

# machine learning 笔记

徐世桐

## 1 基础定义

**二元分类**: 输出分类个数为 2

**多元分类**: 输出分类个数不限

*one - versus - the - rest* OvR: 计算属于每一分类的可能性, 取可能性最大的分类为输出分类

*one - versus - one* OvO: 对所有分类两两使用二元分类, 每一分类器训练只需一部分数据

**multilabel 多标签分类**: 目标检测, 对一图像中的物体加 label

**multioutput 多类分类**: 多标签分类, 每一标签可包含多种信息

**learning schedule**: 根据迭代次数更新学习率

**early stopping**: 提早结束训练

对于每一 epoch, 当验证集 MSE 值增高时, 证明开始 overfit, 停止训练

即在 epoch-error 图中泛化误差最低时停止训练

**semi-supervised learning**: 部分样本有对应标签

**weakly-supervised learning**: 对样本标记包含的物体, 而不标注对应目标的具体位置

**non parametric model**: 无法用有限的 distribution parameter 代表的模型, 如 Nearest neighbour  
在训练中使用正则化代价函数, 训练结束后测试中代价函数不使用正则化项

**curse of dimentionality**

令  $d$  为特征数,  $e$  为一特征覆盖范围

为了在  $n$  总样本中覆盖  $k$  个样本, 平均需要  $e$  满足  $e^d = \frac{n}{k}$

随  $d$  升高,  $e$  值接近 1。即每一特征需覆盖大部分取值范围使得  $k$  样本每个参数能同时被  $d$  个特征覆盖

其余  $n-k$  样本存在特征取值范围的边界上, 即距离  $k$  样本距离几乎相同远

## 2 数学计算

$$MSE = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \bar{x})^2$$

**rigid regression**: 回归方法,  $J(\theta) = MSE(\theta) + \frac{\alpha}{2} \sum_i \theta_i^2$

降低所有权重值

**lasso regression**: 回归方法,  $J(\theta) = MSE(\theta) + \alpha \sum_i |\theta_i|$

降低不重要的权重值

**elastic net**: 回归方法,  $J(\theta) = MSE(\theta) + \gamma \alpha \sum_i |\theta_i| + (1 - \gamma) \frac{\alpha}{2} \sum_i \theta_i^2$

**Normal Equation**:  $\hat{\theta} = (X^T X)^{-1} X^T y$

直接得到权重  $\hat{\theta}$ , 适用于仅有一个输出值的模型

$X$  为 (批量大小, 参数个数) 输入矩阵,  $y$  为 (批量大小,) 向量  
当  $X^T X$  无逆矩阵时, 用 pseudo inverse  $\hat{\theta} = X^+ y$

**pseudo inverse:**

对矩阵  $X = USV^T$ , pseudo inverse  $X^+ = VS^+U^T$ 。  $S^+$  求法:

1. 对所有  $S$  元素, 接近 0 的值赋为 0
2. 对所有非零元素取倒数
3. 取矩阵转置, 得到  $S^+$

**log loss:** 代价函数

$$J(\theta) = -\frac{1}{|B|} \sum_{i=1}^{|B|} [y^{(i)} \log(\hat{p}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{p}^{(i)})]$$

标签值  $y^{(i)}$  为离散 1/0 值, 计算值  $\hat{p}^{(i)} \in [0, 1]$

微分: \*\* 推导 \*\*

$$\frac{dJ(\theta)}{d\theta_j} = \frac{1}{|B|} \sum_{i=1}^{|B|} (\hat{p}^{(i)} - y^{(i)}) x_j^{(i)}$$

**Hinge loss:** 代价函数

$$HingeLoss(y, \hat{y}) = \max(0, 1 - y * \hat{y})$$

应用于 SVM,  $y \in \{0, 1\}$ ,  $\hat{y} \in \mathbb{R}$

代表当预测值  $\hat{y}$  和  $y$  同号,  $\hat{y} \geq 1$ , 则预测值和标签匹配, 代价 = 0。否则  $y * \hat{y} < 1$ 。代价值上升

**Gaussian Radial Basis Function RBF:** 一种 similarity function

$$\phi_\gamma(x, l) = \exp(-\gamma \|x - l\|^2)$$

$l$  为 landmark, 即  $\phi_\gamma$  由一样本  $x_i$  和一 landmark 的距离得来

**Lagrange multipliers method 拉格朗日乘数法**

将 有前提的多项式求最值 问题转化为 无前提多项式最值问题

定义:

对输入向量  $W$ ,  $g(W) \geq 0$  为 constrain。目标为在满足  $g(W) \geq 0$  的前提下取  $f(W)$  最值

Lagrange function  $\mathcal{L}(W, \alpha) = f(W) - \alpha(g(W))$

$\alpha$  为需要求解的变量之一, 参与最终计算  $W$  的值。

当有多个 constrain  $g^{(i)}(W)$  时,  $\vec{\alpha}$  为向量, 求偏导对每一  $\vec{\alpha}^{(i)}$  求导

只有当  $\alpha \geq 0$  或每一  $\vec{\alpha}^{(i)} \geq 0$ , 结果才有效

$\vec{\alpha}^{(i)} = 0$  代表对应的 constrain  $g^{(i)}(W)$  为一个 support vector

计算:

$$\text{对每一 } W \text{ 的元素和 } \alpha \text{ 取偏导, 即向量 } \begin{bmatrix} \frac{d\mathcal{L}(W, \alpha)}{dw_1} \\ \frac{d\mathcal{L}(W, \alpha)}{dw_n} \\ \dots \\ \frac{d\mathcal{L}(W, \alpha)}{dw_n} \\ \frac{d\mathcal{L}(W, \alpha)}{d\alpha} \end{bmatrix}, \text{ 计算向量} = \vec{0} \text{ 时的 } W, \alpha \text{ 取值}$$

**Distance Metrics**

Manhattan distance(L1-norm):  $d(x^{(i)}, x^{(j)}) = \sum_k |x_k^{(i)} - x_k^{(j)}|$

Euclidean distance(L2-norm):  $d(x^{(i)}, x^{(j)}) = \sqrt{\sum_k (x_k^{(i)} - x_k^{(j)})^2}$

Chebyshev distance(Linf-norm):  $d(x^{(i)}, x^{(j)}) = \max_k |x_k^{(i)} - x_k^{(j)}|$

**information entropy**

对单一一组数据  $X = [x_1, \dots, x_n]$ ,  $x_i$  在  $X$  中出现百分比为  $p(x_i)$

$X$  的数据熵  $H(X) = -\sum_i p(x_i) \log_2(p(x_i))$

当  $x_i$  为 continuous, 不为离散值时,  $X$  即一分部。此时  $H(X) = - \int_x p(x) \log_2(p(x)) dx$

### 3 分类模型

#### classification:

binary classification: 拥有 2 类标签

Multi-class classification: 拥有多类标签

Multi-label classification: 单个样本可以属于多个标签

#### logistic regression:

判断输入符合每一输出类别的可能性,

分类:

Simple regression: 单个样本变量个数为 1

Multiple regression: 样本变量个数  $> 1$

Multivariate regression: 单个样本对应标签个数  $> 1$

前向计算:

$$1. \hat{p} = \sigma(\theta^T x + b)$$

$$2. \hat{y} = 1(\text{if } \hat{p} \geq 0.5) \\ = 0(\text{if } \hat{p} < 0.5)$$

代价函数为 log loss

#### SVM

找到分界, 分离多种数据

support vector: 最靠近分界线的样本

hard margin classification 硬性分类: 限制数据必须被分界隔开, 同一类数据不可同时出现在分界 2 端

soft margin classification: 与硬性分类相反, 避免被 outlier 离群值影响

前向计算:  $\hat{p} = f(x_1, x_2, \dots)$ , 其余同 logistic regression

区别:  $f$  可为 polynomial, 非线性函数。可使用 kernel trick

线性分类训练:  $\hat{p} = W^T x + b$ ,  $W$  为参数向量

硬性分类:

$\|W\|_2$  代表线性函数斜率

最小化  $\frac{1}{2} W^T W$ , 使得分界平面的斜率最小, 最大化分界线和两种数据的距离

前提: 对每一样本  $i$ ,  $1.y^{(i)}\hat{p}^{(i)} \geq 1$ , 即标签和计算结果相同

求解: 1. 直接解以上带前提的不等式

' 当样本数高于参数数量时使用, 由于 dual form 的复杂度为  $O(|S|^2) - O(|S|^3)$ , 直接解复杂度为  $O(|S|)$ '

2. 使用拉格朗日乘数法得到 dual form, 其中  $\vec{\alpha}$  为向量。  $\mathbf{x}^{(i)}$  为第  $i$  样本的特征值向量  $\mathcal{L} = \frac{1}{2} W^T W - \sum_{i=1}^{|B|} \vec{\alpha}^{(i)} (y^{(i)} \hat{p}^{(i)} - 1)$

使偏导向量为  $\vec{0}$ , 得到 2.  $W = \sum_{i=1}^m \vec{\alpha}_i y^{(i)} \mathbf{x}^{(i)}$ , 3.  $\sum_{i=1}^m \vec{\alpha}_i y^{(i)} = 0$

$$\text{带入得 } \mathcal{L}(W, \vec{\alpha}) = \frac{1}{2} \sum_{i=1}^{|B|} \sum_{j=1}^{|B|} \vec{\alpha}_i \vec{\alpha}_j y^{(i)} y^{(j)} \mathbf{x}^{(i)T} \mathbf{x}^{(j)} - \sum_{i=1}^{|B|} \vec{\alpha}^{(i)} \\ = \frac{1}{2} \vec{\alpha}^T (\mathbf{x} * \mathbf{y})(\mathbf{x} * \mathbf{y})^T \vec{\alpha} - \sum_{i=1}^{|B|} \vec{\alpha}^{(i)}$$

其中  $(\mathbf{x} * \mathbf{y})$  为广播乘法 将训练集矩阵每一样本乘以对应标签值,  $\mathbf{y}$  为标签列向量

使用 QP solver 得到使  $\mathcal{L}(W, \vec{\alpha})$  最小,  $\vec{\alpha}^{(i)} \geq 0$  的向量  $\vec{\alpha}$

解  $W$ : 由  $\vec{\alpha}$  带入 2. 式计算,  $\vec{\alpha}$  已被 clamp, 见经验 2.

解  $b$ : 由于所有 support vector  $\mathbf{x}^{(i)}$  满足 1. 式, 则对所有 support vector 计算  $b$  取平均值

$$b = E_{\vec{\alpha}^{(i)} \geq 0} (y^{(i)} - W^T \mathbf{x}^{(i)})$$

3. 直接进行梯度下降, 代价函数  $J(W, b) = \frac{1}{2} W^T W + \text{const} \sum_i \text{HingeLoss}(y^{(i)}, \hat{p}^{(i)})$

软性分类:

$$\text{最小化 } \frac{1}{2} W^T W + C \sum_{i=1}^{|B|} \zeta_i$$

$\zeta_i$  定义第  $i$  样本被忽视为误差样本的可能性,  $C$  定义忽视率相对斜率的权重

前提: 对每一样本  $i$ ,  $y^{(i)} \hat{p}^{(i)} \geq 1 - \zeta^{(i)}$

非线性分类方法:

- 使用 polynomial 做  $f$

必须使用拉格朗日乘数法求解, 目的为对  $\phi(x)$  得到线性权重和偏差, 求解使用 dual form, 其中包含  $\phi(a)^T \cdot \phi(b)$  项即可使用 kernel method

权重  $W$  公式不再适用, 由于结果不为线性

$$\text{偏差 } b = \sum_{\vec{\alpha}^{(i)} \geq 0} y^{(i)} - \sum_{\vec{\alpha}^{(j)} \geq 0} \vec{\alpha}^{(j)} * y^{(j)} * K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

- 使用 similarity function:

选择多个 landmark  $\mathcal{L} = l_1, l_2, \dots, l_n$ , 对每一样本  $x_i$  计算其和每一  $l_j$  的  $\phi_\gamma$  值  $\phi_\gamma(x_i, l_j)$

每个样本用新的向量  $x'_i = \begin{bmatrix} \phi_\gamma(x_i, l_1) \\ \phi_\gamma(x_i, l_2) \\ \dots \\ \phi_\gamma(x_i, l_n) \end{bmatrix}$  表示。新的向量组成训练集, 进行 SVM 训练

kernel:

定义: 能够从输入向量  $a, b$ , 不通过计算  $\phi(a), \phi(b)$  直接得到点乘结果  $\langle \phi(a), \phi(b) \rangle$  的函数

例: \*\* 是否通过取 linear 为 phi 得到 kernel 函数 \*\*

linear:  $f(a, b) = a^T b$

polynomial:  $f(a, b) = (\gamma a^T b + r)^d$

poly 的  $\phi(x)$  为对向量  $x$  每一元素进行 poly 运算, 结果向量元素数不变

Gaussian RBF:  $f(a, b) = \exp(-\gamma \|a - b\|^2)$

Sigmoid:  $f(a, b) = \tanh(\gamma a^T b + r)$

经验总结:

1. QP solver 中需限定  $\sum_{i=1}^m \vec{\alpha}_i y^{(i)} = 0$ , 否则得出  $\hat{\alpha}$  不遵循此等式

2. 当样本有重叠, 仍可使用拉格朗日乘数法, 异常样本被分入错误类别。

此时  $\vec{\alpha}$  包含负值, 对应的样本在计算权重 偏差时被忽略, 即需 clamp 使  $\vec{\alpha} \geq 0$

若不进行 clamp, 得到的分界仅有略微差别, 不会造成大幅误差。(在线性 非线性分类都有验证)

3. 梯度下降直接得到最优  $W, b$ , 无法通过梯度下降得到  $\vec{\alpha}$  由于梯度下降忽略限制条件

4. QP solver 需要  $(\mathbf{x} * y)(\mathbf{x} * y)^T$  为 positive definite, 计算时加上对角矩阵  $\text{diag}(\epsilon)$  即可,  $\epsilon$  多

取  $10^{-4}$

否则迭代解 QP 时出现 KKT condition not met 或 positive definite 条件不满足

条件不满足时中断得到的  $\vec{\alpha}$  无法作为有效结果参与后续计算权重和偏差

优先将此矩阵转为 float64 类型, 否则需要  $\epsilon$  较大才能保证 positive definite

## 4 KNN

lazy learner: 仅在得到特征后进行计算, 得到训练集后仅仅保存训练集

定义:

得到样本集  $S$ , 每次得到需要预测的特征向量  $x$

算法:

从样本特征集中选取  $k$  个最邻近  $x$  的样本, 距离由 distance metric 计算, 返回  $k$  个样本标签中占比较大的标签

### distance weighted KNN

算法:

对  $k$  个邻近样本, 每一分配权重  $w_i$ 。

对  $k$  个样本中同一标签下的样本权重求和, 总和较高的标签作为结果

取  $w_i$  方法 1:

$$1. w_i = \frac{1}{d(x^{(i)}, x)}$$

$$2. w_i = \frac{1}{2\pi} \exp\left(-\frac{d(x^{(i)}, x)^2}{2}\right)$$

$d(x^{(i)}, x)$  来自 distance metric

优劣:

1.  $k$  值影响较小, 由于较远的样本  $w_i$  较小

2. 受 curse of dimensionality 影响。可对每一特征加权重 或 feature extraction 解决

### KNN regression

样本标签不为离散值, 而为连续值, 求 regression。

算法:

对所有可能的 feature 向量  $x$ , 取距离  $x$  最近的  $k$  个样本。

$x$  的标签值为  $k$  个样本的平均值。此值即为 regression 结果

### Locally weighted regression

distance-weighted KNN, 将  $K$  个邻近样本的距离作为权重  $w_i$ 。计算  $x$  标签  $= \sum_i w_i \cdot d(x^{(i)}, x)$

## 5 决策树

定义:

节点  $N_i$ :

节点条件: 判断样本进入哪一子节点, 叶节点没有节点条件

sample 属性  $S_i$ : 有多少样本进入  $N_i$  节点, 非满足  $N_i$  节点条件的样本个数

value 属性  $V_i = v_{i1}, \dots, v_{in}$ :  $S_i$  进入节点的样本中  $v_{ij}$  个属于第  $j$  分类

子节点仅有 2 个, 对应节点条件为 true/false 的情况

分类方式: 数据从根节点开始, 根据节点条件传向对应子节点。直到到达叶节点。叶节点中  $V$  属性中最大项即数据分类

在 imbalanced dataset 上训练效果不好

### CART algorithm 创建决策树:

根节点初始化为叶节点, 没有节点条件

对每一叶节点  $S_i$  选取一特征  $k$ , 一特征阈值  $t_k$ , 将样本集分为 2 组  $S_{true}, S_{false}$ 。

选取  $(k, t_k)$  方式：使代价函数  $J(k, t_k) = \frac{S_{true}}{S_i} G_{true} + \frac{S_{false}}{S_i} G_{false}$  最小

gini 属性  $G_i$ ：数据混杂度， $G_i = 1 - \sum_{j=1}^n (\frac{v_{ij}}{S_i})^2$

直到决策树层数达到固定上限，或对所有分组条件  $(k, t_k)$ ， $J(k, t_k) \geq G_i$

### information gain 创建决策树

允许单个节点  $N_i$  有多个子节点  $C_i = N_j$ ，(在特征为离散分类时)

选取子集方式：最大化 information gain  $IG(N_i, C_i) = H(N_i) - \sum_{N_j \in C_i} (\frac{S_j}{S_i} H(S_j))$

$H()$  为 information entropy，替换 CART 法中的 gini 属性

所有特征为实数创建决策树：

将  $S_i$  样本按照实数特征排序，随后选择特征 门槛  $(k, t_k)$ ，将数据分为 2 组，最大化 information gain

即 CART algorithm，使用不同数据混杂度函数

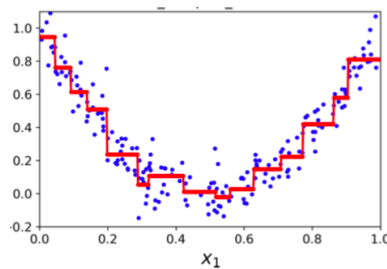
所有特征为类别参数：

选择类别特征  $t$ ， $t$  中每一类别对应一子节点。即  $t$  的值域 = 子节点数

### 避免 overfit

1. 设置决策树层数上限
2. 设置节点样本下限，若样本数量低于下限则停止继续分类
3. pruning

### 使用决策树进行 regression



输入样本，分类进不同值域

更改：

每一节点 value 值为一常数，为  $S_i$  样本的平均值。

输出值为叶节点的 value，非最大 value 对应的类别

$G_i$  为  $S_i$  样本的方差  $\frac{1}{S_i} \sum_{j=1}^{S_i} (x_i^{(j)} - \bar{x}_i)^2$

## 6 ensemble learning & 随机森林

**ensemble learning**：使用一组预测机制进行学习，预测机制可为不同算法

dropout 为一种 ensemble learning，由于丢弃神经元即改变网络结构

### random forest 随机森林：

训练方法：随机选择  $n$  个训练子集  $s_1, s_2, \dots, s_n \in S$ ，训练  $n$  个决策树  $t_1, \dots, t_n$ 。

前向计算：对  $n$  个树产生的  $n$  个分类结果，选取投票最多的一分类作为结果

训练子集选取：bagging：子集可重复选取一样本，pasting：样本不重复

out-off-bag oob 样本：当使用 bagging 选取时，平均只有  $1 - e^{-1}$  样本被选择，余下样本被称为 oob 样本

优化:

random patches 随机贴片: 对特征和训练集同时取子集进行训练

random subspace 随机子空间: 对特征取子集, 对整个总训练集进行训练

\*\* 使用 bagging 无法减少 variance, 需要对特征取子集提高效果 \*\*

extra-trees 极度随机森林: ' 使用随机  $t_k$  而不使用最小化数据混杂度的  $t_k$  '

kfeature importance 特征重要性: 对所有取  $k$  为判断条件的节点  $N_i$ , 计算加权平均值  $\sum_i (S_i \text{imprity} \text{降低百分比})$

(hypothesis) boosting: 合并多个预测机制据结果的方法

AdaBoost: 串联预测机制, 对上一预测机制遗漏的样本加更高权重, 进行训练

gradient boosting

## 7 维度下降

根据 manifold assumption, 高维空间中训练集参数点稀疏。则将数据压缩到低维

**principle component analysis PCA:**

对训练集参数矩阵取  $SVDUSV^T$

取  $V$  中前  $d$  个向量  $V' = [v_1, \dots, v_d]$ , 新训练集  $A_{compressed} = A_{origin} V'$

从 新训练集 延展回 原训练集纬度:  $A_{expand} = A_{compressed} V'^T$

**Incremental PCA:** 无需整个训练集存在内存中即可进行 SVD

**kernel PCA:** \*\*

**local linear Embedding LLE:**

对每一样本  $x^{(i)}$  寻找  $k$  个相邻样本 相邻样本 index 的集合称  $C_{x^{(i)}}$

构建  $(|S|, |S|)$  矩阵  $W$ :

每一行向量  $[W_{i1}, \dots, W_{i|S|}]$  满足  $x^{(i)} - \sum_{j \in C_{x^{(i)}}} W_{ij} x^{(j)}$

每一行向量  $W_i$  求和为 1:  $\sum_{i=1}^{|S|} W_i = 1$

由  $W$  创建新训练集:

令  $z^{(i)}$  为  $x^{(i)}$  在低维的投影

使所有  $z^{(i)}$  满足最小化  $(z^{(i)} - \sum_{j=1}^{|B|} w_{ij} z^{(j)})^2$

## 8 聚类分析

**K-mean:**

将数据分为  $k$  个 cluster, 每个 cluster 有中心点称 centroid

算法:

1. 初始化随机选择  $k$  个样本位置做 centroid, 避免得到空 cluster

当迭代过程中出现空 cluster, 从其他 cluster 中分配一随机参数点给此 cluster

2. 分配样本: 每个样本分入距离最近的 centroid 的 cluster

3. 更新 centroid: 新 centroid 为 cluster 中样本坐标平均值。

重复第 2.3. 步, 直至 centroid 不再移动, 或移动距离小于定值

vornoid diagram:

画有不同 cluster 的分界线的图

迭代:

多次随机初始化 centroid, 选择其中 inertia 最小的 centroid 取法进行训练

$$\text{inertia} = \frac{1}{|S|} \sum_x (C_x - x)^2.$$

$C_x$  为样本  $x$  距离最近的 centroid

k-mean++ 初始化 centroid:

1. 随机选择 1 个样本做 centroid
2. 剩余每一样本  $x^{(i)}$  有  $\frac{D(x^{(i)})}{\sum_{j=1}^{|S|} D(x^{(j)})}$  几率被选做新 centroid  
 $D(x^{(i)})$  为样本  $x^{(i)}$  距离最近的 centroid 的距离
3. 重复 2. 步直至得到  $k$  个 centroid

选择 cluster 数量  $k$ :

elbow approach:

实验多次, 每次选择不同  $k$  值。记录最终 loss 大小,  $k$ -loss 图像应当最初快速减小, 随后连线平缓。选择拐点处的  $k$  值作为最优超参数

cross validation:

数据分为  $n$  fold, 得到  $n$  组训练集, 验证集分配

选择不同  $k$  值, 对每一  $k$  值 在每一训练集上训练, 在验证集得到验证代价值。共得到  $n*k$  验证代价值

取  $k$  使得平均验证代价值最小

sihouette score: 所有样本的 sihouette coefficient 的均值

一样本  $x^{(i)}$  的 sihouette coefficient:  $\frac{b-a}{\max(a,b)}$

$a$  为  $x^{(i)}$  到同一 cluster 内所有样本的平均距离

$$b = \min(E_{x^{(j)} \in \text{othercluster}}(D(x^{(i)} - x^{(j)})))$$

sihouette score  $\in [-1, 1]$ , 偏向取 score 高的 cluster 数

使用 k-mean 进行数据预处理:

将数据首先进行 k-mean 分类, 将每一样本替换为 样本到最近的 centroid 距离, 传入另一模型进行学习

用于半无监督学习: 将数据进行 k-mean 分类, 从每一 cluster 选取离 centroid 最近的样本, 产生大小为  $k$  的训练集。则只需得到  $k$  个样本的标签即可进行训练

### k-mode

选择 centroid 使每一特征值分别为 对应特征中出现次数最多的特征值

### Probability Density Estimate PDE

得到样本分布的 pdf:  $\hat{p}$ , 对特征  $x$  输出可能性  $\hat{p}(x)$

non-parametric approach: 不对数据的部分做任何假设

根据整个训练集集  $S$  训练, lazy learning

kernel density estimation:

$$1. \hat{p}(x) = \frac{1}{|S|} \sum_{x^{(i)} \in S} \frac{1}{h^D} H\left(\frac{x - x^{(i)}}{h}\right)$$

$D$  为 feature 个数,  $h$  为 bandwidth,  $h^D$  即 window 体积

$H$  称 Parzen Window/kernel function

$$H(x) = \begin{cases} 1 & \forall i \in \{1, \dots, D\}, |x_i| < \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$



$$H\left(\frac{x-\hat{x}^{(i)}}{h}\right) \text{ 即判断 } x \text{ 是否在给定数据 } x^{(i)} \text{ 大小 window 内}$$

$$2.\hat{p}(x) = \frac{1}{|S|} \sum_{x^{(i)} \in S} \frac{1}{(2\pi h^2)^{\frac{D}{2}}} \exp\left(-\frac{\|x-x^{(i)}\|^2}{2h^2}\right)$$

parametric approach:

Gaussian distribution: 假设 pdf 为 Normal 分部

直接根据训练集得到最优参数, 没有迭代

1.univariate Normal distribution: 仅有一特征

$$\hat{p}(x) = N(x|\mu, \Sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

$$\mu = \frac{1}{|S|} \sum x^{(i)}$$

$$\sigma^2 = \frac{1}{|S|} \sum (x^{(i)} - \mu)^2$$

即 multivariate Normal dis 中  $D = 1$  情况

2.multivariate Normal distribution: 有多个特征

$$\hat{p}(\mathbf{x}) = N(\mathbf{x}|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^D |\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)\right)$$

$|\Sigma|$  项为  $\Sigma$  determinant

$D$  为特征数

$$\mu = \frac{1}{|S|} \sum_{x^{(i)} \in S} \mathbf{x}^{(i)}$$

$$\text{covariance matrix } \Sigma = \frac{1}{|S|} \sum_{x^{(i)} \in S} (\mathbf{x}^{(i)} - \mu)(\mathbf{x}^{(i)} - \mu)^T$$

计算 performance: neg. log-likelihood

$$\mathcal{L} = -\log(p(S|\mu, \Sigma)) = -\sum_{x^{(i)} \in S} \log(p(\mathbf{x}^{(i)}|\mu, \Sigma))$$

Theorem: 当 neg. log-likelihood 最小化,  $\mu$  和  $\sigma$  有以上公式求解

$$= \frac{N}{2} \log(2\pi) + \frac{N}{2} \log(\sigma^2) + \frac{1}{2\sigma^2} \sum_{x^{(i)} \in S} (x^{(i)} - \mu)^2$$

当  $\frac{d\mathcal{L}}{d\mu} = 0$  和  $\frac{d\mathcal{L}}{d\sigma^2} = 0$ ,  $\mu$  和  $\sigma$  有以上公式求解

**Gaussian Mixtures Model(GMM):** 一种 PDE, 假设所有子分部都为正态分部

参数:  $\theta = \{\pi_k, \mu_k, \Sigma_k | k = 1..K\}$

共  $K$  个子分部, 每一分部  $\sim N(\mu_k, \Sigma_k)$

使用多个子分部之和代表样本 pdf 分部,  $p(x) = \sum_{k=1}^K \pi_k N(x|\mu_k, \Sigma_k)$

$x$  可为向量, 则  $p(x)$  计算方法同 Multivariate Normal Distribution

此时称多元混合高斯分布

$0 \leq \pi_k \leq 1$ ,  $\sum_{k=1}^K \pi_k = 1$ , 保证产生的 pdf 积分为 1

迭代:

1. 随机初始化所有参数, 仅保证  $\sum \pi_k = 1$

2.E-Step

对每一样本  $i$ , 子分部  $k$  计算 responsibility  $r_{ik} = \frac{\pi_k N(\mathbf{x}^{(i)}|\mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j N(\mathbf{x}^{(i)}|\mu_j, \Sigma_j)}$

3.M-Step

定义  $N_k = \sum_{i=1}^{|B|} r_{ik}$  对一子分部的 responsibility 求和

更新  $\mu_k = \frac{1}{N_k} \sum_{i=1}^{|B|} r_{ik} \mathbf{x}^{(i)}$

更新 covariance matrix  $\Sigma_k = \frac{1}{N_k} \sum_{i=1}^{|B|} r_{ik} (\mathbf{x}^{(i)} - \mu_k)(\mathbf{x}^{(i)} - \mu_k)^T$

使用当前 M-Step 已更新的  $\mu$

更新  $\pi_k = \frac{N_k}{|B|}$

4. 当  $\theta$  不再大幅改变, 或当 **neg. log likelihood** 不再下降则停止, 否则回到 2.

neg. log likelihood  $\mathcal{L} = -\sum_{x^{(i)} \in S} \log(p(\mathbf{x}^{(i)}|\mu, \Sigma))$

$p$  为  $K$  个子分部加权求和值, 即一样本输出的 fit 值

调参: 选择子分部个数  $K$

$$BIC_K = \mathcal{L}(K) + \frac{P_K}{2} \log(|B|)$$

$\mathcal{L}(K)$  为使用  $K$  类别时的 neg. log likelihood

当使用特征个数  $n$  时,  $P_K = n * \frac{(n+1)n}{2} * k - 1$  为使用的参数个数

$n$  对应使用的  $\mu$  个数

$\frac{(n+1)n}{2}$  covariance 参数个数, 由于  $\Sigma$  为  $n * n$  symmetric matrix

区别 K-mean:

GMM-EM 可得到一样本  $i$  属于每一类别  $k$  的可能性, 即  $r_{ik}$

GMM-EM cluster 等高线可以为非正圆, K-mean 每一 cluster 为正圆从 centroid 向外发散

GMM-EM cluster 等高线集中程度可不同, K-mean 每一 cluster 等高线间距相同

**cluster 间分界不受等高线的弧形影响, 受交接的 2 子分部影响**

## DBSCAN

适用于一 cluster 内样本密度较高的训练集

算法:

1. 对每一样本  $x_i$  计算集合  $S_{i\epsilon}$ , 称  $\epsilon$ -neighbourhood, 包含所有距离在  $\epsilon$  内的其他样本  
 $|S_{i\epsilon}| >$  超参数  $s_{min}$  的样本称 core instance
2. 所有属于同一  $S_{i\epsilon}$  的样本判为属于同一 cluster, 当一样本  $x_i$  同时存在样本  $x_i, x_j$  的  $\epsilon$ -neighbourhood 中时, 合并  $S_{i\epsilon}, S_{j\epsilon}$ 。
3. 没有被分配进任何  $S_{i\epsilon}$  的样本判为异常值

## 9 GAN 对抗网络

基本结构:

generator  $G$ : 得到正则噪声  $\mathbf{z}$ , 生成伪数据  $\mathbf{x}'$ 。目标为使  $D(\mathbf{x}') = 1$

discriminator  $D$ , 从实际数据集  $\mathbf{x}$  或伪数据集  $\mathbf{x}'$  得到数据判断真伪。即对伪数据输出 0 真数据 1。

一次训练: (首先进行  $D$  更新, 后进行  $G$  更新)

**两部分使用不同 trainer, 每一部分计算代价值后立即更新参数**

1. discriminator 部分:

得到一批量正则噪声  $\mathbf{z}$ , 计算伪数据  $\mathbf{x}' = G(\mathbf{z})$ , 标签为  $\vec{0}$ 。

取一批量真实数据  $\mathbf{x}$ , 标签为  $\vec{1}$ 。

对 2 批量数据分别使用二元交叉熵损失函数训练。最终代价值为两部分代价值平均值。

调用  $D$  部分的 trainer

2. generator 生成数据

与  $D$  部分使用同一批量正则噪声, 使用未更新的  $G$  参数和**已更新的  $G$  参数**前向计算。输出的标签期望为  $\vec{1}$

即代价函数为  $J_G = -y \log(D(G(\mathbf{z})))$

$= -\log(D(G(\mathbf{z})))$  ( $y$  必为 1, 由于期望产生  $\mathbf{x}'$  使  $D(\mathbf{x}') = \vec{1}$ )

调用  $G$  部分的 trainer

## Deep Convolutional GAN

**VAE Divergence**

定义 divergence function  $D : \mathcal{P} \times \mathcal{P} \rightarrow \mathbb{R}$

有  $D(P, Q) \geq 0$ ,  $D(P, P) = 0$

**Jensen's inequality**

有函数  $f : \mathbb{R} \rightarrow \mathbb{R}$ , random variable  $\mathbf{x} \sim D$ 。令  $p(\mathbf{x})$  为  $D$  的 pdf

则有  $E_{p(\mathbf{x})}(f(\mathbf{x})) \geq f(E_{p(\mathbf{x})}(\mathbf{x}))$

Law of Unconscious Statisticians: 对  $Y = g(\mathbf{x})$

若  $E_{p_X(\mathbf{x})}(g(\mathbf{x})) < \infty$ , 则  $E_{p_Y(y)}(y) = E_{p_X(\mathbf{x})}(g(\mathbf{x}))$

Generalized Jensen's inequality:  $E_{p_X(\mathbf{x})}(f(g(\mathbf{x}))) = f(E_{p_X(\mathbf{x})}(g(\mathbf{x})))$

**Kullback-Leibler KL divergence**

divergence KL:  $KL(P\|Q) = \int p(\mathbf{x}) \ln\left(\frac{p(\mathbf{x})}{q(\mathbf{x})}\right) d\mathbf{x}$

令真实数据集分部  $X_{data}$ , GAN 根据参数  $\theta$  生成的数据集分部  $X_\theta$

$\theta$  代价函数  $KL(X_{data}\|X_\theta) = E_{p_{data}(\mathbf{x})}(\ln(p_{data}(\mathbf{x}))) - E_{p_{data}(\mathbf{x})}(\ln(p_\theta(\mathbf{x})))$

即最优参数  $\theta$  为最小化  $E_{p_{data}(\mathbf{x})}(\ln(p_\theta(\mathbf{x})))$

Variational Inference:

1.  $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z}) d\mathbf{z}$

$\mathbf{z}$  为 GAN 生成使用的输入 Gaussian noise, 即  $\mathbf{z} \sim \mathcal{N}(\vec{0}, I)$

$p_\theta(\mathbf{x}|\mathbf{z})$  为使用  $\theta$  生成  $\mathbf{x}$  特征的几率

由于对高维向量  $\mathbf{z}$  求积分, 难以计算

2.  $\log(p_\theta(\mathbf{x})) = \log \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z}) d\mathbf{z}$   
 $= \log \int q(\mathbf{z}) \frac{p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q(\mathbf{z})} d\mathbf{z}$   
 $\geq \int q(\mathbf{z}) \log\left(\frac{p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q(\mathbf{z})}\right) d\mathbf{z}$  (由 Jensen's inequality)  
 $= E_{q(\mathbf{z})}(\log(p_\theta(\mathbf{x}|\mathbf{z}))) - KL(q(\mathbf{z})\|p(\mathbf{z}))$

\*\* 结论:  $q(\mathbf{z}) \approx p_\theta(\mathbf{z}|\mathbf{x})$  时代价值  $E_{q(\mathbf{z})}(\log(p_\theta(\mathbf{x}|\mathbf{z})))$  较小 \*\*

**Variational auto-encoder**

根据上结论, 令  $q(\mathbf{z}) = q_\phi(\mathbf{z}|\mathbf{x})$ ,  $\phi$  为另一神经网络, 得到  $\mathbf{x}$ , 输出  $\mathbf{z}$ 。

令  $\vec{\mu}_\phi(\mathbf{x}), \log(\vec{\sigma}_\phi(\mathbf{x}))$  为网络  $NN_\phi(\mathbf{x})$  输出的均值 log std。有  $q_\phi(\mathbf{z}|\mathbf{x})$  为  $\mathcal{N}(\vec{\mu}_\phi(\mathbf{x}), \text{diag}(\vec{\sigma}_\phi^2(\mathbf{x})))$

VAE 目标即得到  $\phi, \theta$  使得  $L(\phi, \theta)$  最大

$$L(\phi, \theta) = E_{p_{data}(\mathbf{x})}(E_{q_\phi(\mathbf{z}|\mathbf{x})}(\log(p_\theta(\mathbf{x}|\mathbf{z}))) - KL(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})))$$

Theorem:  $KL(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{x})) = \frac{1}{2}(\|\vec{\mu}(\mathbf{x})\|^2 + \|\vec{\sigma}(\mathbf{x})\|^2) - 2(\log(\vec{\sigma}(\mathbf{x})) \cdot \vec{1} - d)$ ,  $d$  为  $\mathbf{z}$  向量元素数求导:

Monte Carlo Estimation:  $L(\phi, \theta)$  中  $E_{q_\phi(\mathbf{z}|\mathbf{x})}(\log(p_\theta(\mathbf{x}|\mathbf{z}))) \approx \log(p_\theta(\mathbf{x}|\mathbf{z}))$ , 其中  $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$

则对  $\theta$  求导:  $\frac{dL(x, \phi, \theta)}{d\theta} = \frac{d \log(p_\theta(\mathbf{x}|\mathbf{z}))}{d\theta}$ , 其中  $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$

对  $\phi$  求导:  $\frac{dL(x, \phi, \theta)}{d\phi} = \frac{dE_{q_\phi(\mathbf{z}|\mathbf{x})}(\log(p_\theta(\mathbf{x}|\mathbf{z})))}{d\phi} - \frac{dKL(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}{d\phi}$

Reparameterisation trick:  $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$  iff  $\mathbf{z} = \vec{\mu}_\phi + \vec{\sigma}_\phi \odot \vec{\epsilon}$ , 其中  $\vec{\epsilon} \sim \mathcal{N}(\vec{0}, I)$

则  $E_{q_\phi(\mathbf{z}|\mathbf{x})}(\log(p_\theta(\mathbf{x}|\mathbf{z}))) = E_{\vec{\epsilon} \sim \mathcal{N}(\vec{0}, I)}(\log(p_\theta(\mathbf{x}|\vec{\mu}_\phi + \vec{\sigma}_\phi \odot \vec{\epsilon})))$

则  $\frac{E_{q_\phi(\mathbf{z}|\mathbf{x})}(\log(p_\theta(\mathbf{x}|\mathbf{z})))}{d\phi} = E_{\vec{\epsilon} \sim \mathcal{N}(\vec{0}, I)}\left(\frac{\log(p_\theta(\mathbf{x}|\vec{\mu}_\phi + \vec{\sigma}_\phi \odot \vec{\epsilon}))}{d\phi}\right)$

$= E_{\vec{\epsilon} \sim \mathcal{N}(\vec{0}, I)}\left(\frac{d\mathbf{z}}{d\phi} \frac{\log(p_\theta(\mathbf{x}|\mathbf{z}))}{d\mathbf{z}} \Big|_{\mathbf{z}=\vec{\mu}_\phi + \vec{\sigma}_\phi \odot \vec{\epsilon}}\right)$

$= \frac{d\mathbf{z}}{d\phi} \frac{\log(p_\theta(\mathbf{x}|\mathbf{z}))}{d\mathbf{z}} \Big|_{\mathbf{z}=\vec{\mu}_\phi + \vec{\sigma}_\phi \odot \vec{\epsilon}}$ , 其中  $\vec{\epsilon} \sim \mathcal{N}(\vec{0}, I)$

应用:

$p_\theta(\mathbf{x}|\mathbf{y})$  中  $\mathbf{y}$  代表期望产生的数据类别, 生成网络输入为  $y, z$

## 10 RL 强化学习

### 基本定义

程序在 environment 环境中根据观测得到的 state 状态，选择 action 行为，得到 reward 反馈  
模型整体符号定义  $\langle A, S, R, P \rangle$

Action space  $A$

State space  $S$

Reward  $R: \sum \times A \times S \rightarrow R$

Transition  $P: \sum \times A \rightarrow S$

第  $i$  决策的符号定义：

$a_i \in A$  采取的行为

$r_i \in R$  得到的 reward，每一行为可以立即得到反馈值

exploring 探索：模型尝试新行为

exploiting 利用：模型使用已知高反馈行为

### policy

根据观测选择  $a_i$  的算法

stochastic policy：policy 中有随机性

随机性提高模型 explore 新行为

genetic algorithm：遗传算法

policy gradients：对参数求导，更新参数

### credit assignment：

对每一决策分配 discounted reward，代表此决策对随后几次决策的反馈值影响  
定义：

$l$  此次试验一共包含的决策数， $\gamma$  为 discount factor

计算：

第  $l-1$  决策有决策有 discounted reward： $d_i = r_i$

第  $i$  决策有 discounted reward： $d_i = r_i + \gamma * d_{i+1}$

正则化：对所有实验中每一次决策  $r_i$  取整体平均值，方差，求标准化

### neural network policy

前向传播：使用神经网络得到行为可能性，根据可能性选择行为。属于 generic gradient policy

单次迭代：

定义：一次决策

1. 从 policy 得到行为可能性

2. 用交叉熵代价函数求代价值， $\mathbf{y\_hat}$  为 1. 中可能性， $\mathbf{y}$  为实际采取的行为

3. 根据代价函数求斜率，斜率使神经网络输出可能性更偏向采取的行为。但不立即使用斜率

1. 随机初始化 1 次模型，对  $n$  个随机初始化环境进行试验，每一试验中包含多个决策

每一环境得到决策数不一定相同，取决于试验中进行的决策次数

2. 对每一决策求 discounted reward，结果包含  $n$  组数组，第  $l_i$  组数组对应第  $i$  次实验的 discount reward 数组

3. 对所有 discounted reward 标准化，平均值 方差为所有 dis reward 值

4. 对每一参数每一决策的斜率乘对应决策的 dis reward。对结果中所有属于同一参数的乘积取平均值，为此参数的斜率
5. 使用斜率更新参数值

### Markov Decison Process MPD

每一状态  $s_i$  有可执行的行为集合  $A_{s_i} \subseteq A$ 。不同状态行为集合可有交集  
 行为  $a_{ij} \in A_{s_i}$  代表状态  $s_i$  执行行为  $a_{ij}$ 。执行  $a_{ij}$  后有  $p_{a_{ij}, s_k}$  几率到达状态  $s_k$   
 optimal state value  $V^*(s_i)$ :

模型到达状态  $s_i$  后 选择最理想的  $a_{ij}$  能得到的 discounted reward 总和

$$V^*(s_i) = \max_j \sum_s [p_{a_{ij}, s_k} (R(s_i, a_{ij}, s_j) + \gamma \cdot V^*(s_j))]$$

Q-value iteration algorithm:

得到 从  $s_i$  选择  $a_{ij}$  后期望的 discounted reward 值

$$\text{迭代: } Q_{n+1}(s_i, a_{ij}) = \sum_j [p(a_{ij}, s_j) (R(s_i, a_{ij}, s_j) + \gamma \cdot \max_{a_{jk}} (Q_n(s_k, a_{jk})))]$$

policy: 状态为  $s_i$  时选取  $a_{ij} = \max_{a_{ij}} Q^*(s_i, a_{ij})$

### Temporal Difference Learning TD learing

在  $p_{a_{ij}, s_j}$ ,  $R$  未知的情况下迭代得到  $V(s_i)$

更新函数:  $V(s) = (1 - \alpha)V(s) + \alpha \cdot (r + \gamma \cdot V(s'))$

$$= \leftarrow_{\alpha} r + \gamma \cdot V(s') \quad \text{简介写法}$$

$s'$  为状态  $s$  能达到的下一状态

$\alpha$  为学习率,  $\gamma$  为 discount factor

### Q-Learning

方法 1: 在  $p_{a_{ij}, s_j}$ ,  $R$  未知的情况下每次决策更新一个  $Q(s_i, a_{ij})$  值。每次决策选择  $Q_{s_i, a_{ij}}$  最大的行为

更新函数:  $Q(s_i, a_{ij}) \leftarrow_{\alpha} r + \gamma \cdot \max_{a_{jk}} (Q(s_j, a_{jk}))$

$\alpha$  为学习率,  $\gamma$  为 discount factor

所有  $Q(s_i, a_{ij})$  初始化为 0

$r$  为实际得到的 reward

$\max_{a_{jk}} (Q_n(s_j, a_{jk}))$  为估计的 discounted reward 总和, 取值为: 决策后实际到达的下一状态  $s_j$  的所有  $Q(s_j, a_{jk})$  中最大值

方法 2: 为避免需要过多实验才能得到准确的  $Q^*(s_i, a_{ij})$ , 使用  $\epsilon - greedy policy$ 。

决策时每一步有  $\epsilon$  几率随机选择下一行为,  $1 - \epsilon$  几率选择  $Q_{s_i, a_{ij}}$  最大的行为

$Q$  更新函数同方法 1

方法 3: 另一随机 explore 算法, 使用 exploration function 根据行为被选择的次数判断行为被 explore 的程度

选择行为时仍选取  $\max_{a_{ij}} Q(s_i, a_{ij})$

更新函数:  $Q(s_i, a_{ij}) \leftarrow_{\alpha} r + \gamma \cdot \max_{a_{jk}} f(Q(s_j, a_{jk}), N(s_j, a_{jk}))$

$N(s_j, a_{jk})$  为行为  $a_{jk}$  在状态  $s_j$  下被选择的次数。

$f(Q, N)$  根据选择次数和  $Q$  值决定行为的优先级。例:  $f(Q, N) = Q + \frac{\kappa}{1+N}$

### Approximate Q-Learning

解决当模型的状态数 行为数过大时训练过慢的问题。通过找到方程  $Q_{\theta}(s_i, a_{ij})$  估计实际  $Q^*(s_i, a_{ij})$ , 根据给定参数向量  $\vec{\theta}$

### deep Q-networks DQNs

使用神经网络估计  $Q_{\theta}(s_i, a_{ij})$  值, 使用的神经网络称 DQN

DQN 得到  $s_i$ , 返回  $Q_{\theta}(s_i, a_{ij})$ 。最终选取行为时根据  $Q_{target}(s_i, a_{ij}) = r + \gamma * \max_{a_{jk}} Q_{\theta}(s_j, a_{jk})$  得到训练集:

一个样本包含  $s_i, a_{ij}$ , 得到的 reward  $r$ , 进入的下一  $s_j$ , bool 值  $done$  代表下一状态为终止状态  
将所有样本加入集合 replay buffer, 取样本时随机选取, 避免相邻样本的 correlation 影响训练  
 $s_i$  可为图片, 无需人工得到参数

迭代:

1. 进行多组试验, 每组试验的每一决策加入 replay buffer。前几次迭代不进行训练, 由于 replay buffer 中样本多样性较低

2. 训练从 replay buffer 取一批量样本, 对于每一样本  $(s_i, a_{ij}, r, s_j, done)$

得到  $Q_{target}(s_i, a_{ij})$

将下一状态  $s_j$  通过 DQN 得到向量  $Q_{\theta}(s_j, \vec{a}_{jk})$ , 元素数为行为数

对  $Q_{\theta}(s_j, \vec{a}_{jk})$  取最大值, 即得到  $\max_{a_{jk}} Q_{\theta}(s_j, a_{jk})$  值

根据  $\max_{a_{jk}} Q_{\theta}(s_j, a_{jk})$  计算  $s_i$  的新 target 值  $Q_{target}(s_i, a_{ij})$

得到  $Q_{\theta}(s_i, a_{ij})$

将  $s_i$  传入 DQN, 得到  $Q_{\theta}(s_i, \vec{a}_{ij})$ 。并和 one hot 向量点乘 得到决策实际选取的行为  $a_{ij}$  的  $Q(s_i, a_{ij})$

3. 代价值为一批量的  $Q_{\theta}(s_i, a_{ij})$  和  $Q_{target}(s_i, a_{ij})$  的平均平方代价值, 根据代价值梯度下降训练 DQN

### Fixed Q-Value Target

使用 2 神经网络, 分别负责计算  $Q_{target}$ ,  $Q_{\theta}$

两网络初始参数一致, 每 n 次循环后将计算  $Q_{\theta}$  的网络参数抄入  $Q_{target}$  网络

计算 target 时使用  $s_i$  传入  $Q_{target}$  网络的结果。即 使得 target 计算不再每一循环一更新而是每 n 循环更新。训练更稳定

### Double DQN

类似 Fixed Q-Value layer, 每次循环更新  $Q_{target}$  网络, 每 n 次循环后将  $Q_{target}$  抄入  $Q_{\theta}$

### Prioritized Experience Relay

### Evolutionary algorithm

通过交换参数的一部分得到更优的参数组合

模型:

对参数向量  $\vec{x} = [x_1, x_2, \dots]$  评估方程  $f(\vec{x}) \in \mathbb{R}$

目标: 得到  $\vec{x}$  使  $f(\vec{x})$  值最大化

算法:

1. 随机初始化 n 参数向量

2. 每一迭代中取 m 个参数向量进入集合 S

取法 1: 每一参数向量  $\vec{x}_i$  有  $\frac{f(\vec{x}_i)}{\sum_i f(\vec{x}_i)}$  几率被选中进入 S

取法 2: 将每一参数向量  $\vec{x}_i$  根据  $f(\vec{x}_i)$  排序, 排名第 j 向量有  $P(\vec{x}_i) = p * (1 - p)^{(j-1)}$  几率被选中进入 S。第 m 参数向量有可能性  $1 - (1 \text{ 到 } m-1 \text{ 参数可能性之和})$

取法 3: 定义参数向量间间距  $D(\vec{x}_i, \vec{x}_j)$ , 用于保证选取的参数分部范围广泛

取  $f(\vec{x}_i)$  最大的  $\vec{x}_i$  放入 m 集合

随后选取 m-1 个向量, 选取  $\vec{x}_j$  使得  $f(\vec{x}_j) * E_{v \in S}(D(\vec{x}_j, v))$  最大的

即选取参数向量使得 其与已经选择的向量平均距离 \* 自身  $f(\vec{x}_j)$  值 最大化  
 tournament 取法：每次随机选择  $n$  个参数向量，选择其中 fit 值最高者加入  $S$   
 支持 concurrent 创建  $S$   
 无需准确得到 fit 值，只需能够比较两参数向量优劣即可  
 Elitism 取法：选择 fit 最优的部分参数向量直接加入  $S$ ，比例多取 10%

- 根据最优  $m$  个参数向量，交换参数得到  $m$  个下一代参数向量，变异出剩余  $n-m$  个参数向量  
 交换参数：对  $\vec{x} = [x_1, x_2, \dots]$ ,  $\vec{y} = [y_1, y_2, \dots]$  交换结果为  $\vec{x}' = [x_1, y_2, \dots]$ ,  $\vec{y}' = [y_1, x_2, \dots]$   
 交换元素的位置根据算法可变  
 变异向量由已有的  $m$  个后代参数向量变异得到  
 对  $\vec{x} = [x_1, x_2, \dots]$ ，变异参数向量每一元素  $x'_i$  有  $x'_i U(-step\_size + x_i, step\_size + x_i)$   
 使用的分部不一定为 Uniform 分部
- 当迭代次数达到限制，当一参数向量 fit 值达到限度，当 fit 值不再大幅改变，停止

### Genetic algorithm

定义：

Gene：单一可选的参数值

Genotype：一段二进制值，对应一参数向量

当参数使用编码导致可选值大于限定值域，将多余编码的评估值设为 0，使其不被选入下一

迭代

Phenotype：将 Genotype 分离成单个参数对应参数向量中每一元素

crossover 交换数据时交换 Genotype 一段 bit 值

mutation 变异：每一 bit 有  $m$  几率取相反值， $m$  常取  $\frac{1}{genotype\_length}$

### Evolutionary strategy ( $\mu + \lambda$ ) - ES

定义：

Genotype：一数组实数

$\mu, \lambda$  为给定超参数。常取  $\frac{\lambda}{\mu} = 5$

初始化：随机创建  $\mu + \lambda$  个参数向量

选择  $S$ ：选择 fit 值最高的  $\mu$  个参数向量

生成下一代参数：

- 使用超参数  $\sigma$ ：随机选择  $\lambda$  个向量  $\vec{x}_i$ ，分别生成  $\vec{x}_j = \vec{x}_i + N(0, \sigma)$   
 + 为对每一  $\vec{x}_i$  中参数值加 normal 变量  
 $\sigma$  较大则数据分散。较小则学习率低，受局部最优影响。
- 在迭代过程中改变  $\sigma$ ，有超参数  $\tau_0 \propto \frac{1}{\sqrt{n}}$ ， $n$  为参数向量元素数  
 前一迭代有  $\sigma_i$ ，选择  $\lambda$  个参数向量  $\vec{x}_i$   
 下一迭代有  $\sigma_j = \sigma_i \exp(\tau_0 N(0, 1))$   
 生成新参数向量  $\vec{x}_j = \vec{x}_i + \sigma_j N(0, 1)$

### Novelty Search

定义：

$Novelty(x, A) = \frac{1}{N} \sum_i^N d(x, x_i)$ ，即参数向量到  $A$  中  $N$  个邻近参数向量的平均距离。 $A$  为 Novelty Archive，为一参数向量集合

behavioural descriptor: 定义参数向量对应的策略类型, Novelty Search 目标为找到合适的 behaviour descriptor

迭代:

fit 函数即 Novelty(x), 一次迭代从参数向量中选择 Novelty 最高向量加入 Archive, 由 archive 生成下一代参数向量

### Novelty Search with Local Competition

为一 Multi-objective EA: 同时最大化 Novelty 和 Local Competition

Local Competition LC(x): 对 Archive 中邻近的 N 个参数向量, 其中 fit 值  $< f(x)$  的向量个数

迭代时将  $f(x_i) < f(x)$  的  $x_i$  替换为 x

### MAP-Elites

将参数向量对应到 2d 网格, 每一格仅存在 0-1 个向量。

向量在网格的分部不一定为 uniform

迭代时随机选择一个参数, 变异。得到新的参数向量对应网格中一格, 若已有一参数向量对应此格, 则选择两参数向量中 fit 值较大的加入 archive, 较小参数被移除

performance 计算:

diversity: archive size, 即网格中有参数对应的格数

fit 值: archive 中参数的最大或平均 fit 值

converge 速度

QD-Score: archive 中参数的 fit 值总和

假设 fit 值全部为正

同时考虑 diversity 和 fit 值的表现

## 11 机器翻译

分析翻译结果

Bilingual Evaluation understudy BLEU-n:

对预测翻译  $\hat{s} = [\hat{s}_1, \hat{s}_2, \dots]$ , 计算和给定目标翻译  $s$  的相似度

$$\text{BLEU-n} = BP * (\prod_i^n (i \cdot \text{precision}(i)))^{\frac{1}{n}}$$

$\text{precision}(i)$ :

使用  $i$  词长度的窗口, 选取  $\hat{s}$  的子字符串  $\{[\hat{s}_1, \dots, \hat{s}_i], [\hat{s}_2, \hat{s}_{i+1}], \dots\}$

$\text{precision}(i) = \text{子字符串中同时为目标翻译 substring 的个数} / \text{子字符串个数}$

BrevityPenaltyBP: 当输出远小于目标长度, 惩罚更多

$$BP = \min(1, \frac{|\hat{s}|}{|s|})$$

BERTScore:

对预测和目标翻译每一词转为词向量

对每一目标词向量, 和每一预测词向量点乘, 取最高值作为权重

求目标词权重平均值, 当赋目标词权重时可计算加权平均值

### Recurrent Network Machine Translate RNMT

使用两 RNN, 分别为 encoder, decoder

encoder RNN 得到输入句, 产生隐藏状态作为 decoder 的隐藏状态。

decoder 初始输入为  $< s >$  代表句子开始产生词序, 直至输出为  $< /s >$  作为句末结束



## 12 分析结果

### 训练过程

分离训练, 验证, 测试数据集:

- 将数据集按 (0.6, 0.2, 0.2) - (0.8, 0.1, 0.1) 分为训练 验证 测试集
- cross validation

样本分为  $N$  个 fold  $F = \{f_1, \dots, f_N\}$

#### 1.k-fold cross validation

当无需对超参数调参时使用, 不能分离验证集的原因为: 验证集过小, 将没有代表性

算法:

$f_1, \dots, f_N$  分别作为测试集  $f_i$ , 剩余  $N-1$  fold  $\{F \setminus f_i\}$  作为训练集。进行  $N$  次训练, 得到  $N$  个同一超参数产生的训练结果

$N$  个训练后模型在对应测试集准确率为  $x_1, \dots, x_N$ 。模型的准确率为  $\frac{1}{N} \sum_i x_i$

#### 2.nested k-fold cross validation

比较  $M$  组不同超参数  $p_1, \dots, p_M$  时使用

算法:

$f_1, \dots, f_N$  分别作为测试集  $f_i$ , 在剩余  $N-1$  fold  $\{F \setminus f_i\}$  中每一 fold 分别作为验证集  $f_j$ 。

- 每次 tuning 使用  $\{F \setminus \{f_i, f_j\}\}$  做训练集,  $f_j$  做验证集。
  - 每一  $p_m$  在  $f_j \in \{F \setminus f_i\}$  上测试有平均验证代价值  $c_{(p_m, f_j)}$
  - 对每一  $p_m$ , 得到  $p_m$  的平均验证代价值。取代价值最小的模型  $p_i^*$  在测试集  $f_i$  上测试最优模型在测试集上得到测试集代价值  $c_{(p_i^*, f_i)}$ 。最终总代价值为  $\sum_i \frac{c_{(p_i^*, f_i)}}{N}$
- 最终测试集代价平均值为选取模型的算法的代价值

hyperparameter tuning 得到最优模型设计:

使用不同超参数在训练集上训练, 得到多个训练结束后的模型。

在验证集上测试, 选取准确度最高的 hyperparameter。选取 hyperparameter 后可将测试集和验证集合并重新训练, 使模型使用的训练数据集更大。

最终在测试集上运行, 得到模型准确度。

**confusion matrix 困惑矩阵:** 分析二元/多元分类

$$\begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}$$

一行对应同一期望输出, 一列对应同一计算输出

$T/F$ : 此位置的计算输出是否和期望输出一致

$P/N$ : 此位置的计算输出是否为真

对一个类别/一元:

$$\text{precision} = \frac{TP}{TP+FP}$$

即  $P$ ( 计算结果匹配 | 计算结果为正 ): 所有计算为真的样本中预测正确的概率

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

即  $P$ ( 计算结果匹配 | 期望结果为正 ): 所有期望为真的样本被正确预测的概率

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

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}$$

precision 和 recall 的调和平均值

$$F_{\beta} = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

$\beta$ : 当 precision 为 recall $\beta$  倍重要

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

**classification error** = 1 - accuracy

**macro-average recall**: 所有 recall 的平均值, 此处 recall 为每行正确预测的概率

**macro-average precision**: 所有 precision 平均值, 即每列求 precision 取均值

多类标签 **confusion matrix**

$$\begin{bmatrix} TP & FN & \dots \\ FP & TN & \text{undefine} \\ \dots & \text{undefine} & TN \end{bmatrix}$$

对第一类标签的 confusion matrix

**micro-avaeraging**: 将所有类别的 TP 之和 / 所有类别的 TP + FN 之和

当所有样本只能符合一个类别时 = accuracy

**imbalanced dataset**

1 类标签样本数远多于另一样本

1. 将 confusion matrix 所有值换为关于横行的百分比, 假设所有期望类别下的样本数相同

2. down/up sample: 舍弃/复制部分样本, 使每一期望类别下样本数相同

无法反应整个模型 generalise 性, 由于实际使用时数据为 imbalance 的

**confidence interval**

true error  $error_D(h)$ : 模型  $h$  实际的代价值

sample error  $error_S(h)$ : 模型  $h$  在测试集上运行的平均代价值 或预测错误类别的样本比例

error 的 confidence interval =  $error_S(h) \pm Z_{\alpha/2} \sqrt{\frac{error_S(h)(1-error_S(h))}{n}}$

**比较两个模型**

randomisation test: 随机交换 2 模型多个测试结果, 比较交换前后模型 performance

two-sample T-test: 两模型在不同测试集上测试, 得到 performance 个数不同。使用 common variance 求 p 值

paired T-test: 两模型在相同测试集上测试, performance 个数相同。取 performance 差求 T-test p 值

**p-hacking**

定义: 模型依赖于无关的参数得到 p 值 < sig. level

例: 对 M pair 特征, 验证是否两个参数存在相关性。当增加实验的特征对时, 仅有小部分相关性真实存在

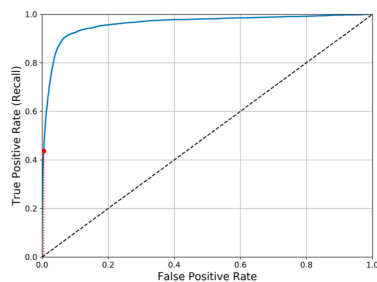
解决:

对 M 组特征的 p 值排序, 得到  $p_i < \dots < p_M$

第  $i$  位置的特征使用 sig. level  $z_i = \text{sig.level} * \frac{i}{M}$

实际存在的特征对  $i$  为  $p_i < z_i$

**ROC curve**: 分析二元/多元分类

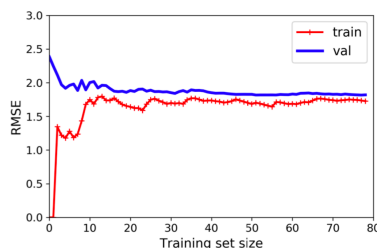


y 轴 recall 值, x 轴 false positive rate  $FPR = \frac{FN}{FN+TN} = \frac{FN}{1-specificity}$

期望的 ROC curve 为 recall 从 0 快速增长到 1。并保持直到  $FPR$  为 1。

即期望曲线下方面积接近 1

**learning curves:** 观察模型是否有 over underfit



x 轴为一整次训练 (包含多次 epoch) 使用的训练集大小, y 轴为 root MSE。

画出训练集 测试集在使用不同训练集大小后的 root MSE。

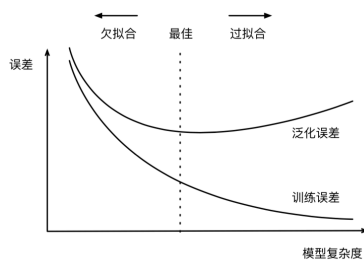
分析:

期望 2 曲线平缓值低且相近,

当 2 曲线平缓值差值较大, 测试集平缓值较低, 则过拟合

当 2 曲线平缓值较高, 则欠拟合

**模型复杂度-error epoch-error:**



2 种图, 形状类似, x 轴内容不同