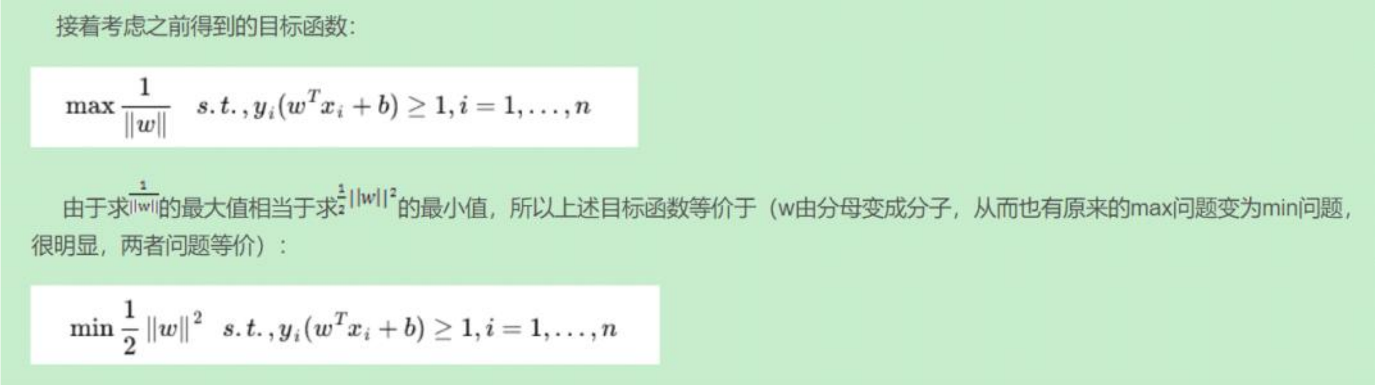
   iii. 具体算法 https://blog.csdn.net/lanyuelvyun/article/details/82384179?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163871073716780271913646%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=163871073716780271913646&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_v2~hot_rank-3-82384179.pc_v2_rank_blog_default&utm_term=pca&spm=1018.2226.3001.4450

   iv. https://blog.csdn.net/program_developer/article/details/80632779?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163871073716780271913646%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=163871073716780271913646&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_v2~hot_rank-2-80632779.pc_v2_rank_blog_default&utm_term=pca&spm=1018.2226.3001.4450

SVM & RBF

05 December 2021 22:01

SVM

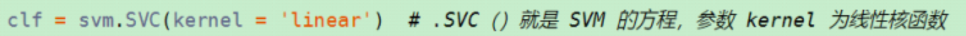
1. 预备阅读
   a. https://blog.csdn.net/v_JULY_v/article/details/7624837?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163870844216780271511516%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163870844216780271511516&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~top_positive~default-2-7624837.pc_v2_rank_blog_default&utm_term=svm&spm=1018.2226.3001.4450
2. 一种线性分类器, 其基本模型定义为特征空间上的间隔最大的线性分类器，其学习策略便是间隔最大化，最终可转化为一个凸二次规划问题的求解。
   a. 求出平面上点到直线的距离, 距离最大的时候分类效果最好
   b. 目标函数
      i. 

      接着考虑之前得到的目标函数:

      $$\max \frac{1}{\|w\|} \quad s.t., y_i(w^T x_i + b) \geq 1, i = 1, \ldots, n$$

      由于求$\frac{1}{\|w\|}$的最大值相当于求$\frac{1}{2}\|w\|^2$的最小值，所以上述目标函数等价于（w由分母变成分子，从而也有原来的max问题变为min问题，很明显，两者问题等价）：

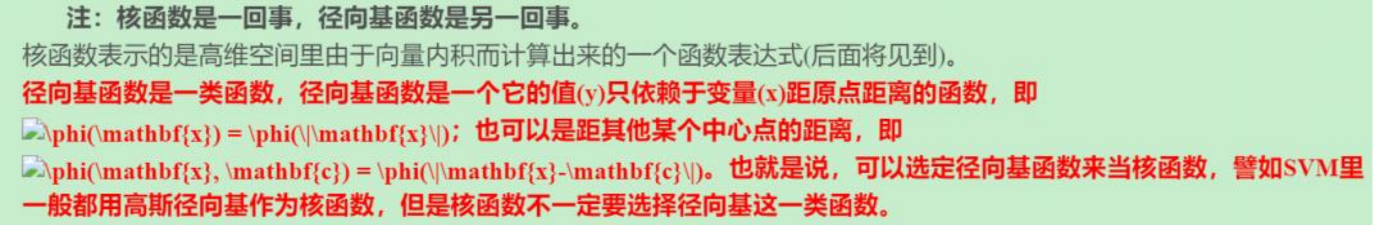
      $$\min \frac{1}{2}\|w\|^2 \quad s.t., y_i(w^T x_i + b) \geq 1, i = 1, \ldots, n$$

      ii. 其中||w||为范数, 可以理解为距离
   c. 由于目标函数是二次且约束条件是线性的, 故这是一个凸二次规划问题, 可以求出在一定约束条件下目标的最优解
   d. 可以利用拉格朗日对偶性变换到等价的求对偶问题的最优解, 而在对偶变换后可以引入核函数, 这样就可以推广到非线性分类中(如rbf kernel)
3. sklearn.svm用法
   a. 只有一个参数kernel, 可选多种核函数
   b. 

```
clf = svm.SVC(kernel = 'linear')  # .SVC () 就是 SVM 的方程, 参数 kernel 为线性核函数
```

4. svm内核的优缺点比较
   a. https://blog.csdn.net/qq_29462849/article/details/89516133
   b. https://www.cnblogs.com/lsm-boke/p/11761534.html
   c. https://blog.csdn.net/gracejpw/article/details/103023352

RBF

1. 预备阅读
   a. https://blog.csdn.net/qq_15295565/article/details/80888607?ops_request_misc=&request_id=&biz_id=102&utm_term=rbf%E5%87%BD%E6%95%B0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~sobaiduweb~default-1-80888607.pc_v2_rank_blog_default&spm=1018.2226.3001.4450
   b. https://blog.csdn.net/u013630349/article/details/48162589?spm=1001.2101.3001.6650.4&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-4.essearch_pc_relevant&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-4.essearch_pc_relevant
   c. https://blog.csdn.net/red_stone1/article/details/101302684
2. RBF函数与RBF kernel的区别
   a. 

   注：核函数是一回事，径向基函数是另一回事。

   核函数表示的是高维空间里由于向量内积而计算出来的一个函数表达式(后面将见到)。

   径向基函数是一类函数，径向基函数是一个它的值(y)只依赖于变量(x)距原点距离的函数，即 \phi(\mathbf{x}) = \phi(\|\mathbf{x}\|)；也可以是距其他某个中心点的距离，即 \phi(\mathbf{x}, \mathbf{c}) = \phi(\|\mathbf{x}-\mathbf{c}\|)。也就是说，可以选定径向基函数来当核函数，譬如SVM里一般都用高斯径向基作为核函数，但是核函数不一定要选择径向基这一类函数。
3. RBF原理及实现
   a. https://blog.csdn.net/weixin_42398658/article/details/83215916?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163871515016780271928587%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163871515016780271928587&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-2-83215916.first_rank_v2_pc_rank_v29&utm_term=rbf+kernel&spm=1018.2226.3001.4187

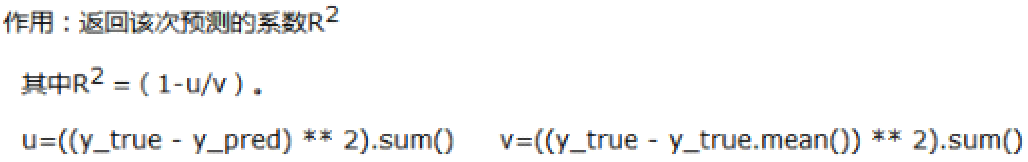
Lab1

02 October 2021 17:40

Outline

1. Load txt file and plot
2. Fit a simple linear regression
3. Make a prediction

Linear regression

1. https://blog.csdn.net/weixin_39175124/article/details/79465558
2. reg = sklearn.linear_model.LinearRegression(fit_intercept=True, normalize=False, copy_X=True, n_jobs=1)
3. LinearRegression.fit()函数必须输入2维数组，1维数组会报错，可以通过x = x[:,None]给数组增加维度
4. 先定义一个回归对象reg，并把回归的值赋给他
5. reg有两个可用属性，coef_代表系数，intercept_代表偏置
6. LinearRegression有四个方法：fit（x,y），predict(x), score(x,y), get_params()
    a. fit（）用来训练模型
    b. predict（）用来预测训练好的模型在一点的值，注意输入必须是二维的
    c. score（x，y）用来评估，最好的得分是1
        i. 

作用：返回该次预测的系数R^2

其中R^2 = (1-u/v) .

u=((y_true - y_pred) ** 2).sum()   v=((y_true - y_true.mean()) ** 2).sum()

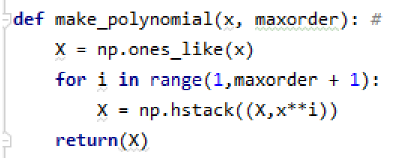
Lab2

14 October 2021 12:41

Outline

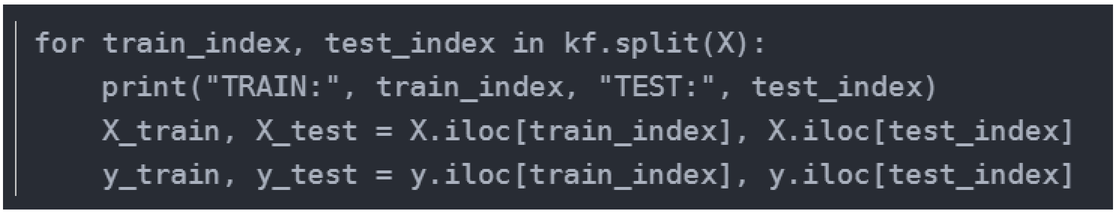
1. Load data and plot
2. Polynomial regression
   a. Rescale data
   b. Construct the design matrix with polynomials
3. Choose the best order
   a. Cross-validation

Polynomial regression

1. references
   a. https://blog.csdn.net/qq_33182424/article/details/83420878
   b. https://blog.csdn.net/qq_36523839/article/details/82924804
2. 构建多项式矩阵（design matrix of polynomial）
   a. 

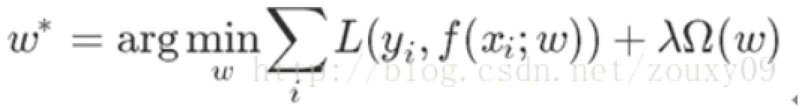
```
def make_polynomial(x, maxorder): #
    X = np.ones_like(x)
    for i in range(1,maxorder + 1):
        X = np.hstack((X,x**i))
    return(X)
```

Cross-validation(K-Fold)

1. KFold()在sklearn中属于model_slection模块
2. Cv = KFold(n_splits=’ warn’ , shuffle=False, random_state=None)
   a. n_splits 表示划分为几块（至少是2）
   b. shuffle 表示是否打乱划分，默认False，即不打乱
   c. random_state 表示是否固定随机起点，Used when shuffle == True
3. 方法
   a. Cv.get_n_splits()：返回分成的份数
   b. Cv.split()：用于获得训练集和测试集
      i. 

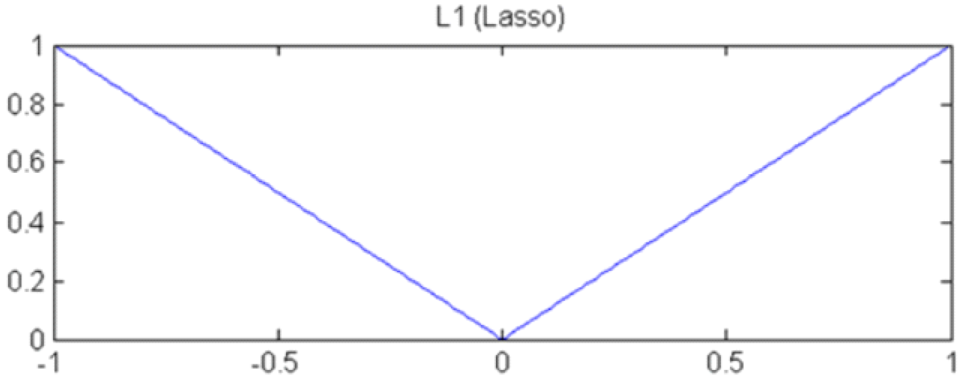
```
for train_index, test_index in kf.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
```

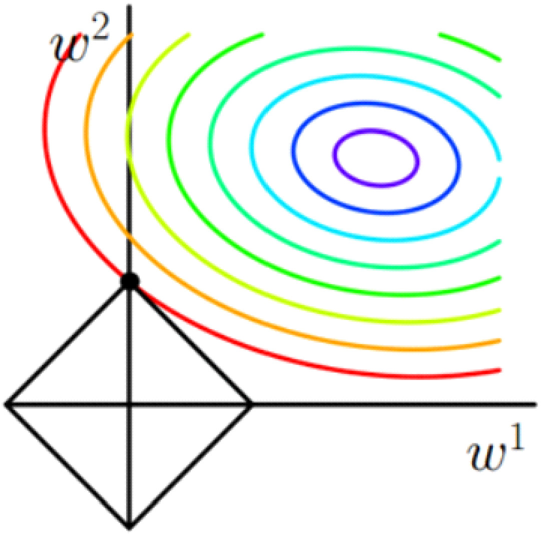
L0,L1,L2范数

1. reference
   a. https://blog.csdn.net/zouxy09/article/details/24971995
   b. https://blog.csdn.net/zouxy09/article/details/24972869?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163465467616780357250797%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=163465467616780357250797&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_v2~rank_v29-2-24972869.pc_v2_rank_blog_default&utm_term=%E8%8C%83%E6%95%B0&spm=1018.2226.3001.4450
2. 监督机器学习就是在规则化参数的同时最小化误差。最小化误差是为了让我们的模型拟合我们的训练数据，而规则化参数是防止我们的模型过分拟合我们的训练数据。在数学上表示为最小化以下函数
   a. 

$$w^* = \arg\min_w \sum_i L(y_i, f(x_i; w)) + \lambda\Omega(w)$$

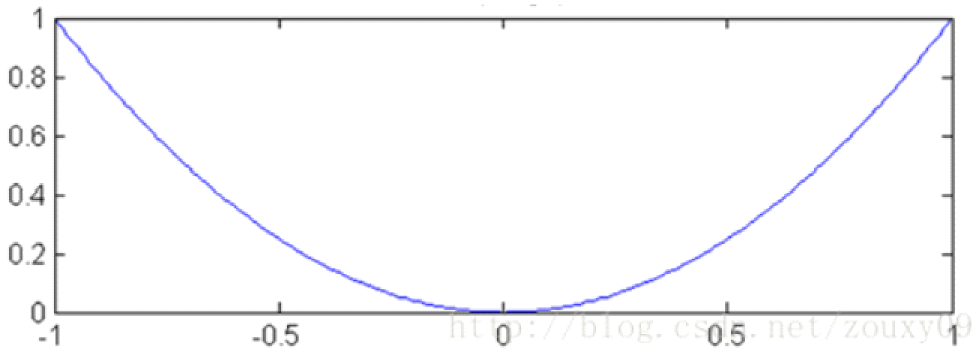
   b. 其中L为预测值和真实值之间的误差，Ω为模型的测试误差
   c. L的不同算法区分了大部分模型，square loss为最小二乘，hinge loss为SVM，exp loss为Boosting，log loss为Logistics regression
   d. Ω就是常说的范数

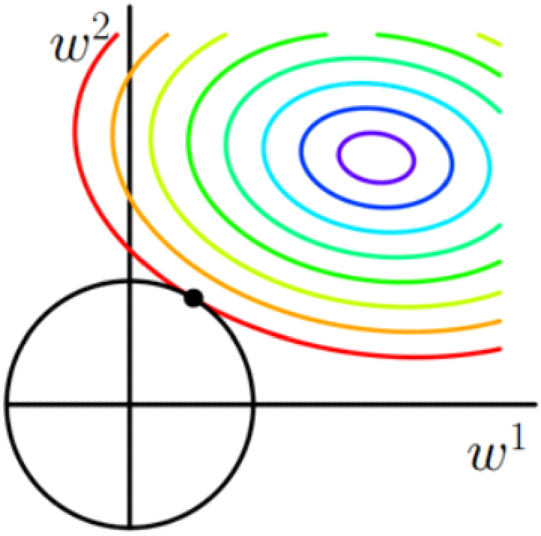
3. L0范数
    a. 可以实现稀疏，但优化求解麻烦，所以多用L1
4. L1（lasso）
    a. 让绝对值最小
    b. $$\hat{\mathbf{w}}_{lasso} = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{N}(\mathbf{t}-\mathbf{X}\mathbf{w})^T(\mathbf{t}-\mathbf{X}\mathbf{w}) + \alpha \sum_d |w_d|$$
    c. 

    d. 

(a) ℓ_1-ball meets quadratic function. ℓ_1-ball has corners. It's very likely that the meet-point is at one of the corners.

    e. 圆圈与菱形交界点为最优, 交点在y轴, 所以会得到很多0, 具有稀疏性, 特征前面的系数为0, 代表不需要这个特征, 依赖特征较少
5. L2（ridge）
    a. 让平方最小
    b. $$\hat{\mathbf{w}}_{ridge} = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{N}(\mathbf{t}-\mathbf{X}\mathbf{w})^T(\mathbf{t}-\mathbf{X}\mathbf{w}) + \alpha \mathbf{w}^T\mathbf{w}$$

c. 

d. 

(b) ℓ_2-ball meets quadratic function. ℓ_2-ball has no corner. It is very unlikely that the meet-point is on any of axes.”

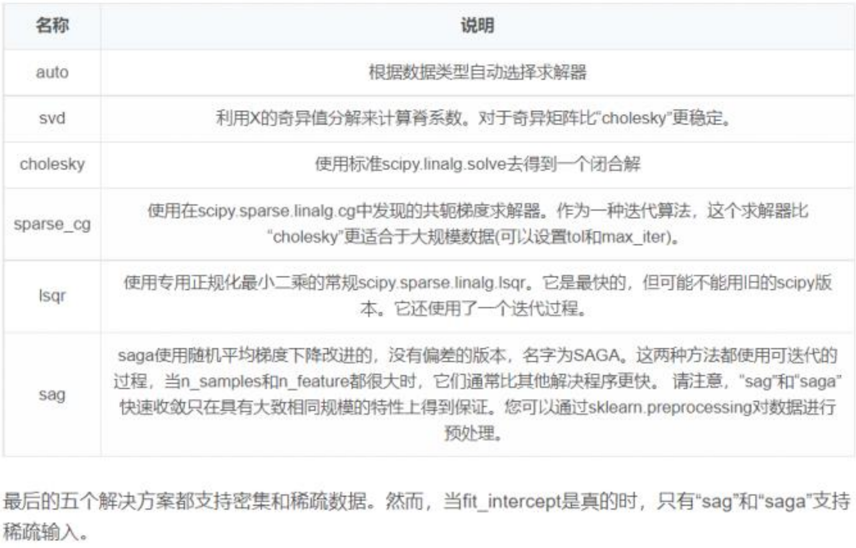
e. L2不具有稀疏性

Ridge

1. Reference
   a. https://blog.csdn.net/weixin_43374551/article/details/83688913?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163464975416780261965231%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163464975416780261965231&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-83688913.first_rank_v2_pc_rank_v29&utm_term=lasso%E5%9B%9E%E5%BD%92&spm=1018.2226.3001.4187
2. ridge（岭回归）原理
   a. $w = (X^T X + \lambda I)^{-1} X^T y$
   b. 损失函数中加入λI项，其中λ为正则化力度，I为单位矩阵
   c. 为了达到样本数量n和特征数量p之间的平衡，需要实现方差（模型之间的差异）和偏差（模型预测值和实测数据之间的差异）的折中，避免过拟合或欠拟合
3. Ridge的语法
   a. from sklearn.linear_model import Ridge
   b. Ridge = Ridge(alpha=1.0, fit_intercept=True, normalize=False, copy_X=True, max_iter=None, tol=1e-3, solver=”auto”, random_state=None)
   c. alpha：正则化力度，正浮点数
   d. fit_intercept：是否计算结局，取决于数据是否已经中心化
   e. normalize： 默认为F，当fit_intercept为F时自动忽略。T时对数据做归一化处理
   f. copy_x：default T
   g. tol： 处理精度
   h. solver：求解程序

i.

ii.



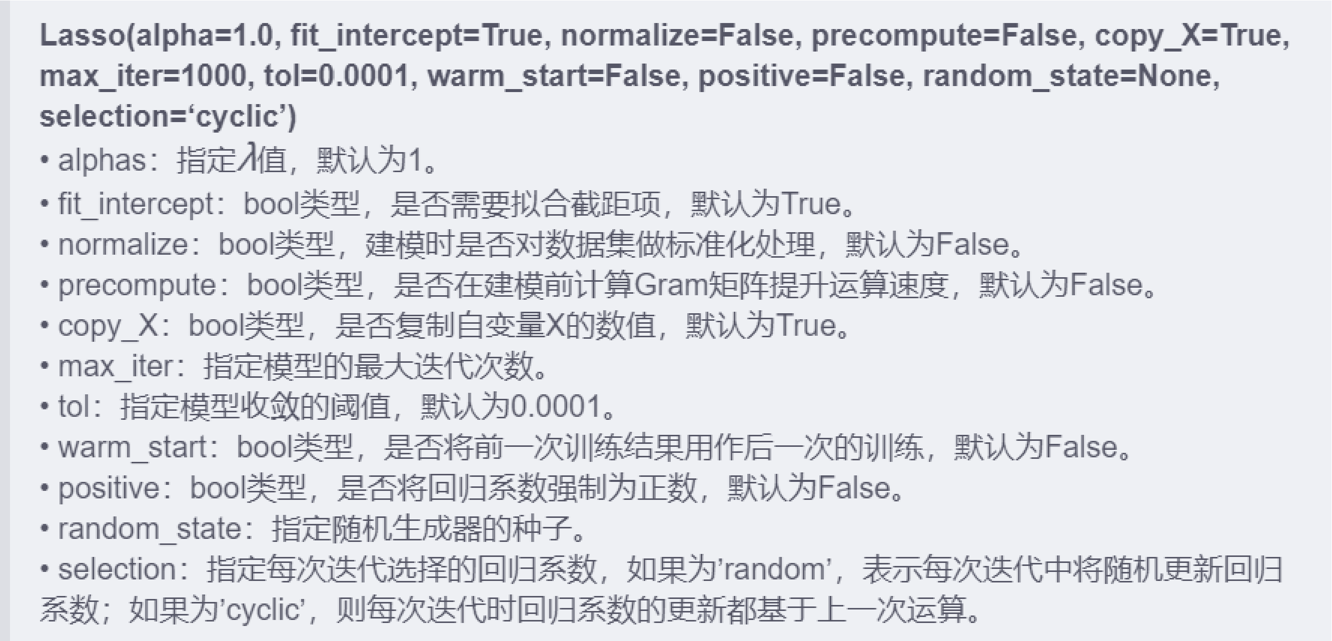
名称	说明
auto	根据数据类型自动选择求解器
svd	利用X的奇异值分解来计算脊系数。对于奇异矩阵比“cholesky”更稳定。
cholesky	使用标准scipy.linalg.solve去得到一个闭合解
sparse_cg	使用在scipy.sparse.linalg.cg中发现的共轭梯度求解器。作为一种迭代算法，这个求解器比“cholesky”更适合于大规模数据(可以设置tol和max_iter)。
lsqr	使用专用正规化最小二乘的常规scipy.sparse.linalg.lsqr。它是最快的，但可能不能用旧的scipy版本。它还使用了一个迭代过程。
sag	saga使用随机平均梯度下降改进的，没有偏差的版本，名字为SAGA。这两种方法都使用可迭代的过程，当n_samples和n_feature都很大时，它们通常比其他解决程序更快。 请注意，“sag”和“saga”快速收敛只在具有大致相同规模的特性上得到保证。您可以通过sklearn.preprocessing对数据进行预处理。

最后的五个解决方案都支持密集和稀疏数据。然而，当fit_intercept是真的时，只有“sag”和“saga”支持稀疏输入。

4. Ridge 的属性
    a. Ridge.coef_: 权重向量
    b. Ridge.intercept_ : 独立项
    c. Ridge.n_iter_: 迭代次数

Lasso

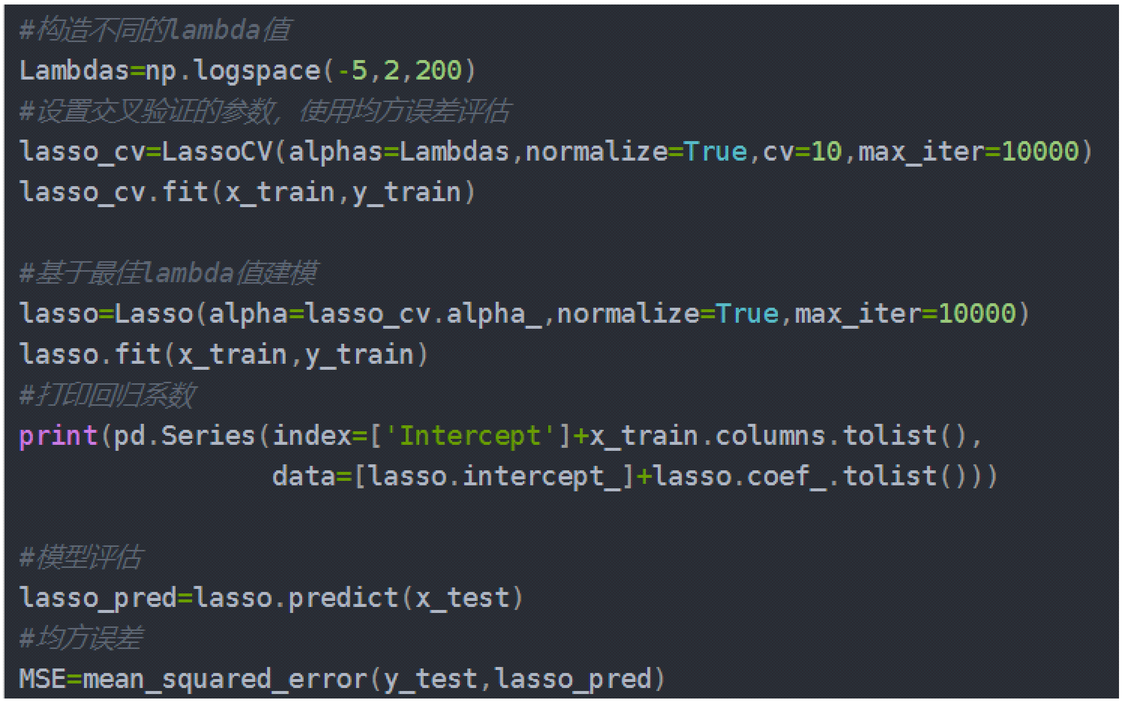
1. Reference
    a. https://blog.csdn.net/weixin_43374551/article/details/83688913?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163464975416780261965231%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163464975416780261965231&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-83688913.first_rank_v2_pc_rank_v29&utm_term=lasso%E5%9B%9E%E5%BD%92&spm=1018.2226.3001.4187
2. 语法
    a.



Lasso(alpha=1.0, fit_intercept=True, normalize=False, precompute=False, copy_X=True, max_iter=1000, tol=0.0001, warm_start=False, positive=False, random_state=None, selection=‘cyclic’)

- alphas：指定λ值，默认为1。
- fit_intercept：bool类型，是否需要拟合截距项，默认为True。
- normalize：bool类型，建模时是否对数据集做标准化处理，默认为False。
- precompute：bool类型，是否在建模前计算Gram矩阵提升运算速度，默认为False。
- copy_X：bool类型，是否复制自变量X的数值，默认为True。
- max_iter：指定模型的最大迭代次数。
- tol：指定模型收敛的阈值，默认为0.0001。
- warm_start：bool类型，是否将前一次训练结果用作后一次的训练，默认为False。
- positive：bool类型，是否将回归系数强制为正数，默认为False。
- random_state：指定随机生成器的种子。
- selection：指定每次迭代选择的回归系数，如果为’random’，表示每次迭代中将随机更新回归系数；如果为’cyclic’，则每次迭代时回归系数的更新都基于上一次运算。

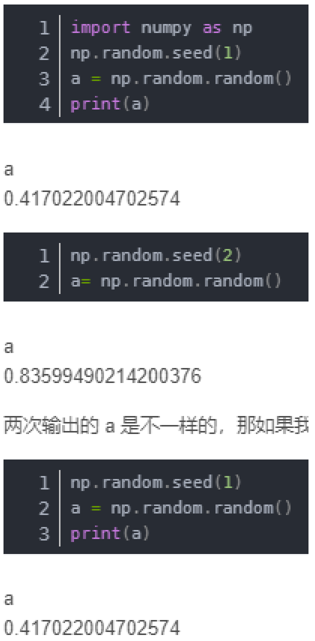
3. 用法

a. 

```
#构造不同的lambda值
Lambdas=np.logspace(-5,2,200)
#设置交叉验证的参数，使用均方误差评估
lasso_cv=LassoCV(alphas=Lambdas,normalize=True,cv=10,max_iter=10000)
lasso_cv.fit(x_train,y_train)

#基于最佳lambda值建模
lasso=Lasso(alpha=lasso_cv.alpha_,normalize=True,max_iter=10000)
lasso.fit(x_train,y_train)
#打印回归系数
print(pd.Series(index=['Intercept']+x_train.columns.tolist(),
                data=[lasso.intercept_]+lasso.coef_.tolist()))

#模型评估
lasso_pred=lasso.predict(x_test)
#均方误差
MSE=mean_squared_error(y_test,lasso_pred)
```

其他

1. np.random.seed(1)
   a. https://blog.csdn.net/econe_wei/article/details/90384226

   b. 生成固定的随机值，每次括号内值相同时生成的随机值相同
   c. 可以理解为种子堆，括号中为5则表示第五堆种子，从每堆种子中取出的顺序是固定的
   d. 每次用之前需要先声明堆序号
   e. 

```
import numpy as np
np.random.seed(1)
a = np.random.random()
print(a)
```

```
a
0.417022004702574
```

```
np.random.seed(2)
a= np.random.random()
```

```
a
0.83599490214200376
```

两次输出的 a 是不一样的，那如果我

```
np.random.seed(1)
a = np.random.random()
print(a)
```

```
a
0.417022004702574
```

2. GridSearchCV的语法
   a. https://www.cnblogs.com/dalege/p/14175192.html
   b. https://blog.csdn.net/weixin_41988628/article/details/83098130?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163422450516780262528361%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163422450516780262528361&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~top_positive~default-1-83098130.pc_v2_rank_blog_default&utm_term=GridSearchCV&spm=1018.2226.3001.4450
   c. 一种贪心算法，拿当前对模型影响最大的参数调优，直到最优化；再拿下一个影响最大的参数调优，如此下去，直到所有的参数调整完毕。这个方法的缺点就是可能会调到局部最优而

不是全局最优，但是省时间省力

Lab3

20 October 2021 00:15

Knn

1. Reference
   a. https://blog.csdn.net/qq_40195360/article/details/86714337
2. Workflow
   a. Import
   n_neighbors =

```
clf=KNeighborsClassifier(n_neighbors)
clf.fit(X,t)
Z=clf.predict(np.c_[xx.ravel(),yy.ravel()])
Z=Z.reshape(xx.shape)
```

3. weight参数
   a. 可选择"uniform"，“distance” 或自定义权重
   b. 默认uniform，即所有权重都一样，适合分布扎堆
   c. distance表示权重和距离成反比，适合样本分布较乱

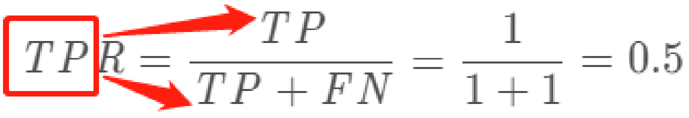
Logistic

1. Reference
   a. https://blog.csdn.net/qq_30031221/article/details/109753202?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163471337516780269893471%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163471337516780269893471&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~top_click~default-1-109753202.pc_v2_rank_blog_default&utm_term=logisticregression+%E5%8F%82%E6%95%B0&spm=1018.2226.3001.4450
2. Workflow
   a. from sklearn.linear_model import LogisticRegression
   clf = LogisticRegression()
   clf.fit(X,t)
   Z = clf.predict_proba(np.c_[xx.ravel(), yy.ravel()])[:, 1]
3. 语法
   a. https://blog.csdn.net/qq_30031221/article/details/109753202?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163471337516780269893471%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163471337516780269893471&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~top_click~default-1-109753202.pc_v2_rank_blog_default&utm_term=logisticregression+%E5%8F%82%E6%95%B0&spm=1018.2226.3001.4450
4. predict与predict_proba区别
   a. https://blog.csdn.net/qq_43468807/article/details/105740396
   b. predict返回单个预测值
   c. predict_proba返回的是一个 n 行 k 列的数组，n 表示测试集中样本的个数， 第 i 行 j列的数值是模型预测 第 i 个预测样本为某个标签的概率，并且每一行的概率和为1。
5. L2 Regularised Logistic Regression.
   a. Workflow
      i. from sklearn.svm import l1_min_c
      cs = l1_min_c(X, t, loss='log')

   b. 语法
      i. https://scikit-learn.org/stable/modules/generated/sklearn.svm.l1_min_c.html
      ii. 默认 loss='squared_hinge'，代表L2正则化loss
      iii. Loss = log代logistics专用回归模型

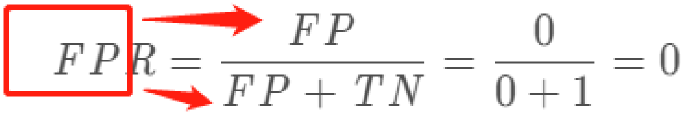
ROC & AUC

1. Reference
   a. https://blog.csdn.net/yinyu19950811/article/details/81288287?ops_request_misc=&request_id=&biz_id=102&utm_term=ROC&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~sobaiduweb~default-0-81288287.pc_v2_rank_blog_default&spm=1018.2226.3001.4450
2. roc_curve
   a. https://blog.csdn.net/w1301100424/article/details/84546194
   b. Workflow
      i. from sklearn import metrics
         clf1 = LogisticRegression().fit(X_train, y_train)
         y_pred1 = clf1.predict(X_test)
         y_pred_proba1 = clf1.predict_proba(X_test)[:,1]
         fpr1, tpr1, _ = metrics.roc_curve(y_test, y_pred_proba1)
   c. 上面最后一句中的_代表threshold，是判断数据的依据
   d. 公式



$$\boxed{TP}R = \frac{TP}{TP+FN} = \frac{1}{1+1} = 0.5$$

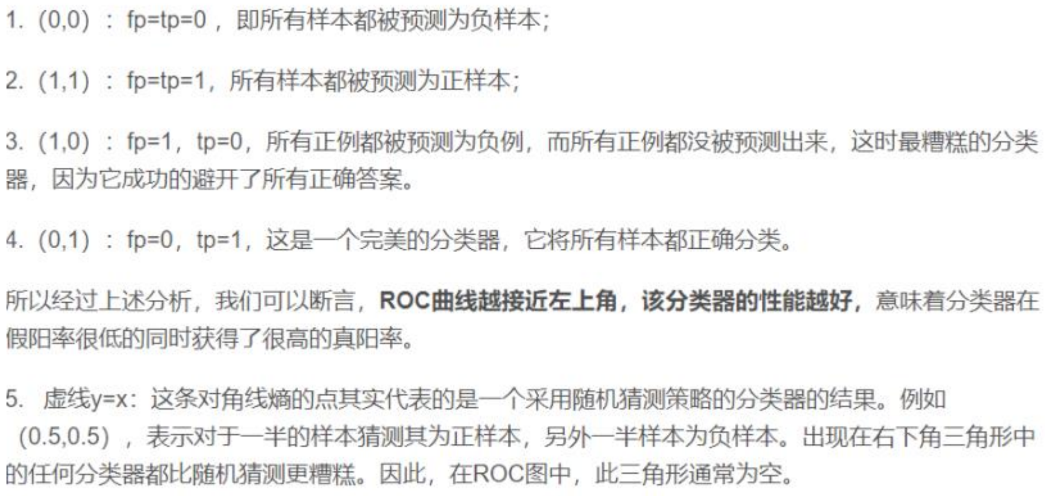
      i.



$$\boxed{FP}R = \frac{FP}{FP+TN} = \frac{0}{0+1} = 0$$

      ii. 可以按照箭头指示速记，即分子和分母第一项为公式名字，分母第二项为第一项取反

   e.



1. (0,0) ：fp=tp=0，即所有样本都被预测为负样本；

2. (1,1) ：fp=tp=1，所有样本都被预测为正样本；

3. (1,0) ：fp=1，tp=0，所有正例都被预测为负例，而所有正例都没被预测出来，这时最糟糕的分类器，因为它成功的避开了所有正确答案。

4. (0,1) ：fp=0，tp=1，这是一个完美的分类器，它将所有样本都正确分类。

所以经过上述分析，我们可以断言，**ROC曲线越接近左上角，该分类器的性能越好，**意味着分类器在假阳率很低的同时获得了很高的真阳率。

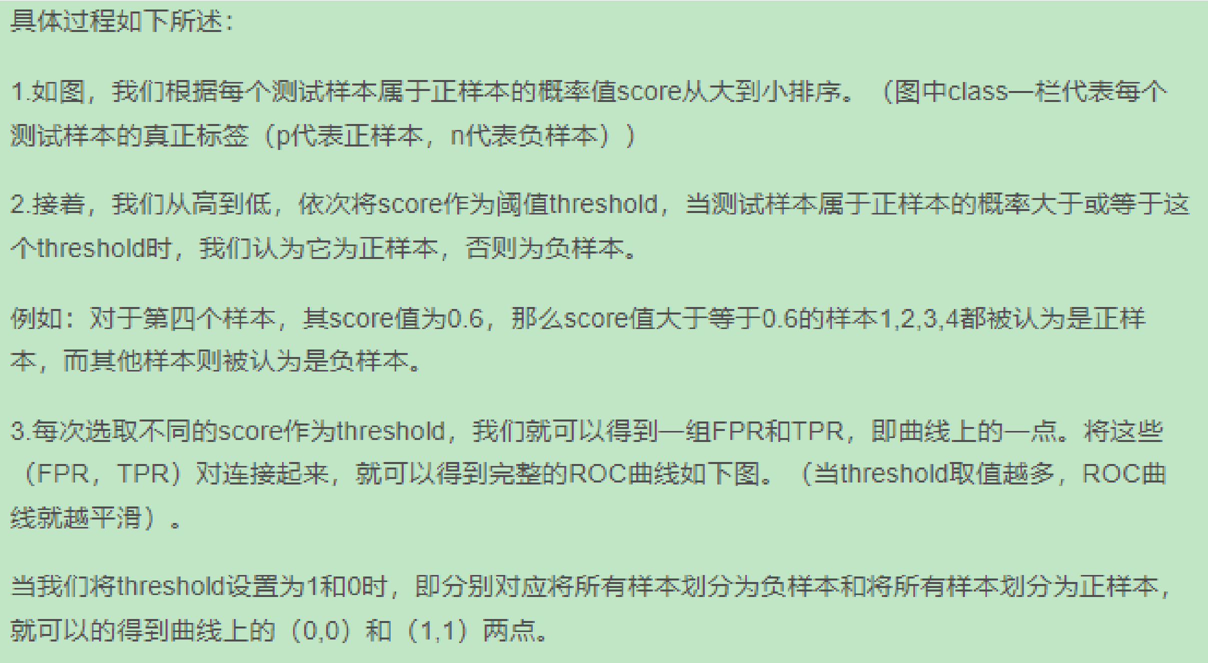
5. 虚线y=x：这条对角线熵的点其实代表的是一个采用随机猜测策略的分类器的结果。例如(0.5,0.5)，表示对于一半的样本猜测其为正样本，另外一半样本为负样本。出现在右下角三角形中的任何分类器都比随机猜测更糟糕。因此，在ROC图中，此三角形通常为空。

   f. 通常情况下ROC曲线是在y=x曲线上方的曲线，左上角为最理想模型，越靠近左上角越好

3. ROC, AUC and confusion Matrix
   a. 一组confusion matrix对应一个ROC曲线上的点

b. 通过更改thershold 获得多组ROC的值

c. 具体流程

i. 

具体过程如下所述：

1.如图，我们根据每个测试样本属于正样本的概率值score从大到小排序。（图中class一栏代表每个测试样本的真正标签（p代表正样本，n代表负样本））

2.接着，我们从高到低，依次将score作为阈值threshold，当测试样本属于正样本的概率大于或等于这个threshold时，我们认为它为正样本，否则为负样本。

例如：对于第四个样本，其score值为0.6，那么score值大于等于0.6的样本1,2,3,4都被认为是正样本，而其他样本则被认为是负样本。

3.每次选取不同的score作为threshold，我们就可以得到一组FPR和TPR，即曲线上的一点。将这些（FPR，TPR）对连接起来，就可以得到完整的ROC曲线如下图。（当threshold取值越多，ROC曲线就越平滑）。

当我们将threshold设置为1和0时，即分别对应将所有样本划分为负样本和将所有样本划分为正样本，就可以的得到曲线上的（0,0）和（1,1）两点。

4. roc_auc_score
   a. https://blog.csdn.net/NockinOnHeavensDoor/article/details/83384844?spm=1001.2101.3001.6650.1&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-1.no_search_link&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-1.no_search_link
   b. AUC为ROC曲线与x轴围成的面积，通常处于（0.5,1）之间，越接近1越好

其他

1. 用ListedColormap作plt.scatter()
   a. https://blog.csdn.net/MaxxiChen/article/details/106137315?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163465879316780265426346%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163465879316780265426346&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_click~default-1-106137315.first_rank_v2_pc_rank_v29&utm_term=ListedColormap&spm=1018.2226.3001.4187
2. np.meshgrid
   a. https://blog.csdn.net/lllxxq141592654/article/details/81532855
   b. 根据坐标矩阵生成网格点
3. Np.c_ / np.r_
   a. https://blog.csdn.net/weixin_41797117/article/details/80048688
   b. np.r_是按列连接两个矩阵，就是把两矩阵上下拼接，要求列数相等。

      np.c_是按行连接两个矩阵，就是把两矩阵左右拼接，要求行数相等。
4. Np.ravel
   a. https://blog.csdn.net/hanshuobest/article/details/78882425
   b. 把多维数组编程一位
5. cross_val_score
   a. https://blog.csdn.net/qq_36523839/article/details/80707678

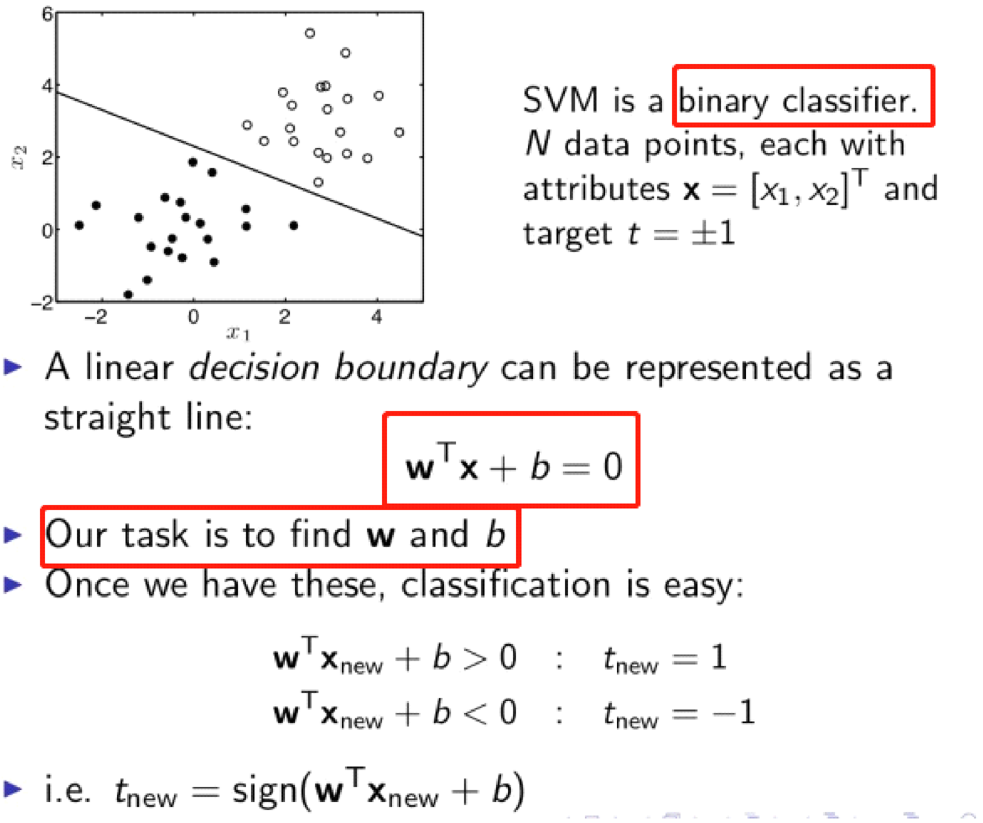
    b. https://blog.csdn.net/weixin_42211626/article/details/100064842
    c. 交叉验证

6. Np.logspace()
    a. https://blog.csdn.net/haiyan09/article/details/99717526
7. train_test_split
    a. https://blog.csdn.net/qq_41551450/article/details/106336005?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163470902616780264089017%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163470902616780264089017&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_click~default-2-106336005.first_rank_v2_pc_rank_v29&utm_term=train_test_split&spm=1018.2226.3001.4187
    b. test_size：样本与测试的比
    c. random_state：默认为None，即每次生成的 都不一样。如果为整数则每次生成的都一样

Lab4

31 October 2021 21:49

SVM

1. Reference
    a. https://blog.csdn.net/qq_31347869/article/details/88071930
    b. https://zhuanlan.zhihu.com/p/31886934
    c. https://blog.csdn.net/ab1213456/article/details/102214451
2. Hard margin
    a. 

SVM is a binary classifier. N data points, each with attributes $\mathbf{x} = [x_1, x_2]^\mathsf{T}$ and target $t = \pm 1$

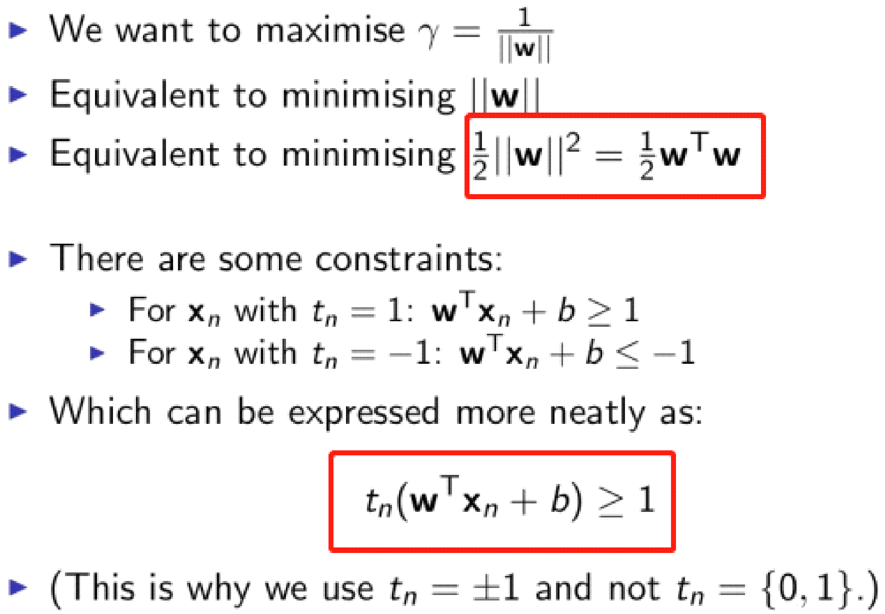
- A linear *decision boundary* can be represented as a straight line:

$$\boxed{\mathbf{w}^\mathsf{T}\mathbf{x} + b = 0}$$

- Our task is to find $\mathbf{w}$ and b
- Once we have these, classification is easy:

$$\begin{aligned} \mathbf{w}^\mathsf{T}\mathbf{x}_{\text{new}} + b > 0 \quad &: \quad t_{\text{new}} = 1 \\ \mathbf{w}^\mathsf{T}\mathbf{x}_{\text{new}} + b < 0 \quad &: \quad t_{\text{new}} = -1 \end{aligned}$$

- i.e. $t_{\text{new}} = \text{sign}(\mathbf{w}^\mathsf{T}\mathbf{x}_{\text{new}} + b)$

    b. 

- We want to maximise $\gamma = \frac{1}{||\mathbf{w}||}$
- Equivalent to minimising $||\mathbf{w}||$
- Equivalent to minimising $\boxed{\frac{1}{2}||\mathbf{w}||^2 = \frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{w}}$

- There are some constraints:
    - For $\mathbf{x}_n$ with $t_n = 1$: $\mathbf{w}^\mathsf{T}\mathbf{x}_n + b \geq 1$
    - For $\mathbf{x}_n$ with $t_n = -1$: $\mathbf{w}^\mathsf{T}\mathbf{x}_n + b \leq -1$
- Which can be expressed more neatly as:

$$\boxed{t_n(\mathbf{w}^\mathsf{T}\mathbf{x}_n + b) \geq 1}$$

- (This is why we use $t_n = \pm 1$ and not $t_n = \{0, 1\}$.)

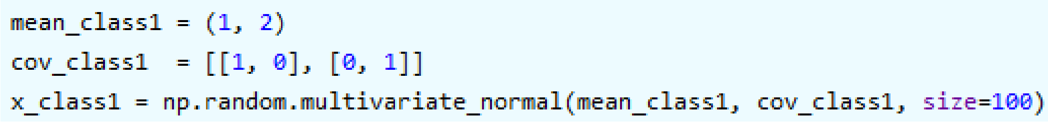
3. Soft margin

   a. $$\min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\zeta_i$$
      i. 在hard margin的基础上再加上一项，用C来限制
      ii. C=1时前后比重相同，>1则主要优化后半部分，反之优化前半部分
4. RBF
   a. $$k(\mathbf{x}_n, \mathbf{x}_m) = \exp\left\{-\beta(\mathbf{x}_n - \mathbf{x}_m)^{\mathrm{T}}(\mathbf{x}_n - \mathbf{x}_m)\right\}$$
      i. 这里的β就是语法里面的gamma
      ii. rbf函数需要找到合适的gamma，过大过拟合，过小欠拟合

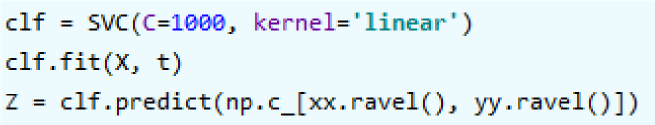
Rbf

1. SVM如果使用rbf核函数的技巧的话，不太适应于大样本和大的特征数的情况, 因此提出了单独的RBF函数。
2. https://towardsdatascience.com/radial-basis-function-rbf-kernel-the-go-to-kernel-acf0d22c798a

其他

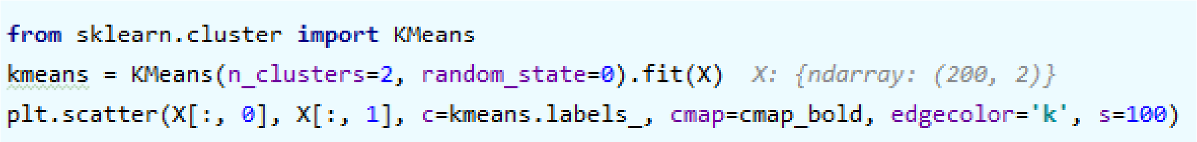
1. np.random.multivariate_normal
   a. https://blog.csdn.net/zch1990s/article/details/80005940
   b. https://www.zhihu.com/question/288946037/answer/649328934
   c. 

```
mean_class1 = (1, 2)
cov_class1  = [[1, 0], [0, 1]]
x_class1 = np.random.multivariate_normal(mean_class1, cov_class1, size=100)
```

   d. 生成一个100x2的数组，第一列均值1方差1，第二列均值2方差1
   e. cov可以用np.eye（）生成，必须是对角矩阵
2. sklearn.svm.SVC
   a. https://blog.csdn.net/icanx/article/details/96270752
   b. 

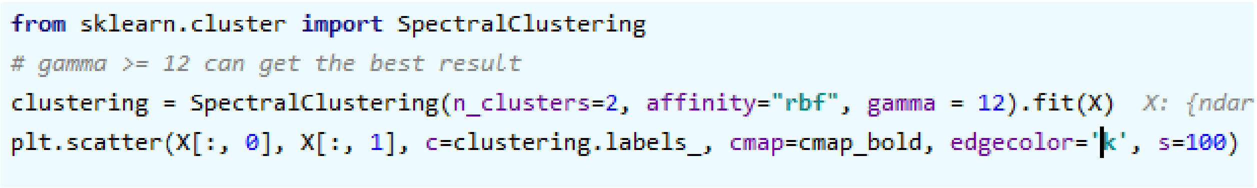
```
clf = SVC(C=1000, kernel='linear')
clf.fit(X, t)
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
```

   c. C为错误惩罚参数，默认为1
   d. C是间隔大小和噪声的一种平衡，C过大导致过拟合，过小导致欠拟合甚至不会收敛
   e. kernel为算法的内核函数，’ linear’ ，‘poly’ ，‘rbf’ ，‘sigmoid’ ，‘precomputed’ 或者callable之一， 默认rbf
3. Kmeans
   a. https://blog.csdn.net/mago2015/article/details/82729209
   b. 对于给定的样本集，按照样本之间的距离大小，将样本集划分为K个簇。让簇内的点尽量紧密的连在一起，而让簇间的距离尽量的大。

c. 

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2, random_state=0).fit(X)   X: {ndarray: (200, 2)}
plt.scatter(X[:, 0], X[:, 1], c=kmeans.labels_, cmap=cmap_bold, edgecolor='k', s=100)
```

   i. n_cluster: 中心的数量
   ii. Random_state:中心的状态条件
   iii. max_iter:迭代次数

4. SKlearn.clustering.spectralClustering
   a. https://blog.csdn.net/Hero_Never_GIVE_UP/article/details/88635495
   b. 

```
from sklearn.cluster import SpectralClustering
# gamma >= 12 can get the best result
clustering = SpectralClustering(n_clusters=2, affinity="rbf", gamma = 12).fit(X)   X: {ndar
plt.scatter(X[:, 0], X[:, 1], c=clustering.labels_, cmap=cmap_bold, edgecolor='k', s=100)
```

Coursework

21 October 2021 17:20

Model selection for clustering

1.

Feature engineering: normalisation, selection

1. https://blog.csdn.net/song430/article/details/87944768
2. https://blog.csdn.net/zpainter/article/details/98588658

Interpretability

1. https://www.zhihu.com/question/320688440/answer/697034632

Visualisation (Prediction Uncertainty, Embeddings, and Projection)

1. 输入图像，处理提高精度

Engines of Machine Learning algorithms: Inference methods

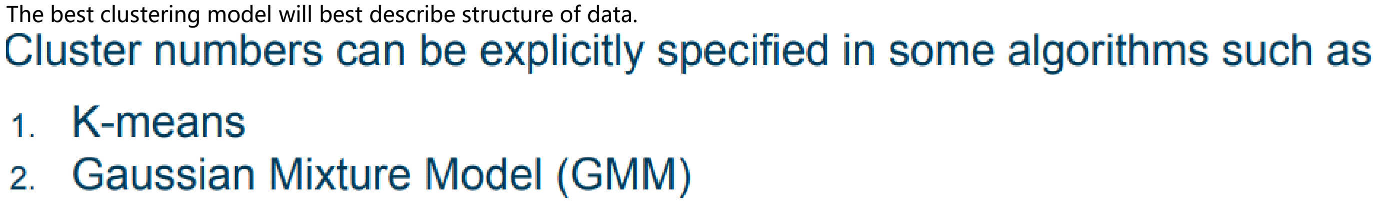
1. MCMC
    a. https://www.cnblogs.com/pinard/p/6625739.html
2. ABC
    a. https://baike.baidu.com/item/ABC/12000974?fr=aladdin

Model selection for clustering

28 October 2021 14:19

Slides

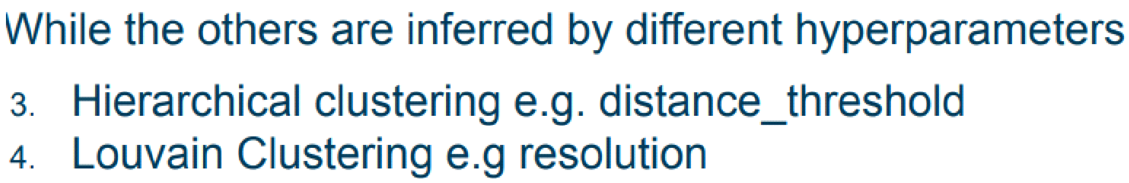
1. For machine learning in general, it is a process of choosing the best model candidate among a family of algorithms and/or among different hyperparameters
2. The best clustering model will best describe structure of data.



Cluster numbers can be explicitly specified in some algorithms such as

1. K-means
2. Gaussian Mixture Model (GMM)

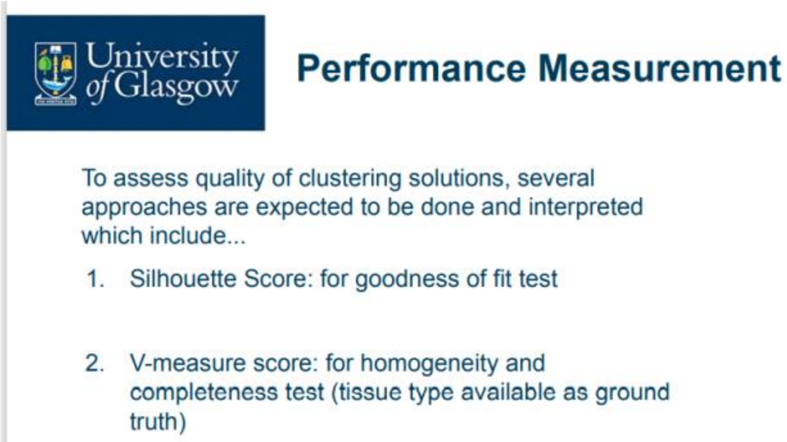
3.



While the others are inferred by different hyperparameters

3. Hierarchical clustering e.g. distance_threshold
4. Louvain Clustering e.g resolution

4.

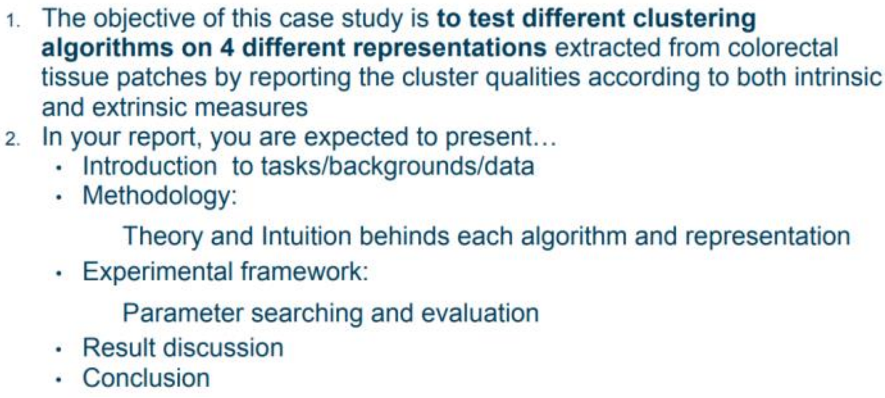


Performance Measurement

To assess quality of clustering solutions, several approaches are expected to be done and interpreted which include...

1. Silhouette Score: for goodness of fit test
2. V-measure score: for homogeneity and completeness test (tissue type available as ground truth)

5.



1. The objective of this case study is **to test different clustering algorithms on 4 different representations** extracted from colorectal tissue patches by reporting the cluster qualities according to both intrinsic and extrinsic measures
2. In your report, you are expected to present…
   - Introduction to tasks/backgrounds/data
   - Methodology:

     Theory and Intuition behinds each algorithm and representation
   - Experimental framework:

     Parameter searching and evaluation
   - Result discussion
   - Conclusion

模型选择

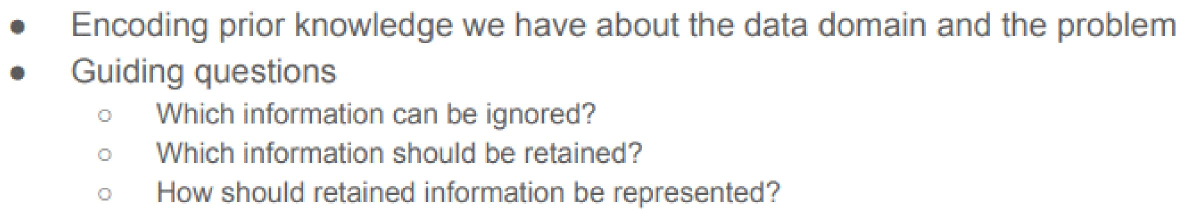
1. https://www.bilibili.com/video/BV1kX4y1g7jp?spm_id_from=333.999.0.0
2. https://www.bilibili.com/video/BV1fi4y1K7Ke?from=search&seid=12474986287880934605&spm_id_from=333.337.0.0
3. https://blog.csdn.net/ustbbsy/article/details/80960652?spm=1001.2101.3001.6650.3&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-3.no_search_link&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-3.no_search_link

Feature engineering

28 October 2021 14:05

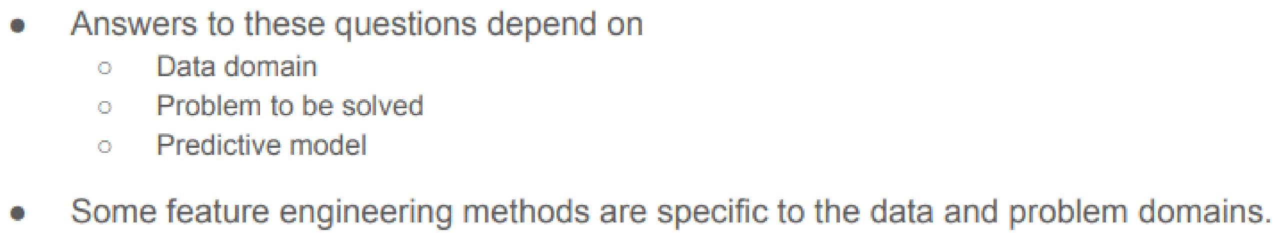
Slides

1. Feature engineering is the process of transforming raw data into something that better represents the learning problem to the predictive model, resulting in improved generalization to unseen data.



- Encoding prior knowledge we have about the data domain and the problem
- Guiding questions
  - Which information can be ignored?
  - Which information should be retained?
  - How should retained information be represented?

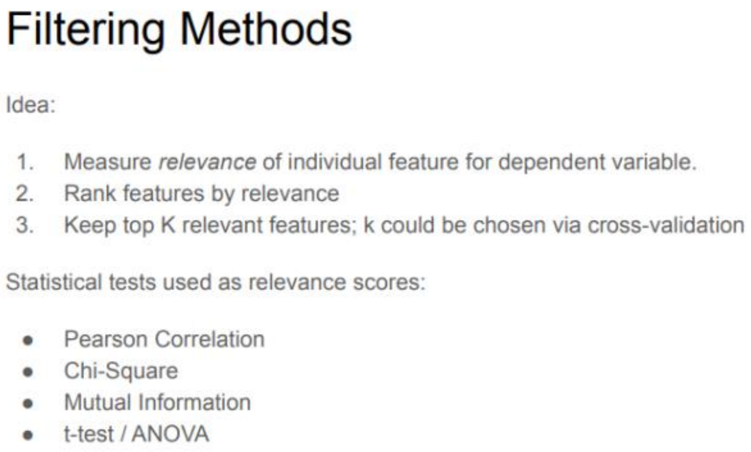
2.



- Answers to these questions depend on
  - Data domain
  - Problem to be solved
  - Predictive model
- Some feature engineering methods are specific to the data and problem domains.

3. Feature selection

a.



Filtering Methods

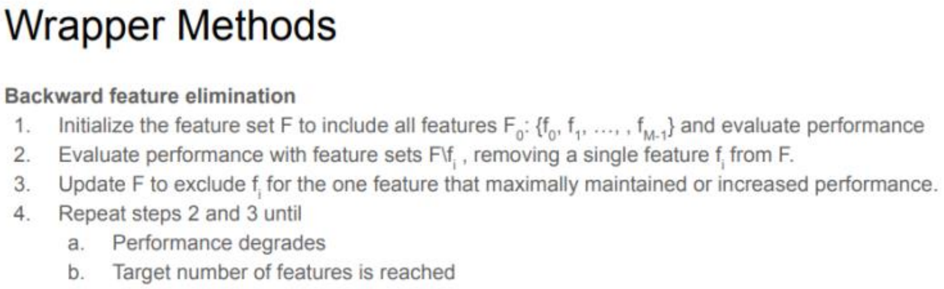
Idea:

1. Measure *relevance* of individual feature for dependent variable.
2. Rank features by relevance
3. Keep top K relevant features; k could be chosen via cross-validation

Statistical tests used as relevance scores:

- Pearson Correlation
- Chi-Square
- Mutual Information
- t-test / ANOVA

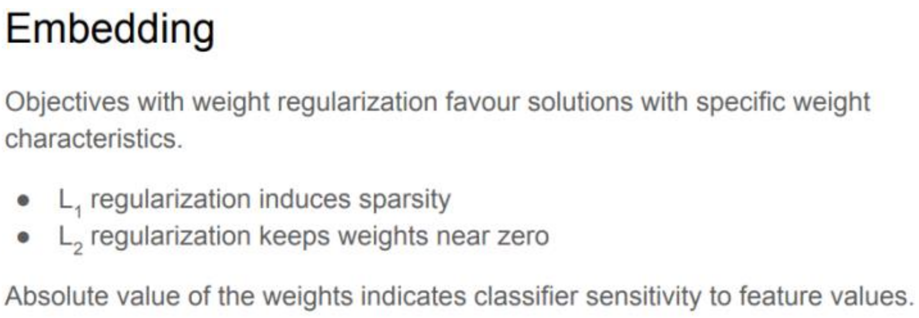
b.



Wrapper Methods

Backward feature elimination

1. Initialize the feature set F to include all features F_0: $\{f_0, f_1, \dots, f_{M-1}\}$ and evaluate performance
2. Evaluate performance with feature sets $F \backslash f_i$, removing a single feature f_i from F.
3. Update F to exclude f_i for the one feature that maximally maintained or increased performance.
4. Repeat steps 2 and 3 until
   a. Performance degrades
   b. Target number of features is reached



Embedding

Objectives with weight regularization favour solutions with specific weight characteristics.

c.

- L_1 regularization induces sparsity
- L_2 regularization keeps weights near zero

Absolute value of the weights indicates classifier sensitivity to feature values.

d.

斯坦福21秋季

1. https://www.bilibili.com/video/BV1t44y1x7Hw?from=search&seid=8393385331080645665&spm_id_from=333.337.0.0
2. 机器学习的算法比较喜欢定义的比较好的、它能比较好的去处理的、固定长度的输入输出
3. 常见数据的特征工程
   a. 表数据
      i. 对于整型或浮点型的数据，可以直接用或者是把最大最小值拿出来，再把这个数据分成n个区间，如果值在区间中，则会给它对应区间的下标i【这样可以让机器学习算法不去纠结一个具体的值（细粒度的值）】
      ii. 对于类别的数据，一般采用one-hot（独热）编码（虽然有n列，但是只有每一列有值）【虽然有很多的类别但是常见的只有几个类，可以将少数的类别变成不确定的类别，只保留那些比较重要的类别，这样可以把这些重要的类别放到可控的单元内】
      iii. 对于时间的特征，将时间的数据弄成机器学习算法能知道这些天数中是有特殊意义的日子（周末、休息日、新年之类的)
      iv. 特征组合：这样子能拿到两两特征之间相关性的东西
   b. 文本数据
      i. Bag of woeds(BoW) model：把每一个词元(token)弄成one-hot编码，再把句子里的所有词元加起来【这里要注意的是 怎么样把词典构造出来，不能太大也不能太小；BoW model最大的问题在于原句子的信息丢失了】。
      ii. Word Embeddings(词嵌入)：将词变成一个向量，向量之间具有一定的语义性的(两个词之间对应的向量之间的内积比较近的话，说明这两个词在语义上来说是比较相近的)
   c. 图片与视频
      i. 利用已经预训练好的网络的结果做特征

Visualization

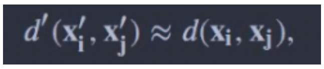
22 November 2021 18:15

Target

特征提取 到达98%的准确率

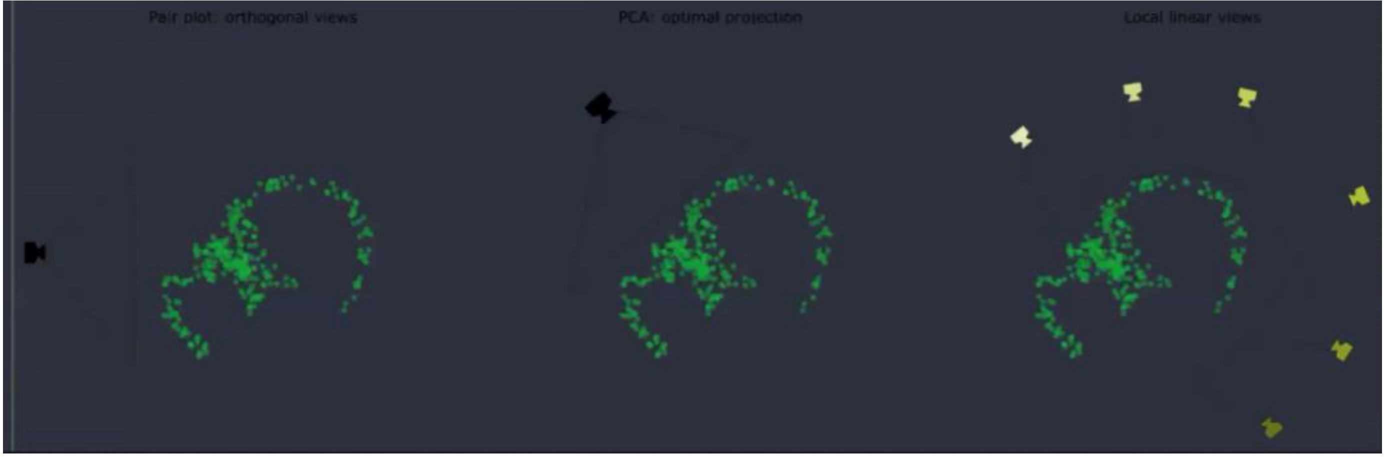
看图调参数调整到85%, 75%过关

Introduction

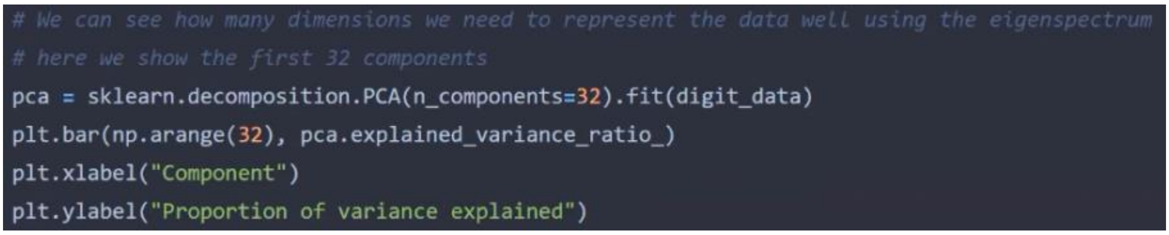
1. 人处理高纬度数据困难, 要把高纬度数据map到低位空间中以获得更好的效果
2. 

   $d'(\mathbf{x}'_i, \mathbf{x}'_j) \approx d(\mathbf{x}_i, \mathbf{x}_j),$
3. 为了更准确的map, 需要计算d之间的距离

Dimensional reduction: embeddings

1. https://blog.csdn.net/qq_42797457/article/details/100675654?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163809444116780269869799%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163809444116780269869799&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~baidu_landing_v2~default-2-100675654.first_rank_v2_pc_rank_v29&utm_term=sklearn.manifold.Isomap&spm=1018.2226.3001.4187
2. Projection
   a. Pairplot, PCA, Local linear views
   b. 

PCA(principal component analysis)

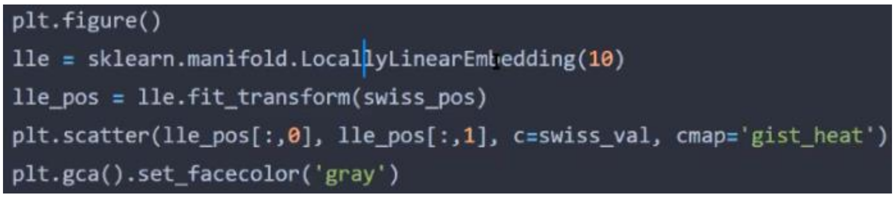
1. 简介
   a. https://blog.csdn.net/foxchopin/article/details/78064684
   b. https://blog.csdn.net/sxb0841901116/article/details/83816356?spm=1001.2101.3001.6650.1&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-1.no_search_link&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-1.no_search_link
2. 降维
   a. 

```
# We can see how many dimensions we need to represent the data well using the eigenspectrum
# here we show the first 32 components
pca = sklearn.decomposition.PCA(n_components=32).fit(digit_data)
plt.bar(np.arange(32), pca.explained_variance_ratio_)
plt.xlabel("Component")
plt.ylabel("Proportion of variance explained")
```

   b. 查看每种降维数据可以显示出的特征数量

LLE(locally linear embedding)

1. 简介
   a. https://blog.csdn.net/scott198510/article/details/76099630?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163809298016780265416239%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163809298016780265416239&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-76099630.first_rank_v2_pc_rank_v29&utm_term=LLE&spm=1018.2226.3001.4187
   b. https://blog.csdn.net/weixin_39777626/article/details/79710482?spm=1001.2101.3001.6650.1&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-1.no_search_link&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-1.no_search_link
   c. https://blog.csdn.net/weixin_40411446/article/details/78310264?spm=1001.2101.3001.6650.2&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-2.no_search_link&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-2.no_search_link

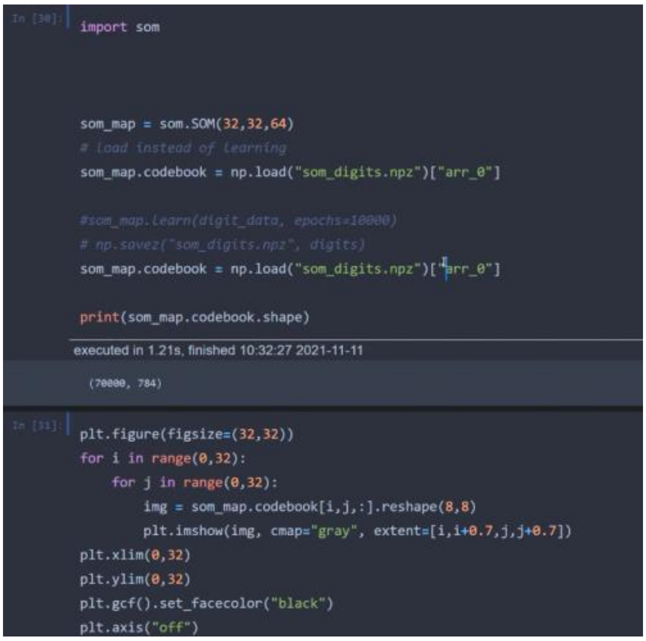
2. 使用
    a. 

```
plt.figure()
lle = sklearn.manifold.LocallyLinearEmbedding(10)
lle_pos = lle.fit_transform(swiss_pos)
plt.scatter(lle_pos[:,0], lle_pos[:,1], c=swiss_val, cmap='gist_heat')
plt.gca().set_facecolor('gray')
```

ISOMAP()

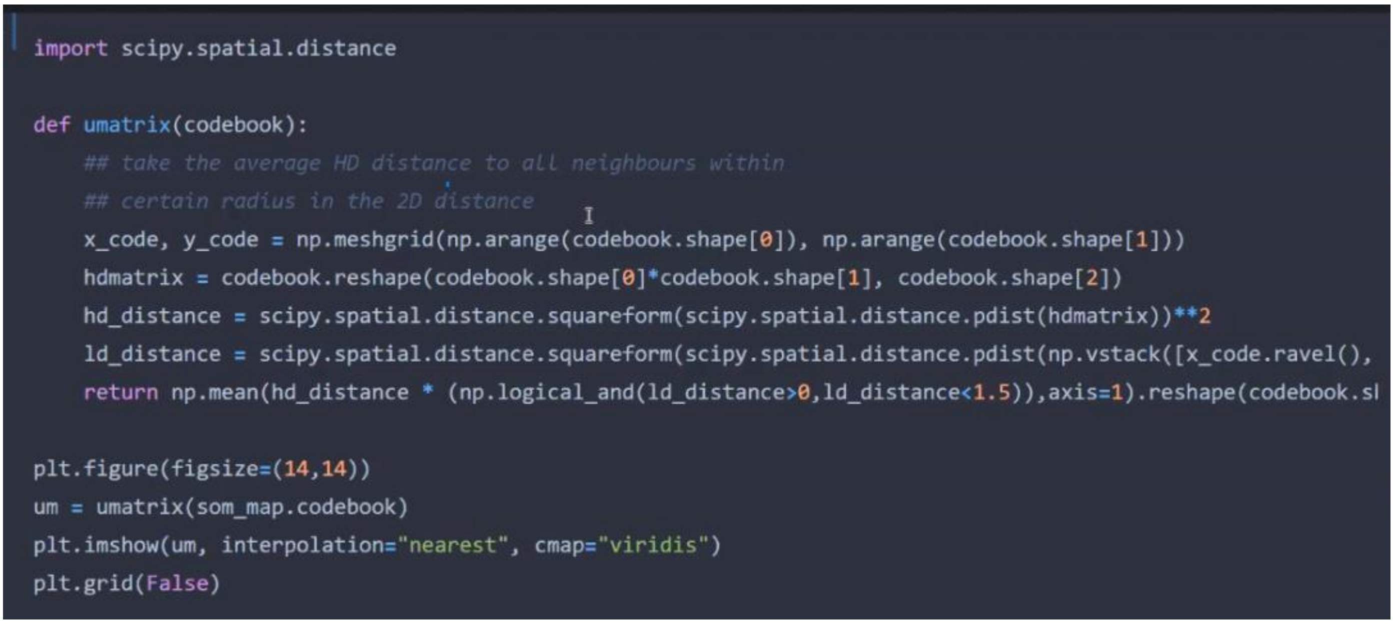
1. 简介
    a. https://blog.csdn.net/weixin_45234485/article/details/109965084?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163809444116780269869799%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163809444116780269869799&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~baidu_landing_v2~default-1-109965084.first_rank_v2_pc_rank_v29&utm_term=sklearn.manifold.Isomap&spm=1018.2226.3001.4187
    b. https://blog.csdn.net/inkky/article/details/81128650?spm=1001.2101.3001.6650.11&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-11.no_search_link&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-11.no_search_link

Self-organizing map

1. 介绍
2. 用法
    a. 

```
In [30]:  import som



          som_map = som.SOM(32,32,64)
          # load instead of learning
          som_map.codebook = np.load("som_digits.npz")["arr_0"]

          #som_map.learn(digit_data, epochs=10000)
          # np.savez("som_digits.npz", digits)
          som_map.codebook = np.load("som_digits.npz")["arr_0"]

          print(som_map.codebook.shape)
         executed in 1.21s, finished 10:32:27 2021-11-11

           (70000, 784)

In [31]:  plt.figure(figsize=(32,32))
          for i in range(0,32):
              for j in range(0,32):
                  img = som_map.codebook[i,j,:].reshape(8,8)
                  plt.imshow(img, cmap="gray", extent=[i,i+0.7,j,j+0.7])
          plt.xlim(0,32)
          plt.ylim(0,32)
          plt.gcf().set_facecolor("black")
          plt.axis("off")
```

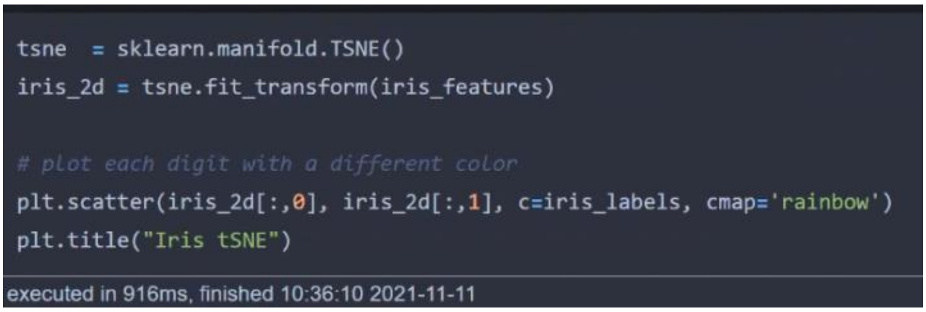
U-Matrix

1. 介绍
    a. 用于观测图像像素点变化大小
2. 用法
    a. 

```
import scipy.spatial.distance

def umatrix(codebook):
    ## take the average HD distance to all neighbours within
    ## certain radius in the 2D distance
    x_code, y_code = np.meshgrid(np.arange(codebook.shape[0]), np.arange(codebook.shape[1]))
    hdmatrix = codebook.reshape(codebook.shape[0]*codebook.shape[1], codebook.shape[2])
    hd_distance = scipy.spatial.distance.squareform(scipy.spatial.distance.pdist(hdmatrix))**2
    ld_distance = scipy.spatial.distance.squareform(scipy.spatial.distance.pdist(np.vstack([x_code.ravel(),
    return np.mean(hd_distance * (np.logical_and(ld_distance>0,ld_distance<1.5)),axis=1).reshape(codebook.sl

plt.figure(figsize=(14,14))
um = umatrix(som_map.codebook)
plt.imshow(um, interpolation="nearest", cmap="viridis")
plt.grid(False)
```

Tsne

1. 简介
   a. https://blog.csdn.net/qq_23534759/article/details/80457557?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163809608816780269879932%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163809608816780269879932&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~baidu_landing_v2~default-4-80457557.first_rank_v2_pc_rank_v29&utm_term=sklearn.manifold.tsne&spm=1018.2226.3001.4187
   b. https://blog.csdn.net/weixin_39393430/article/details/110560489?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163809608816780269879932%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163809608816780269879932&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-2-110560489.first_rank_v2_pc_rank_v29&utm_term=sklearn.manifold.tsne&spm=1018.2226.3001.4187
2. 用法
   a.

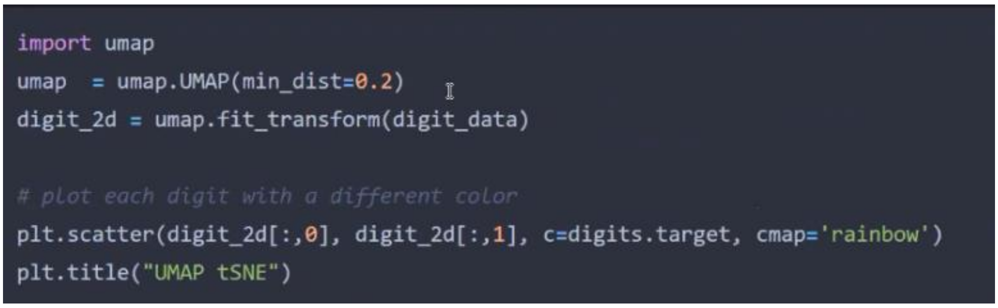


```
tsne  = sklearn.manifold.TSNE()
iris_2d = tsne.fit_transform(iris_features)

# plot each digit with a different color
plt.scatter(iris_2d[:,0], iris_2d[:,1], c=iris_labels, cmap='rainbow')
plt.title("Iris tSNE")
```

```
executed in 916ms, finished 10:36:10 2021-11-11
```

UMAP(uniform manifold approximation and projection)

1. 简介
2. 用法
   a.

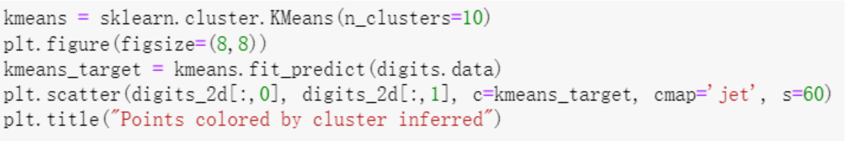


```
import umap
umap  = umap.UMAP(min_dist=0.2)
digit_2d = umap.fit_transform(digit_data)

# plot each digit with a different color
plt.scatter(digit_2d[:,0], digit_2d[:,1], c=digits.target, cmap='rainbow')
plt.title("UMAP tSNE")
```

   b. 一般最先用这个方法来看数据是结构

Clustering

1. 简介
2. 用法
   a.



```
kmeans = sklearn.cluster.KMeans(n_clusters=10)
plt.figure(figsize=(8,8))
kmeans_target = kmeans.fit_predict(digits.data)
plt.scatter(digits_2d[:,0], digits_2d[:,1], c=kmeans_target, cmap='jet', s=60)
plt.title("Points colored by cluster inferred")
```

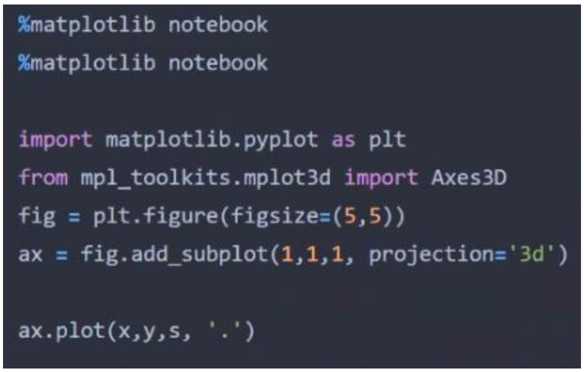
窗函数的比较

1.

其他

1. sklearn参数解析
   a. https://blog.csdn.net/NOT_GUY/article/details/84932222
2. sklearn.manifold
   a. https://blog.csdn.net/The_Time_Runner/article/details/89737002?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163809444116780269828245%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=163809444116780269828245&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-3-89737002.first_rank_v2_pc_rank_v29&utm_term=sklearn.manifold.Isomap&spm=1018.2226.3001.4187
3. fig, ax = plt.subplots(figsize = (a, b))
   a. https://blog.csdn.net/weixin_46649052/article/details/107424134
   b. fig为图像, ax为小图的对象
4. Plt.rcParams()
   a. https://blog.csdn.net/hezuijiudexiaobai/article/details/104778250/
5. Sns.pariplot()
   a. https://www.cnblogs.com/cgmcoding/p/13274481.html
6. 3d画图

a.



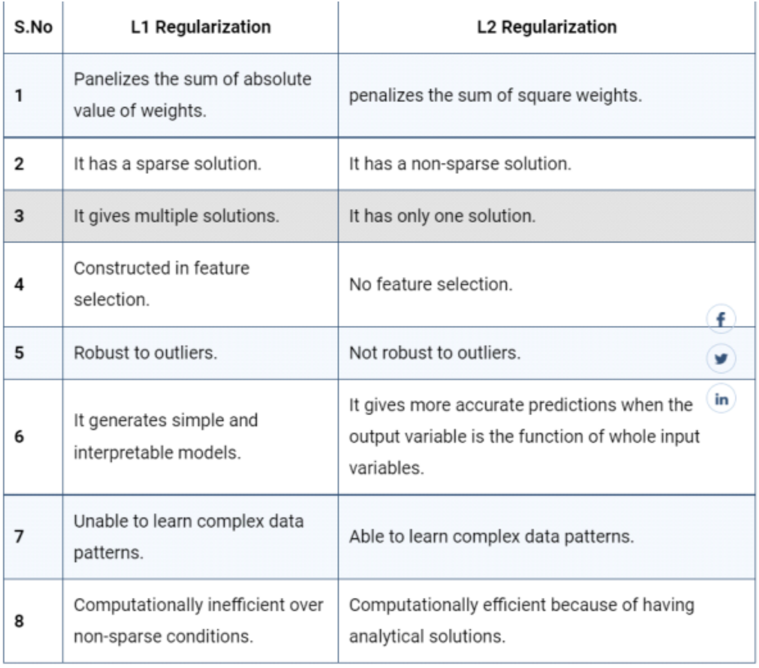
```
%matplotlib notebook
%matplotlib notebook

import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(figsize=(5,5))
ax = fig.add_subplot(1,1,1, projection='3d')

ax.plot(x,y,s, '.')
```

Exam

26 November 2021 14:12

2021_solution

1. Olympic
    - a. Rescale method
        - i. Lab2
        - ii. Normalization and Standardization
        - iii. After rescale, it does not become unfeasible to fit polynomials
        - iv. 答案给出的是Z-score中心化
            - 1) https://fanfanzhisu.blog.csdn.net/article/details/84498469?spm=1001.2101.3001.6650.3&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-3.no_search_link&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-3.no_search_link
            - 2) 计算方法为: 新数据=（原数据-均值）/标准差
        - v. https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35
        - vi. https://machinelearningmastery.com/how-to-improve-neural-network-stability-and-modeling-performance-with-data-scaling/
        - vii. https://blog.csdn.net/weixin_39918928/article/details/110567055
    - b. 会用到模型, 建议提前准备好
    - c. Rbf的参数
        - i. https://towardsdatascience.com/radial-basis-function-rbf-kernel-the-go-to-kernel-acf0d22c798a
        - ii. how would you mitigate the risk[unknown]
    - d. Fourier analysis
2. Classification question
    - a. logistic regression, 检测w是否合适, 说明合适和不合适的情况
        - i. likelihood function [unknown]
    - b. 给出数据反推模型
        - i. polynomial regression: 数据波动大, 绝对值大
        - ii. L2-regularised logistic regression: 由于含平方项绝对值比L1的小
        - iii. L1-regularised logistic regression: 有0值
    - c. 比较b三种模型的效果
        - i. With the same x, L1, L2 better generality
        - ii. https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c
        - iii.



S.No	L1 Regularization	L2 Regularization
1	Panelizes the sum of absolute value of weights.	penalizes the sum of square weights.
2	It has a sparse solution.	It has a non-sparse solution.
3	It gives multiple solutions.	It has only one solution.
4	Constructed in feature selection.	No feature selection.
5	Robust to outliers.	Not robust to outliers.
6	It generates simple and interpretable models.	It gives more accurate predictions when the output variable is the function of whole input variables.
7	Unable to learn complex data patterns.	Able to learn complex data patterns.
8	Computationally inefficient over non-sparse conditions.	Computationally efficient because of having analytical solutions.

        - iv. 有0值的一定是L1,
        - v. 区分L1,L2和lasso和ridge, 通常情况下用L1和L2, Lasso和ridge只有线性回归的时候才能用
            - 1) https://blog.csdn.net/a13526863959/article/details/84314031

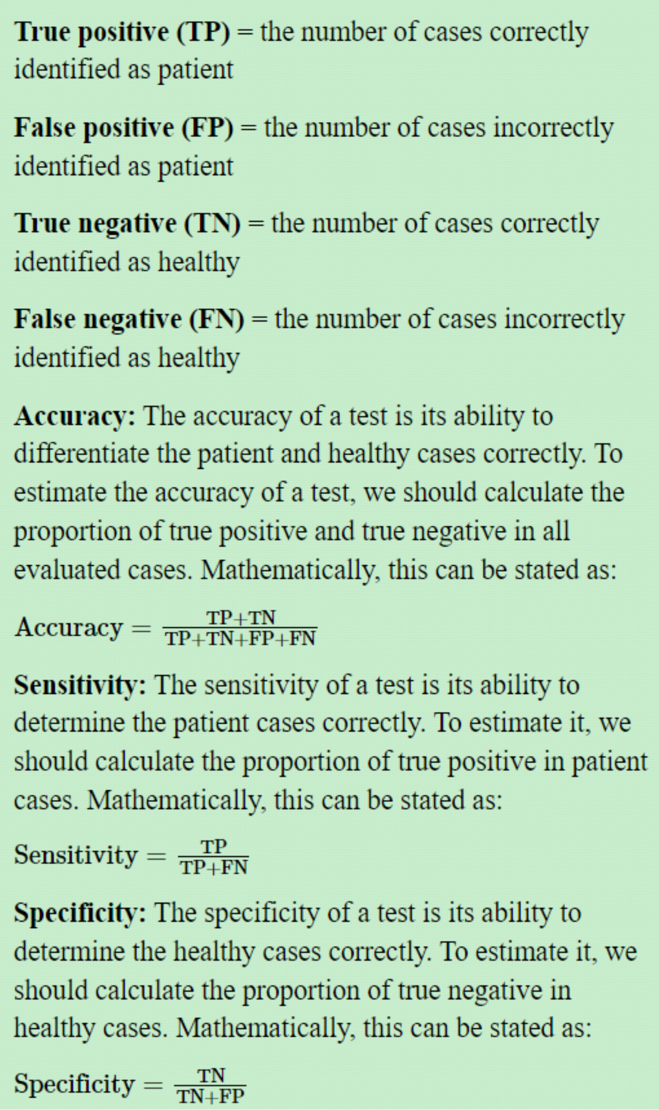
    d. Likelihood of a single label
        i. CSI, 假阳

3. Clustering question[unknown]
    a. Describe clustering results with parameters estimation, initial conditions and selecting the number of clusters.
    b. 调参方法
    c. 同a
    d. 调参方法

Mock paper

1. Linear regression
    a. Squared loss and absolute loss[unknown]
    b. Square loss - assumption that noise is normal distribution
    c. Complexity - the more complex the model, the less the loss, loss favor complex model
    d. 带入求解
    e. rbf参数的选择方法[unknown]
        i. https://blog.csdn.net/wn314/article/details/79972988
        ii. Gamma: 单个训练样本的大小, 值越小影响越大
        iii. c: 误分类样本和分类界面的平衡, 值越低分界面越平滑, 越高分界面约曲折使得被正确分类
    f. Difference between RBF and linear regression

2. Classification
    a. Definition of generalization and over-fitting
    b. Compute the accuracy, sensitivity and specificity
        i. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4614595/
        ii.



True positive (TP) = the number of cases correctly identified as patient

False positive (FP) = the number of cases incorrectly identified as patient

True negative (TN) = the number of cases correctly identified as healthy

False negative (FN) = the number of cases incorrectly identified as healthy

Accuracy: The accuracy of a test is its ability to differentiate the patient and healthy cases correctly. To estimate the accuracy of a test, we should calculate the proportion of true positive and true negative in all evaluated cases. Mathematically, this can be stated as:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

Sensitivity: The sensitivity of a test is its ability to determine the patient cases correctly. To estimate it, we should calculate the proportion of true positive in patient cases. Mathematically, this can be stated as:

$$\text{Sensitivity} = \frac{TP}{TP+FN}$$

Specificity: The specificity of a test is its ability to determine the healthy cases correctly. To estimate it, we should calculate the proportion of true negative in healthy cases. Mathematically, this can be stated as:

$$\text{Specificity} = \frac{TN}{TN+FP}$$

c. 解释AUC和ROC

d. 求AUC

e. Q2-e不会考, inner product

3. Unsupervised learning

a. Steps of k-means

b. How to select number of clusters in K-means

i. https://towardsdatascience.com/an-approach-for-choosing-number-of-clusters-for-k-means-c28e614ecb2c

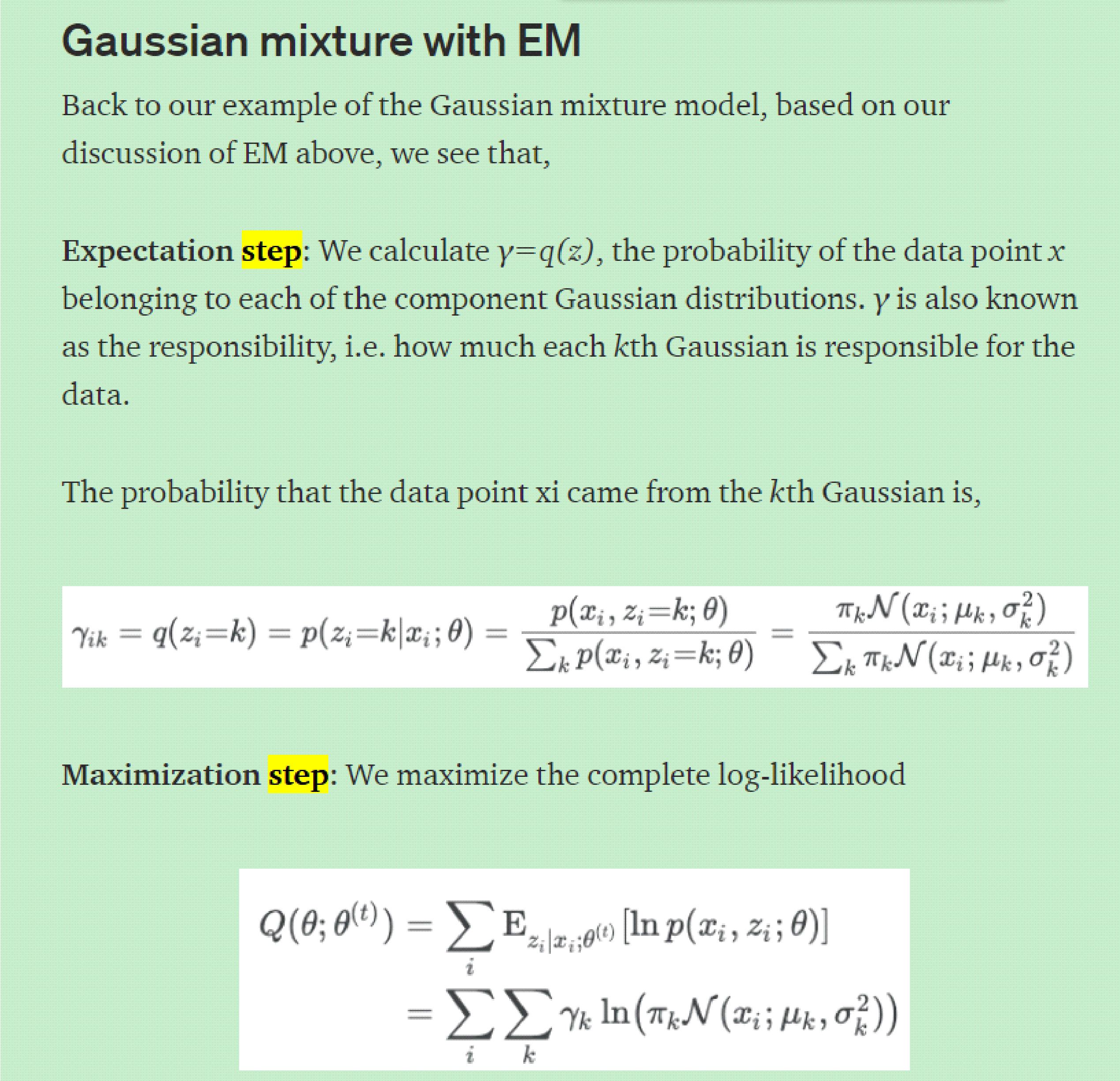
ii. 中心数越多, 几何距离越短, 因此不能用几何距离作为挑选中心数的标准

c. EM

i. https://aishack.in/tutorials/expectation-maximization-gaussian-mixture-model-mixtures/

ii. In the Estep, the estimated parameters are the expected assignment probabilities for each data point to each Gaussian component [2]. In the Mstep, the mean and covariance of Gaussian components and the mixing coefficients [2]

iii.



Gaussian mixture with EM

Back to our example of the Gaussian mixture model, based on our discussion of EM above, we see that,

Expectation step: We calculate $\gamma=q(z)$, the probability of the data point x belonging to each of the component Gaussian distributions. γ is also known as the responsibility, i.e. how much each kth Gaussian is responsible for the data.

The probability that the data point xi came from the kth Gaussian is,

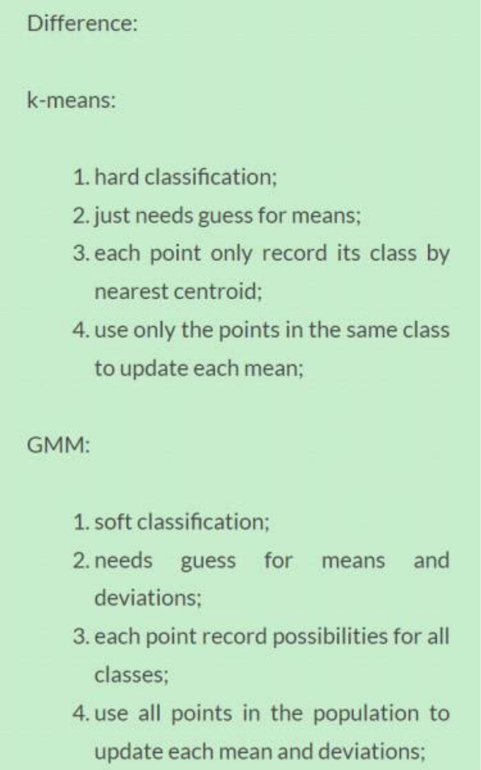
$$\gamma_{ik} = q(z_i{=}k) = p(z_i{=}k|x_i;\theta) = \frac{p(x_i, z_i{=}k;\theta)}{\sum_k p(x_i, z_i{=}k;\theta)} = \frac{\pi_k \mathcal{N}(x_i;\mu_k,\sigma_k^2)}{\sum_k \pi_k \mathcal{N}(x_i;\mu_k,\sigma_k^2)}$$

Maximization step: We maximize the complete log-likelihood

$$\begin{aligned} Q(\theta;\theta^{(t)}) &= \sum_i \mathbf{E}_{z_i|x_i;\theta^{(t)}}\left[\ln p(x_i,z_i;\theta)\right] \\ &= \sum_i \sum_k \gamma_k \ln\left(\pi_k \mathcal{N}(x_i;\mu_k,\sigma_k^2)\right) \end{aligned}$$

d. differences between K-means and Gaussian mixture models

i. https://ultraquartz.wordpress.com/2016/04/22/difference-between-k-means-and-gmm/

ii. 

Difference:

k-means:

1. hard classification;
2. just needs guess for means;
3. each point only record its class by nearest centroid;
4. use only the points in the same class to update each mean;

GMM:

1. soft classification;
2. needs guess for means and deviations;
3. each point record possibilities for all classes;
4. use all points in the population to update each mean and deviations;

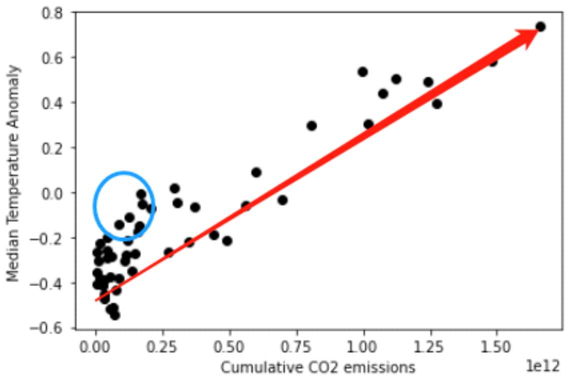
e. overcoming the local optimality of K-means
   i. CV
   ii. Perform multiple restarts

f. When to choose mixture model
   i. The distance is not easy to compute
   ii. The likelihood is easy to compute

08 December 2021 17:04

Q1

1.
   a. Rescale method: z-score, by applying x_new = (x-mean(x))/std(x)
   b. Reason: this method can make x_new small so that the computation is stable. If we do not rescale the data, the value can be too big to calculate after polynomial regression and even cause stack overflow.
2. a
   a. As we are using a polynomial regression model with order of 1, it should be linear regression, which means we want to fine a line best fits the data.
   b. Bad Subset: data away from the red line. For example, data points in the blue circle.
   c.



3.
   a. Advantage: flexibility
   b. Disadvantage: with the increase of iteration, the numerical stability will decrease and it will become overfitting.
4.
   a. A: lasso, as it generally not overfit.
   b. B: linear regression, as the model is overfitted.
   c. C: ridge, as there are zero points.

Q2

1.
   a. E = W(transpose) * Xn
   b. P = 1-1/(1+exp(-E))

	Class0	Class0	Class0	Class0	Class1	Class1	Class1	Class1
	E	P	E	P	E	P	E	P
[0.6,0.1]	0.7	0.33	0.6	0.35	0.7	0.67	0.1	0.52
[0.6,0.8]	1.4	0.2	0.6	0.35	1.4	0.8	0.8	0.69

2.
   a. AUC = 0.67
   b. Range of possible value: [0.3, 2.2). That is because the smallest value of noisy labeled data is 2.2 and the biggest value of correct labeled data is 0.3.
   c.
3.
   a. I prefer to correct the point with score of 8.8

Q3

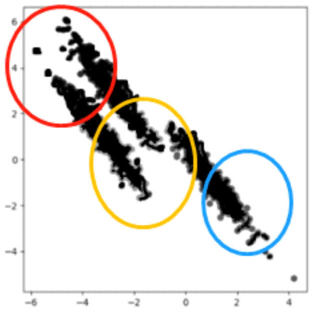
1.
   a. We cannot achieve the clustering objective if we directly apply K-means with Euclidean distance.
   b. K-means clustering divides all the data into K clusters according the distance between the cluster center and the data point. Thus, data points might be grouped into three groups locates on the top left(red), on the center(yellow) and on the bottom right(blue).
   c. 

Figure 3 A Original Data

2.
   a. Kernel K-means can help in this dataset as it makes K-means more flexible. With the help of kernel function, we can project the dataset into another domain which the features of data can be shown straight forward.
3.
   a. Mixture models can perform a better result than k-means.
   b. Mixture models is a method helps us think generatively. When we use mixture models, we can choose the distribution we want and make a linear combination of them. It is good for density estimation
4.
   a. We might achieve the clustering objective if we directly apply K-means with Euclidean distance.
   b. K-means clustering divides all the data into K clusters according the distance between the cluster center and the data point. K-means is useful when the data is distribute evenly. Thus, we can have the result we want.
5.
   a. Kernel K-means can help in this dataset as it makes K-means more flexible. With the help of kernel function, we can project the dataset into another domain which the features of data can be shown straight forward.
6.
   a. Mixture models can perform a better result than k-means.
   b. With the help of GMM, we can make a linear combination of Gaussian distribution. It is obvious that the datapoints follow the Gaussian distribution. Moreover, they have overlapping areas, where K-means performs poorly.
7.
   a. Why: We need feature selection as commonly the number of features is too large to compute. In order to lower the dimensions of features, we have to select the important features and abandon the useless features.
   b. Method 1: PCA
      i. PCA is a visualization method that projects data from a higher dimension to a lower dimension, for example from 2D to 1D. We can choose the appropriate

value of PCA by cross-validation.

c. Method 2: K-NN
   i. K-NN is a non-probabilistic method for feature selection. It finds the nearest K points to the center and cluster them into a group. With the increase of K, small classes will disappear, only the significant classes are remain. Thus, we can transfer multiple features into one feature. We can choose the appropriate value of K by cross-validation.

Introduction

08 November 2022 16:05

Note created by Zak, more information

https://github.com/Zak3225/Glasgow_COMPSCI5100_ML-AI/