面为

$$0.2x_1 + 0.4x_3 - 0.4 = 0$$

4.7.3 LDA 算法

线性判别式分析（Linear discriminant analysis），又称为 Fisher 线性判别。

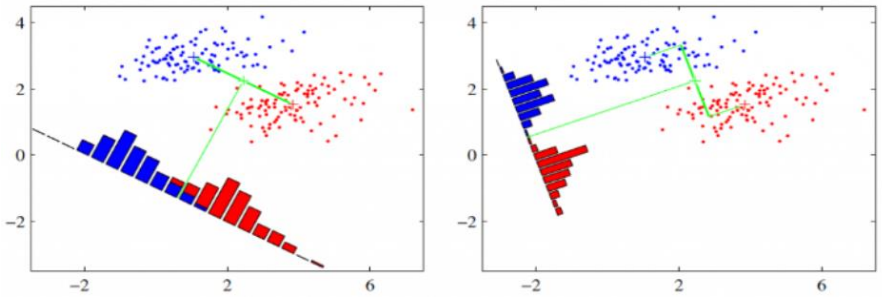


图 39 投影

通过调整权重向量组件，可选择一个投影方向，最大化分离类别。

设有 K 类样本，每一类的样本个数为 $N_1, N_2, \cdots N_k$，即 $\boldsymbol{x}_1^1 \boldsymbol{x}_1^2 \cdots \boldsymbol{x}_1^{N_1}$ 对应第一类，$\boldsymbol{x}_k^1 \boldsymbol{x}_k^2 \cdots \boldsymbol{x}_k^{N_k}$ 对应第 K 类。

设 $\tilde{\boldsymbol{x}}_i^j$ 为 $\boldsymbol{x}_i^j$ 变换后的样本，则根据 PCA 部分投影知识，可知

$$\tilde{\boldsymbol{x}}_i^j = \langle \boldsymbol{x}, \boldsymbol{\mu} \rangle = \left(\boldsymbol{x}^T \boldsymbol{\mu} \right) \boldsymbol{\mu}$$

因此第 K 类之间的样本方差为：

$$
\begin{aligned}
\boldsymbol{S}_k &= \sum_{\tilde{\boldsymbol{x}}\in D_k} (\tilde{\boldsymbol{x}}-\tilde{\boldsymbol{m}})^T(\tilde{\boldsymbol{x}}-\tilde{\boldsymbol{m}}) \\
&= \sum_{\boldsymbol{x}\in D_k} \left[\left(\boldsymbol{x}^T\boldsymbol{u}\right)\boldsymbol{u}^T - \left(\boldsymbol{m}^T\boldsymbol{u}\right)\boldsymbol{u}^T\right]\left[\left(\boldsymbol{x}^T\boldsymbol{u}\right)\boldsymbol{u} \quad -\left(\boldsymbol{m}^T\boldsymbol{u}\right)\boldsymbol{u}\right] \\
&= \sum_{\boldsymbol{x}\in D_k} \left[\left(\boldsymbol{x}^T\boldsymbol{u}\right)^2 - 2\left(\boldsymbol{x}^T\boldsymbol{u}\right)\left(\boldsymbol{m}^T\boldsymbol{u}\right) + \left(\boldsymbol{m}^T\boldsymbol{u}\right)^2\right]\boldsymbol{u}^T\boldsymbol{u}
\end{aligned}
$$

设$\boldsymbol{u}^T\boldsymbol{u}=a$

$$
\begin{aligned}
\Rightarrow \frac{\boldsymbol{S}_k}{N_k} &= \left(\frac{\sum_{\boldsymbol{x}\in D_k}\left(\boldsymbol{x}^T\boldsymbol{u}\right)^2}{N_k} - 2\frac{\sum_{\boldsymbol{x}\in D_k}\boldsymbol{x}^T\left(\boldsymbol{u}\boldsymbol{m}^T\boldsymbol{u}\right)}{N_k} + \frac{\sum_{\boldsymbol{x}\in D_k}\left(\boldsymbol{m}^T\boldsymbol{u}\right)^2}{N_k}\right)a \\
&= \left(\frac{\sum \boldsymbol{u}^T\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{u}}{N_k} - 2\frac{\sum \boldsymbol{x}^T}{N_k}\boldsymbol{u}\boldsymbol{m}^T\boldsymbol{u} + \left(\boldsymbol{m}^T\boldsymbol{u}\right)^2\right)a \\
&= \left(\boldsymbol{u}^T\frac{\boldsymbol{x}\boldsymbol{x}^T}{N_k}\boldsymbol{u} - \left(\boldsymbol{m}^T\boldsymbol{u}\right)^2\right)a \\
&= \left(\boldsymbol{u}^T\frac{\sum \boldsymbol{x}\boldsymbol{x}^T}{N_k}\boldsymbol{u} - \boldsymbol{u}^T\boldsymbol{m}\boldsymbol{m}^T\boldsymbol{u}\right)a \\
&= \left(\boldsymbol{u}^T\left(\frac{\sum \boldsymbol{x}\boldsymbol{x}^T}{N_k} - \boldsymbol{m}\boldsymbol{m}^T\right)\boldsymbol{u}\right)a
\end{aligned}
$$

均值

总体样本方差如下：

$$
\begin{aligned}
\sum_k \frac{\boldsymbol{S}_k}{N_k} &= \sum_k \boldsymbol{u}^T\left(\frac{\sum \boldsymbol{x}\boldsymbol{x}^T}{N_k} - \boldsymbol{m}_k\boldsymbol{m}_k^T\right)\boldsymbol{u}a \\
&= \left(\boldsymbol{u}^T\sum_k\frac{\sum \boldsymbol{x}\boldsymbol{x}^T}{N_k} - \boldsymbol{m}_k\boldsymbol{m}_k^T\boldsymbol{u}\right)a \\
&= \boldsymbol{u}^T\boldsymbol{S}_w\boldsymbol{u}a
\end{aligned}
$$

样本间的均值距离为：

$$
\begin{aligned}
\boldsymbol{S}_{ij} &= \left(\tilde{\boldsymbol{m}}_i - \tilde{\boldsymbol{m}}_j\right)^T\left(\tilde{\boldsymbol{m}}_i - \tilde{\boldsymbol{m}}_j\right) \\
&= \boldsymbol{u}^T\left(\boldsymbol{m}_i - \boldsymbol{m}_j\right)^T\left(\boldsymbol{m}_i - \boldsymbol{m}_j\right)\boldsymbol{u}\boldsymbol{u}^T\boldsymbol{u} \\
&= \boldsymbol{u}^T\left(\boldsymbol{m}_i - \boldsymbol{m}_j\right)^T\left(\boldsymbol{m}_i - \boldsymbol{m}_j\right)\boldsymbol{u}a
\end{aligned}
$$

则所有样本之间的均值距离为：

$$\sum_{\substack{i,j\\ i\neq j}} \boldsymbol{S}_{ij} = \boldsymbol{u}^T \sum_{\substack{i,j\\ i\neq j}} \left(\boldsymbol{m}_i - \boldsymbol{m}_j\right)^T \left(\boldsymbol{m}_i - \boldsymbol{m}_j\right) \boldsymbol{u}a$$
$$= \boldsymbol{u}^T \boldsymbol{S}_b \boldsymbol{u}a$$

为了使得样本内的方差最小，且样本间的均值距离越大，则有下面的优化函数：

$$\max J(\boldsymbol{u}) = \frac{\boldsymbol{u}^T \boldsymbol{S}_b \boldsymbol{u}a}{\boldsymbol{u}^T \boldsymbol{S}_w \boldsymbol{u}a} = \frac{\boldsymbol{u}^T \boldsymbol{S}_b \boldsymbol{u}}{\boldsymbol{u}^T \boldsymbol{S}_w \boldsymbol{u}}$$

现假设 $\boldsymbol{u}^T \boldsymbol{S}_w \boldsymbol{u}=1$，则等价于

$$\max \boldsymbol{u}^T \boldsymbol{S}_b \boldsymbol{u}$$
$$s.t. \boldsymbol{u}^T \boldsymbol{S}_w \boldsymbol{u} = 1$$

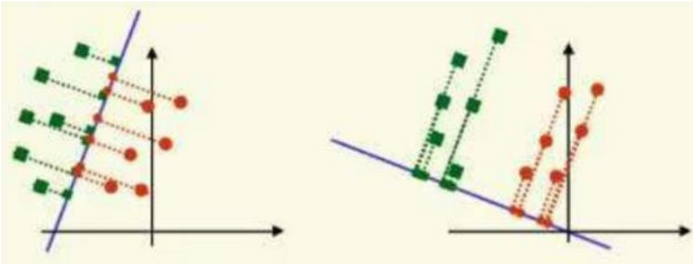
构造拉格朗日函数：

$$L(\boldsymbol{u}, \lambda) = \boldsymbol{u}^T \boldsymbol{S}_b \boldsymbol{u} + \lambda\left(1 - \boldsymbol{u}^T \boldsymbol{S}_w \boldsymbol{u}\right)$$

求导得

$$\frac{\partial L}{\partial \boldsymbol{u}} = 0 \Rightarrow \boldsymbol{S}_b \boldsymbol{u} = \lambda \boldsymbol{S}_w \boldsymbol{u} \Rightarrow \boldsymbol{S}_w^{-1} \boldsymbol{S}_b \boldsymbol{u} = \lambda \boldsymbol{u}$$

从而变成了对 $\boldsymbol{S}_w^{-1} \boldsymbol{S}_b$ 求特征值。



PCA 最小重构误差，使得投影后的值和原来的值尽量接近，属于非监督学习；

LDA 最大化类间距离，最小化类内距离，使得投影后的不同类别的样本分的更开，属于监督学习。

4.7.4 决策树

4.7.4.1 基本流程

决策树学习的目的是为了产生一棵泛化能力强，即处理未见示例能力强的决策树，本质上是一种特征与结果的相关度。其基本流程遵循简单且直观的"分而治之"(divide-and-conquer)策略，如图 40 所示。

输入： 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
　　属性集 $A = \{a_1, a_2, \ldots, a_d\}$.
过程： 函数 TreeGenerate(D, A)
1: 生成结点 node;
2: **if** D 中样本全属于同一类别 C **then**
3: 　将 node 标记为 C 类叶结点; **return**
4: **end if**
5: **if** $A = \varnothing$ **OR** D 中样本在 A 上取值相同 **then**
6: 　将 node 标记为叶结点, 其类别标记为 D 中样本数最多的类; **return**
7: **end if**
8: 从 A 中选择最优划分属性 a_*;
9: **for** a_* 的每一个值 a_*^v **do**
10: 　为 node 生成一个分支; 令 D_v 表示 D 中在 a_* 上取值为 a_*^v 的样本子集;
11: 　**if** D_v 为空 **then**
12: 　　将分支结点标记为叶结点, 其类别标记为 D 中样本最多的类; **return**
13: 　**else**
14: 　　以 TreeGenerate(D_v, $A \setminus \{a_*\}$)为分支结点
15: 　**end if**
16: **end for**
输出： 以 node 为根结点的一棵决策树

图 40 决策树学习基本算法

显然，决策树的生成是一个递归过程。在决策树基本算法中，有三种情形会导致递归返回：(1)当前结点包含的样本全属于同一类别，无需划分；(2)当前属性集为空，或是所有样本在所有属性上取值相同，无法划分；(3)当前结点包含的样本集合为空，不能划分。

在第(2)种情形下，我们把当前结点标记为叶结点，并将其类别设定为该结点所含样本最多的类别；在第(3)种情形下，同样把当前结点标记为叶结点，但将其类别设定为其父结点所含样本最多的类别。注意这两种情形的处理实质不同：情形(2)是在利用当前结点的后验分布，而情形(3)则是把父结点的样本分布作为当前结点的先验分布。

4.7.4.2 划分选择

（1） 信息增益

为了便于说明，先给出熵与条件熵的定义.

在信息论与概率统计中，熵（entropy）是表示随机变量不确定性的度量. 设 X 是一个取有限个值的离散随机变量，其概率分布为

$$P(X=x_i)=p_i, \quad i=1,2,\cdots,n$$

则随机变量 X 的熵定义为

$$H(X)=-\sum_{i=1}^{n} p_i \log p_i \tag{5.1}$$

在式 (5.1) 中，若 $p_i=0$，则定义 $0\log 0=0$. 通常，式 (5.1) 中的对数以 2 为底或以 e 为底（自然对数），这时熵的单位分别称作比特（bit）或纳特（nat）. 由定义可知，熵只依赖于 X 的分布，而与 X 的取值无关，所以也可将 X 的熵记作 $H(p)$，即

$$H(p)=-\sum_{i=1}^{n} p_i \log p_i \tag{5.2}$$

熵越大，随机变量的不确定性就越大. 从定义可验证

$$0 \leqslant H(p) \leqslant \log n \tag{5.3}$$

当随机变量只取两个值，例如 1，0 时，即 X 的分布为

$$P(X=1)=p, \quad P(X=0)=1-p, \quad 0 \leqslant p \leqslant 1$$

熵为

$$H(p)=-p\log_2 p-(1-p)\log_2(1-p) \tag{5.4}$$

设有随机变量 (X,Y)，其联合概率分布为

$$P(X=x_i,Y=y_j)=p_{ij}, \quad i=1,2,\cdots,n; \quad j=1,2,\cdots,m$$

条件熵 $H(Y|X)$ 表示在已知随机变量 X 的条件下随机变量 Y 的不确定性. 随机变量 X 给定的条件下随机变量 Y 的条件熵（conditional entropy）$H(Y|X)$，定义为 X 给定条件下 Y 的条件概率分布的熵对 X 的数学期望

$$H(Y|X)=\sum_{i=1}^{n} p_i H(Y|X=x_i) \tag{5.5}$$

这里，$p_i=P(X=x_i)$，$i=1,2,\cdots,n$.

当熵和条件熵中的概率由数据估计（特别是极大似然估计）得到时，所对应的熵与条件熵分别称为经验熵（empirical entropy）和经验条件熵（empirical conditional entropy）. 此时，如果有 0 概率，令 $0\log 0=0$.

信息增益（information gain）表示得知特征 X 的信息而使得类 Y 的信息的不确定性减少的程度.

“信息熵”(information entropy)是度量样本集合纯度最常用的一种指标. 假定当前样本集合 D 中第 k 类样本所占的比例为 p_k $(k = 1, 2, \ldots, |\mathcal{Y}|)$, 则 D 的信息熵定义为

$$\mathrm{Ent}(D) = -\sum_{k=1}^{|\mathcal{Y}|} p_k \log_2 p_k \ . \tag{4.1}$$

$\mathrm{Ent}(D)$ 的值越小, 则 D 的纯度越高.

假定离散属性 a 有 V 个可能的取值 $\{a^1, a^2, \ldots, a^V\}$, 若使用 a 来对样本集 D 进行划分, 则会产生 V 个分支结点, 其中第 v 个分支结点包含了 D 中所有在属性 a 上取值为 a^v 的样本, 记为 D^v. 我们可根据式(4.1) 计算出 D^v 的信息熵, 再考虑到不同的分支结点所包含的样本数不同, 给分支结点赋予权重 $|D^v|/|D|$, 即样本数越多的分支结点的影响越大, 于是可计算出用属性 a 对样本集 D 进行划分所获得的“信息增益”(information gain)

$$\mathrm{Gain}(D, a) = \mathrm{Ent}(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|} \mathrm{Ent}(D^v) \ . \tag{4.2}$$

一般而言, 信息增益越大, 则意味着使用属性 a 来进行划分所获得的“纯度提升”越大. 因此, 我们可用信息增益来进行决策树的划分属性选择, 即在图 4.2 算法第 8 行选择属性 $a_* = \arg\max_{a \in A} \mathrm{Gain}(D, a)$. 著名的 ID3 决策树学习算法 [Quinlan, 1986] 就是以信息增益为准则来选择划分属性.

4.8 无监督学习算法

4.8.1 PCA

可见 1.12.3。

4.8.2 k-均值聚类

给定样本集 $D=\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$，"k 均值"(k-means)算法针对聚类所得簇划分 $\mathcal{C}=\{C_1, C_2, \ldots, C_k\}$ 最小化平方误差

$$E=\sum_{i=1}^{k}\sum_{\boldsymbol{x}\in C_i}\left\|\boldsymbol{x}-\boldsymbol{\mu}_i\right\|_2^2$$

其中 $\boldsymbol{\mu}_i=\frac{1}{|C_i|}\sum_{\boldsymbol{x}\in C_i}\boldsymbol{x}$ 是簇 C_i 的均值向量。

输入: 样本集 $D=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_m\}$;
聚类簇数 k.
过程:
从 D 中随机选择 k 个样本作为初始均值向量 $\{\boldsymbol{\mu}_1,\boldsymbol{\mu}_2,\ldots,\boldsymbol{\mu}_k\}$
repeat
  令 $C_i=\varnothing\ (1\leqslant i\leqslant k)$
  for $j=1,2,\ldots,m$ **do**
    计算样本 $\boldsymbol{x}_j$ 与各均值向量 $\boldsymbol{\mu}_i\ (1\leqslant i\leqslant k)$ 的距离: $d_{ji}=||\boldsymbol{x}_j-\boldsymbol{\mu}_i||_2$;
    根据距离最近的均值向量确定 $\boldsymbol{x}_j$ 的簇标记: $\lambda_j=\arg\min_{i\in\{1,2,\ldots,k\}}d_{ji}$;
    将样本 $\boldsymbol{x}_j$ 划入相应的簇: $C_{\lambda_j}=C_{\lambda_j}\bigcup\{\boldsymbol{x}_j\}$;
  end for
  for $i=1,2,\ldots,k$ **do**
    计算新均值向量: $\boldsymbol{\mu}_i'=\frac{1}{|C_i|}\sum_{\boldsymbol{x}\in C_i}\boldsymbol{x}$;
    if $\boldsymbol{\mu}_i'\neq\boldsymbol{\mu}_i$ **then**
      将当前均值向量 $\boldsymbol{\mu}_i$ 更新为 $\boldsymbol{\mu}_i'$
    else
      保持当前均值向量不变
    end if
  end for
until 当前均值向量均未更新
输出: 簇划分 $\mathcal{C}=\{C_1,C_2,\ldots,C_k\}$

图 41 k 均值算法

4.9 随机梯度下降

随机梯度下降（stochastic gradient descent, SGD）。随机梯度下降是第 3.3 节介绍的梯度下降算法的一个扩展。

机器学习算法中的代价函数通常可以分解成每个样本的代价函数的总和。例如，训练数据的负条件对数似然可以写成

$$J(\boldsymbol{\theta})=\mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim\hat{p}_{\text{data}}}L(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\theta})=\frac{1}{m}\sum_{i=1}^{m}L\left(\boldsymbol{x}^{(i)},\boldsymbol{y}^{(i)},\boldsymbol{\theta}\right)$$

其中 L 是每个样本的损失 $L\left(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}\right)=-\log p(y \mid \boldsymbol{x} ; \boldsymbol{\theta})$。

对于这些相加的代价函数，梯度下降需要计算

$$\nabla_{\theta} J(\boldsymbol{\theta})=\frac{1}{m} \sum_{i=1}^{m} \nabla_{\theta} L\left(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}\right)$$

这个运算的计算代价是 $O(m)$，随着样本规模的增加，计算一步梯度也会消耗相当长的时间。

随机梯度下降的核心是，梯度是期望。期望可使用小规模的样本近似估计。具体而言，在算法的每一步，我们从训练集中均匀抽出一小批量（minibatch）样本 $\mathbb{B}=\left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m')}\right\}$。小批量的数目 m' 通常是一个相对较小的数，从一到几百。重要的是，当训练集大小 m 增长时，m' 通常是固定的。我们可能在拟合几十亿的样本时，每次更新计算只用到几百个样本。

梯度的估计可以表示成

$$\boldsymbol{g}=\frac{1}{m'} \nabla_{\theta} \sum_{i=1}^{m'} L\left(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}\right)$$

使用来自小批量 $\mathbb{B}$ 的样本，然后，随机梯度下降算法使用如下的梯度下降估计：

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}-\alpha \boldsymbol{g}$$

作业：

a、k-均值和 PCA 练习

https://github.com/hanbaoan123/ml-coursera-python-assignments/blob/master/Exercise7/exercise7.ipynb

b、阅读聚类算法

https://scikit-learn.org/stable/modules/clustering.html#overview-of-clustering-methods

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters or distance threshold	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters or distance threshold, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes	Distances between nearest points
OPTICS	minimum cluster membership	Very large `n_samples`, large `n_clusters`	Non-flat geometry, uneven cluster sizes, variable cluster density	Distances between points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers
Birch	branching factor, threshold, optional global clusterer.	Large `n_clusters` and `n_samples`	Large dataset, outlier removal, data reduction.	Euclidean distance between points

c、阅读梯度下降

https://www.benfrederickson.com/numerical-optimization/

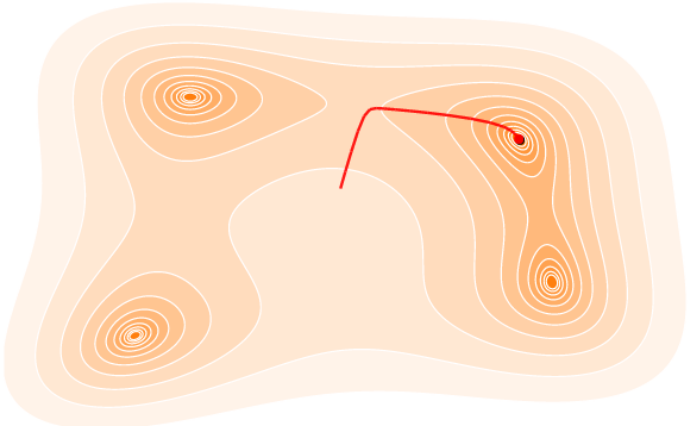
Gradient Descent

One possible direction to go is to figure out what the gradient $\nabla F(X_n)$ is at the current point, and take a step down the gradient towards the minimum. The gradient can be calculated by symbolically differentiating the loss function, or by using automatic differentiation like Torch and TensorFlow does. Using a fixed step size α this means updating the current point X_n by:

$$X_{n+1} = X_n - \alpha \nabla F(X_n)$$

This leads to taking the path of the steepest descent, which looks vaguely like a ball rolling down a hill towards one of the local minima:

$f(x,y)$ $= (x^2 + y - 11)^2 + (x + y^2 - 7)^2$



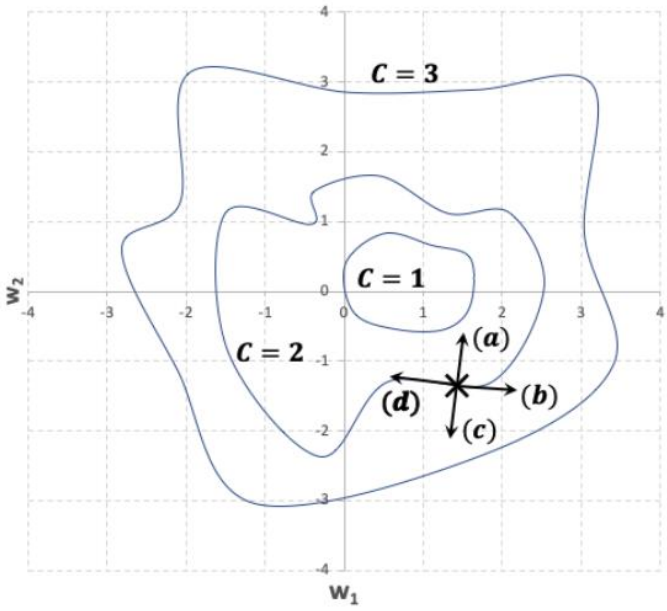
d、比较随机梯度下降与批梯度下降。

随机梯度下降在每轮迭代中，随机优化某一条训练数据上的损失函数，这样每一轮参数的更新速度大大加快，但可能无法达到全局最优。

批梯度下降每次计算一小部分训练数据的损失函数，大大减小收敛所需要的迭代次数，同时可以使收敛到的结果更加接近梯度下降的效果。

e.假设我们的模型参数$\mathbf{w} = [w_1, w_2]^\intercal$，优化目标为最小化代价函数 C 的值。下图曲线表示模型参数空间中 C 的等高线。C=2 表示的等高线上所有的w_1, w_2组合都对应代价函数值 2。假设当参数$\mathbf{W}$从曲线 C=3 向曲线 C=2 移动时，C 值单调递减。当前的$\mathbf{W}$在图中“x”所示位置。试从(a),(b),(c),(d)中选出正确的梯度下降更新方向。

答案为（a）。



4.10 构建机器学习算法

几乎所有的深度学习算法都可以被描述为一个相当简单的配方：特定的数据集、代价函数、优化过程和模型。

4.11 促使深度学习发展的挑战

4.11.1 维数灾难

许多传统机器学习算法假设在一个新点的输出应大致和最接近的训练点输出相同。而在高维空间中，参数配置数目远大于样本数目，新点附近少有或没有样本，对学习造成困难。

4.11.2 局部不变性和平滑正则性

机器学习算法广泛使用局部不变性先验，即学习的函数在小区域内不会发生很大的变化。但仅依靠该先验难以有效表示复杂的函数。

机器学习方法往往针对特定问题会提出很强的假设，因此泛化能力较差。

深度学习假设数据由因素或特征组合产生，这些温和的通用假设允许了样本数目和可区分区间数目之间的指数增益，有效解决维数灾难带来的挑战。

4.11.3 流形学习

流形学习假设大部分区域都是无效的输入，有意义的输入只分布在包含少量数据点的子集构成的一组流形中。

我们认为，在处理图像、文本、声音时，流形假设至少是近似对的。