

机器学习知识点

机器学习知识点

1. 机器学习相关基础概念

1.1 Variance 方差 与 Bias 偏差

1.2 Overfitting 过拟合

1.3 常用性能指标

1.4 L1 与L2 正则化

1.5 梯度下降法与牛顿法

1.6 Cross-validation

1.7 Resampling 重采样

1.8 机器学习分类

1.9 End-to-End ML Workflow

Clarify the problem and constraints

Establish metrics

Understand data sources

Explore your data

Clean your data

Feature engineering

Model selection

Model training & evaluation

1.10 生成模型与判别模型

1.11 Linear regression 线性回归

1.12 Logistic regression 逻辑回归

1.12 Logistic regression 逻辑回归

Multiclass Logistic regression

Cross Entropy

面试问题

1.13 SVM 支持向量机

1.14 Naïve Bayes 朴素贝叶斯

面试问题

1.15 TREE-BASED 树相关

Classification tree

Decision tree

Random Forest

面试问题

1.16 常见聚类方法

面试问题

1.17 集成学习

Boosting、Bagging、Stacking

Gradient Boosting

GBDT 梯度提升树

XGBoost

比较

面试问题

1.18 Dimensionality Reduction 降维

PCA

LDA

面试问题

经验风险最小化 (ERM) 与结构风险最小化 (SRM)

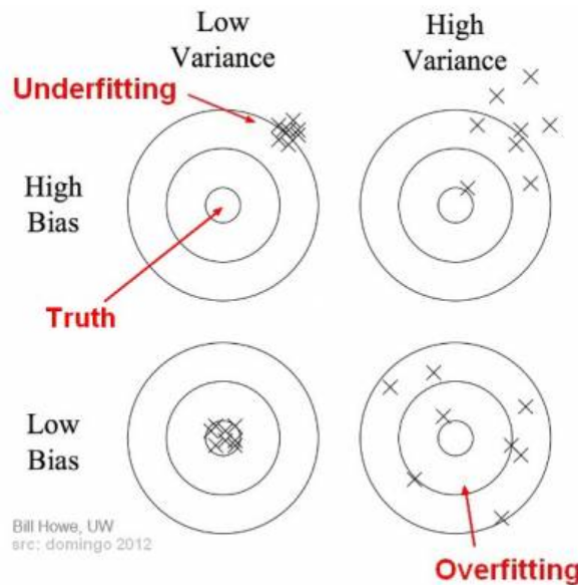
极大似然估计 (MLE) 与最大后验概率估计 (MAP)

- 2. 线性神经网络
- 3. 卷积神经网络
 - 2.1 卷积相关定义
 - 2.2 LeNet卷积神经网络
 - 2.3 AlexNet 深度卷积神经网络
 - 2.4 VGG 使用块的网络
 - 2.5 Batch Normalization 批量规范化
 - 2.6 ResNet 残差网络
 - 2.7 GoogLeNet
- 3. 优化算法
- 4. 计算机视觉
 - 4.1 图像增广
 - 4.2 Fine-tuning 微调
 - 4.3 语义分割
 - 4.4 HRNet 高分辨网络
- 5. Transformer 模型
 - 5.1 Transformer
 - 5.2 Vision Transformer
- 6. Variational autoencoder(VAE)

1. 机器学习相关基础概念

1.1 Variance 方差 与 Bias 偏差

- Loss function (误差函数) 通过将Loss (或者叫error) 最小化的过程来提高模型的性能 (performance)。然而我们学习一个模型的目的是为了解决实际的问题 (或者说是训练数据集这个领域 (field) 中的一般化问题)，单纯地将训练数据集的loss最小化，并不能保证在解决更一般的问题时模型仍然是最优，甚至不能保证模型是可用的。这个训练数据集的loss与一般化的数据集的loss之间的差异就叫做generalization error
 - Generalization error又可以细分为Bias和Variance两个部分， **Error (误差) = Bias + Variance**
- **Variance:** the extent to which model prediction error changes based on training inputs
 - Variance反映的是模型每一次输出结果与模型输出期望之间的误差，即模型的稳定性
 - High variance: pay lots of attention to training data and do not generalize on the data it hasn't seen
 - represents a model's sensitivity to small fluctuations in the training data
- **Bias:** difference btw average prediction of our model and the correct value which we are trying to predict (预测和真实的差距)
 - Bias反映的是模型在样本上的输出与真实值之间的误差，即模型本身的精准度
 - High bias: pay little attention to training data and oversimplifies the model



- **Error** 反映的是整个模型的准确度
- Irreducible error: variation due to inherently noisy observation processes
 - Balance between bias and variance → to minimize the total error = bias² + variance + irreducible
$$Err(x) = (E[\hat{f}(x)] - f(x))^2 + E[(\hat{f}(x) - E[\hat{f}(x)])^2] + \sigma_e^2$$
- Bias Variance Tradeoff
 - If our model is too simple and has very few parameters then it may have high bias and low variance. On the other hand if our model has a large number of parameters then it's going to have high variance and low bias. So we need to find the right/good balance without overfitting and underfitting the data
 - 如果要降低模型的Bias，就一定程度上会提高模型的Variance，反之亦然。造成这种现象的根本原因是试图用有限训练样本去估计无限的真实数据。想要尽量保证模型在训练样本上的准确度，但会减少模型的Bias。模型会失去一定的泛化能力，从而造成过拟合，降低模型在真实数据上的表现，增加模型的不确定性
 - 相反，如果更加相信我们对于模型的先验知识，在学习模型的过程中对模型增加更多的限制，就可以降低模型的variance，提高模型的稳定性，但也会使模型的Bias增大
- <https://www.zhihu.com/question/27068705>
- 面试: given a specific situation
 - 例如model有high variance
 - source additional data to fix the issue, reduce feature /complexity, add regularization term
 - 如果model有high bias
 - 应该 increase the complexity of model

1.2 Overfitting 过拟合

- The ultimate goal: a model that can generalize to learn some relationships within datasets
- Overfit: when a model fits exactly against its training data — train error小, test error大
 - Overfit: when a model fits exactly against its training data — train error小, test error大
- Overfit: when a model fits exactly against its training data — train error小, test error大
 - Unable to capture the relationship between input and output, generating a high error on both training and unseen data
- 面试: 经常问how to detect it, how to avoid it
- Detect Overfit
 - Perform well on train, bad on test
 - (maybe cross-validation, 如果不同subset, error相差比较大 → overfit)
 - learning curve
 - By analyzing the learning curve. If the training error is improving, but the validation error is not, it is overfitting and the training can be stopped
 - If there's a large gap between training curve and validation curve & 不变小as training time increases data wasn't representative
 - Learning Curves: plots of model learning performance over time, X = experience(time, training set size), y can be a metric of learning
- Avoid Overfit
 - more data & reduce complexity
 - Cross-validation (k-fold)
 - split into k groups → 1 for test and the others for train → repeat until each individual group has been used for testing
 - Data augmentation
 - Artificially increase the size of data; Ex. image → flipping, rotating, rescaling
 - Feature selection
 - Select most important features for training
 - PCA
 - L1/L2 Regularization
 - Regularization is a technique to constrain our network from learning a model that is too complex → aims to reduce the complexity of models
 - Add a penalty term on the cost function to push the estimated coefficients towards zero
 - Early Stopping
 - First, train the model for an arbitrarily large number of epochs and plot the validation loss graph (e.g., using hold-out). Once the validation loss begins to degrade (e.g., stops decreasing but rather begins increasing), we stop the training and save the current model

- [8 Simple Techniques to Prevent Overfitting](#)

1.3 常用性能指标

- 回归任务性能指标

均方误差 (mean squared error) $MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$

- 分类任务性能指标

- **错误率**是分类错误的样本数占样本总数的比例, **精度**则是分类正确的样本数占样本总数的比例
- 根据样本预测样例与真实类别的组合可划分为四类: TP (真正例), FP (假正例), TN (真反例), FN (假反例)
- **准确率** $Precision = \frac{TP}{TP+FP}$
- **召回率** $Recall = \frac{TP}{TP+FN}$
- **F1-score**是一个综合考虑查准率与查全率的度量 $F1 = \frac{2PR}{P+R}$
- ROC (receiver operating characteristic) 全称是“受试者工作特性”曲线。其综合考虑了概率预测排序的质量, 体现了学习器在不同任务下的“期望泛化性能”的好坏
 - ROC曲线的纵轴是“真正例率” (True Positive Rate, TPR), 横轴是“假正例率” (False Positive Rate, FPR)

$$TPR = \frac{TP}{TP+FP}$$

$$FPR = \frac{FP}{TN+FP}$$

- <https://blog.csdn.net/lrs1353281004/article/details/79411552>

1.4 L1 与L2 正则化

- Regularization is a technique to constrain our network from learning a model that is too complex
 - **Aims to reduce the complexity of models, prevent overfitting**
 - Significantly reduce variance while only slightly increase bias
- 方法: By introducing a regularization term to a general loss function: adding a term to the minimization problem
- L1 & L2
 - Add a penalty term on the cost function to push the estimated coefficients towards zero
 - λ : tuning parameter that decides how much we want to penalize the flexibility of our model
- L1 (Lasso): sum of absolute value of coefficients as a penalty 绝对值
 - coefficients can be adjusted based on cost function
 - Can be feature selection, since coefficients can be shrink to 0
 - L1 可以shrink to 0, so it leads to sparser models
 - Why: <https://medium.com/analytics-vidhya/effects-of-l1-and-l2-regularization-explained-5a916ecf4f06>
- L2 (Ridge): sum of squared values of coefficients as a penalty 平方

$$\sum_{i=1}^n (Y_i - \sum_{j=1}^p X_{ij}\beta_j)^2 + \lambda \sum_{j=1}^p |\beta_j|$$

- Only toward 0

$$\sum_{i=1}^n (Y_i - \sum_{j=1}^p X_{ij}\beta_j)^2 + \lambda \sum_{j=1}^p \beta_j^2$$

- Elastic Net: linear combination of L1 and L2
- Difference: <https://medium.com/analytics-vidhya/l1-vs-l2-regularization-which-is-better-d01068e6658c>
 - The main intuitive difference is that L1 tries to estimate the median of the data while the L2 tries to estimate the mean to avoid overfitting
 - L1 helps in feature selection by eliminating the features that are not important → helpful when #features are large in number

L1 Regularization	L2 Regularization
1. L1 penalizes sum of absolute values of weights.	1. L2 penalizes sum of square values of weights.
2. L1 generates model that is simple and interpretable.	2. L2 regularization is able to learn complex data patterns.
3. L1 is robust to outliers.	3. L2 is not robust to outliers.

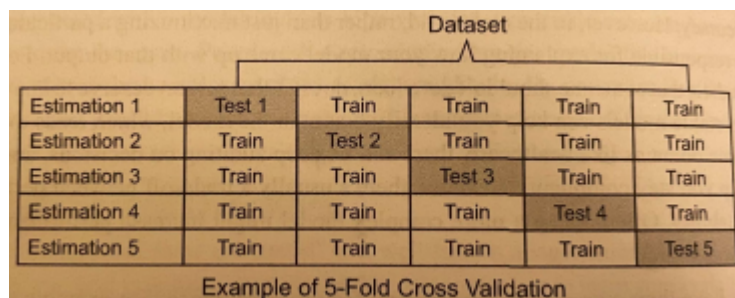
- 面试：正则化有什么用，为什么有用，L1正则为什么能使参数稀疏，为什么能防止过拟合
 - L1正则化和L2正则化降低模型的复杂度
 - 稀疏矩阵指的是很多元素为0，只有少数元素是非零值的矩阵，即得到的线性回归模型的大部分系数都是0
 - 正则化倾向于让权值尽可能小，构造一个所有参数都比较小的模型。一般认为参数值小的模型比较简单，能适应不同的数据集，也在一定程度上避免了过拟合现象
- L1 和 L2如何选择
 - L1: sum of absolute value of coefficients 当feature不多并且correlation高的时候用
 - L2: sum of squared value of coefficients 当feature很多但correlation不高的时候用
- 为啥L1能shrink to 0
 - there are "corners" in the constraint, which in two dimensions corresponds to a diamond. If the sum of squares "hits" one of these corners, then the coefficient corresponding to the axis is shrunk to zero
 - As p increases, the multidimensional diamond has an increasing number of corners, and so it is highly likely that some coefficients will be set equal to zero
- L1 more robust (and more robust to outliers)
 - robust: its forecasts are consistently accurate, even if one or more of the input variables or assumptions are drastically changed
 - 原因： L2 regularization takes the square of the weights, so the cost of outliers present in the data increases exponentially. L1 regularization takes the absolute values (cost increases linearly)

1.5 梯度下降法与牛顿法

- Gradient Descent is an iterative optimization algorithm used to find a local minimum/maximum of a given function, which is commonly used to minimize a loss function.
- Gradient: a slope of a curve at a given point in a specified direction
 - iteratively calculates the next point by using a gradient at the current position → scales it (by a learning rate) → subtracts the obtained value from the current position (makes a step)
 - subtracts: because we want to minimize (to maximize it would be adding)
- 梯度下降是一种无约束优化方法，但其成功取决于函数的各种条件，例如函数的可微性等
 - 如果函数是 non-convex，则无法保证全局最优
 - Convex (line segment connecting two function's points lays on or above its curve (it does not cross it))
- Always converge to similar points
 - No. Because in some cases (non-convex), they reach a local optima point instead of global optima, and this is governed by the data and starting conditions
- <https://www.cnblogs.com/pinard/p/5970503.html>
- <http://blog.csdn.net/lipengcn/article/details/52698895>
- [https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21#:~:text=Gradient%20descent%20\(GD\)%20is%20an,e.g.%20in%20a%20linear%20regression](https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21#:~:text=Gradient%20descent%20(GD)%20is%20an,e.g.%20in%20a%20linear%20regression)

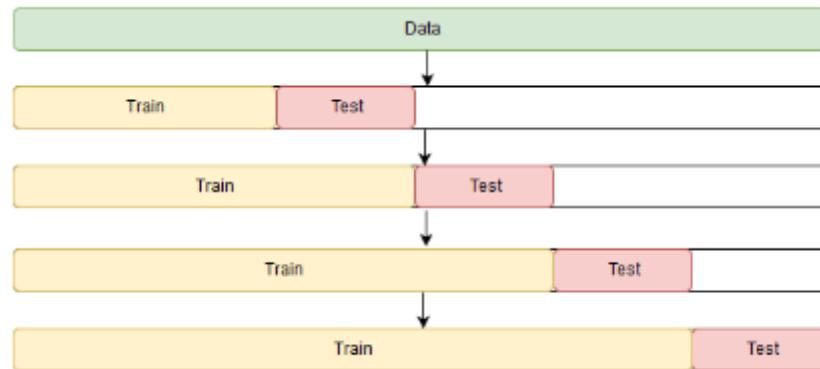
1.6 Cross-validation

- It's a technique used for model selection, by assessing the performance in several sub-samples of training data and how it generalizes on unseen
- Method for estimating the performance on unseen data by generating many non-overlapping train/test splits into training data and reporting the avg test set performance across all data splits
 - Used for model selection
 - Used when data不够 or getting more data is costly
- K-fold
 - Randomly split data into equally-sized folds
 - For each fold, train on all other folds and validate on this fold
 - Average k validation errors to get estimate of true error



- Leave-one-out LOOCV

- Testing on every single data point
- 面试: how to apply cv in time series data
 - 不能直接k-fold, 因为不能用未来data预测过去
 - Use historical data up to a given point, and vary that point in time from beginning till the end



1.7 Resampling 重采样

- repeatedly drawing samples from a training set and refitting a model on each sample in order to obtain additional information about the fitted model
 - 例子: estimate model variability, fit model 在不同sample
- **Cross-validation**
 - Resampling procedure that used to estimate test prediction error of a ml model to evaluate its performance, or to select the appropriate level of model flexibility
 - K-fold (less variance); LOOCV (less bias)
 - Cross-validation on classification problem (Class imbalance)
 - Stratified cross validation: ensure that each fold has the same ratio of each class as the original data set
 - More robust metric to estimate validation error: precision, recall, f1 score
- **Bootstrap**
 - random sampling with replacement
 - Used most commonly to provide a measure of accuracy of parameter estimate / of a given ml model
 - 例子: 可以estimate standard error of coefficients → confidence interval for coefficient
 - Useful when dataset is small
 - Can help deal with class imbalance: for classes that are rare → generate new samples
- Why resampling:
 - Evaluate the generalization/performance of model, by fitting the model on different train sets generated through resampling
 - Enrich the dataset when it is small
- Sampling bias

- Selection bias
 - A problematic situation in which error is introduced due to a non-random population sample.
 - Ex. only sample users from a city
 - Undercoverage bias
 - occurs when some members of a population are inadequately represented in the sample
 - Ex. volunteer response
- Survivorship bias
 - occurs when an individual only considers the surviving observation without considering those data points that didn't survive in the event
 - Ex. when we try to get information from the product reviews, we ignore the people who didn't write a review → info from the reviews cannot tell a whole story of how people feel about this product

1.8 机器学习分类

- **基本概念**

- 学习 (learning)
 - 将数据给机器分析，以此来训练机器，培养机器给数据分类的能力。学习指的就是找到特征与标签的映射 (mapping) 关系。这样当有特征而无标签的未知数据输入时，我们就可以通过已有的关系得到未知数据标签。
- 分类 (classification) 定性输出称为分类，或者说是离散变量预测。
- 回归 (regression) 定量输出称为回归，或者说是连续变量预测。
- 聚类 (clustering) 聚类的结果将产生一组集合，集合中的对象与同集合中的对象彼此相似，与其他集合中的对象相异。

- **有监督学习 (supervised learning)**

- 不仅把训练数据给机器，而且还把分类的结果 (标签) 也一并给机器分析。
- 机器进行学习之后，再给它新的未知的数据，它也能计算出该数据导致各种结果的概率，给一个最接近正确的结果。
- 由于计算机在学习的过程中不仅有训练数据，而且有训练结果 (标签)，因此训练的效果通常不错。
- 有监督学习的结果可分为两类：分类或回归

- **无监督学习 (unsupervised learning)**

- 只给计算机训练数据，不给结果标签，所有的数据都是一样的
- 计算机无法准确地知道哪些数据具有哪些标签，只能凭借强大的计算能力分析数据的特征，从而得到一定的成果，通常是得到一些集合，集合内的数据在某些特征上相同或相似
- 无监督学习的任务是从给定的数据集中，挖掘出潜在的结构

- 非监督学习中，虽然照片分为了猫和狗，但是机器并不知道哪个是猫，哪个是狗。对于机器来说，相当于分成了 A、B 两类
- **半监督学习 (semi-supervised learning)**
 - 半监督学习训练数据的一部分是有标签的，另一部分没有标签，而没标签数据的数量常常远远大于有标签数据数量（这也是符合现实情况的）。
 - 隐藏在半监督学习下的基本规律在于：数据的分布必然不是完全随机的，通过一些有标签数据的局部特征，以及更多没标签数据的整体分布，就可以得到可以接受甚至是非常好的分类结果。
 - 半监督分类
 - 半监督分类(Semi-Supervised Classification)：是在无类标签的样例的帮助下训练有类标签的样本，获得比只用有类标签的样本训练得到的分类器性能更优的分类器，弥补有类标签的样本不足的缺陷，其中类标签 取有限离散值。
 - 半监督回归
 - 半监督回归(Semi-Supervised Regression)：在无输出的输入的帮助下训练有输出的输入，获得比只用有输出的输入训练得到的回归器性能更好的回归器，其中输出取连续值。
 - 半监督聚类
 - 半监督聚类(Semi-Supervised Clustering)：在有类标签的样本的信息帮助下获得比只用无类标签的样例得到的结果更好的簇，提高聚类方法的精度。
 - 半监督降维
 - 半监督降维(Semi-Supervised Dimensionality Reduction)：在有类标签的样本的信息帮助下找到高维输入数据的低维结构，同时保持原始高维数据和成对约束(Pair-Wise Constraints)的结构不变，即在高维空间中满足正约束(Must-Link Constraints)的样例在低维空间中相距很近，在高维空间中满足负约束(Cannot-Link Constraints)的样例在低维空间中距离很远。
- **强化学习**
 - 强化学习更接近生物学习的本质，因此有望获得更高的智能。它关注的是智能体如何在环境中采取一系列行为，从而获得最大的累积回报
 - 通过强化学习，一个智能体应该知道在什么状态下应该采取什么行为
- https://blog.csdn.net/haishu_zheng/article/details/77927525

1.9 End-to-End ML Workflow

Clarify the problem and constraints

- Dependent variable, baseline performance
- Is ML needed? Is it legal?
- How do end users benefit from the solution
- Business value, how will incorrect prediction impact business
- A full self-driving algorithm, or separate smaller algorithms?

Establish metrics

- Start your answer with a single metric, then hedge your answer by mentioning other potential metrics to track
- 举例: spam classifier: Accuracy precision & recall F-1 score
- Accuracy precision & recall F-1 score
 - Constrain recall to be at least 0.95 while optimizing for precision
 - Blending multiple metrics into one by weighting different sub-metrics, (false positive VS. false negative)

Understand data sources

- Acquire data: Can ethically scrape the data? Buy second- and third-party dataset?
- Understand data: how fresh? How often is it updated? Data dictionary? How was it collected? Bias?

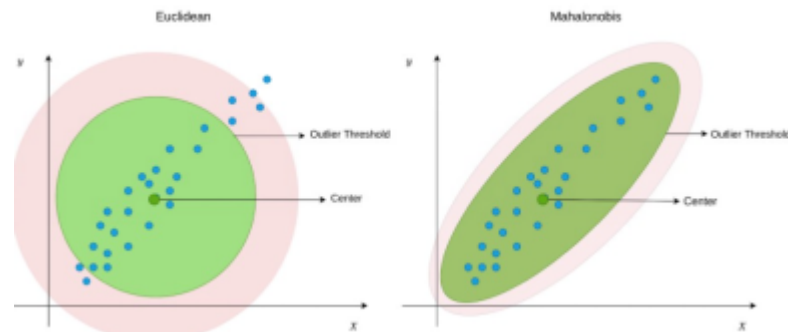
Explore your data

- Columns, mean, median, quantiles
- Visualize distribution, range of continuous variables, plot categories of categorical variables, class count
- Correlation matrix

Clean your data

- Drop duplicate, irrelevant data
- Missing value
 - 要先find root cause, 量大不大, 是否predictive, 如果是classification, 在每个class里都一样missing吗
 - **Why a problem:** can impact the performance of the model by creating a bias in the dataset. loss in values might contain crucial insights or information for model development
 - mean / median
 - 缺点是没考虑correlation with other features, 比如transaction price of a different category in different location 是不一样的
 - Model / distribution
 - Distribution: 根据avg和std随机生成
 - Model: 找correlated variable或者相关的variables → linear regression / knn
 - Check performance after imputing, 如果performance没有significantly increased, 说明可能不重要, 可以直接删掉
 - <https://towardsdatascience.com/how-to-handle-missing-data-8646b18db0d4>
- Outlier
 - Outlier主要问题是会increase variance, Outliers are those data points which differ significantly from other observations

- 出现原因: It can occur because of variability in measurement and due to misinterpretation in filling data points
- Identify 方法
 - Univariate: IQR, z-score, box plot
 - Multivariate:
 - Regression: Cook's distance
 - <https://medium.com/@jcpineda/removing-outliers-based-on-cooks-distance-8d5d8913c8eb>
 - General: Clustering, Mahalanobis distance
 - <https://towardsdatascience.com/multivariate-outlier-detection-in-python-e946cfc843b3>



- Handle: (need handle outlier with high leverage)
 - Deal with outliers themselves
 - Delete (if due to data entry/processing error and are small in numbers)
 - Transform: (to make them less abnormal) Continuous into bins/ Log transformation
 - Use metrics / models which are more robust to outliers
 - Metrics: 例如mse < mae; Metrics: 例如mse < mae
- Outlier可以是overfit的元凶, 也可以起到regularization的作用
 - Importance of outlier (hidden treasure of information) Abnormal detection
 - Fraud detection – the detection of fraudulent credit card transactions, insurance claims, expense reports or financial information.
 - Intrusion detection – the detection of unauthorized access or attempts to computer networks or systems.
 - Activity monitoring – the detection of (malicious) phone calls, messages and other forms of chatter which provides intelligence about people with bad intentions.
 - Quality control – the detection of production defects or product characteristics that do not fit the same standards as the other products that a company manufactures.
 - Image analysis – the detection of changed imagery. This can be applied to medical scans to detect certain types of diseases, or satellite imagery to detect abnormal or changed patterns.

- Pharmaceutical research – the detection of medical or patient outliers that can advance pharmaceutical or medical research
 - [Multivariate Outlier Detection in Python](#)
 - <https://medium.com/analytics-vidhya/how-to-remove-outliers-for-machine-learning-24620c4657e8>
- Multicollinearity
 - a phenomenon in which two or more input variables in the regression models are highly correlated
 - 造成的问题
 - Variance: one variable change, correlated ones change → increase variance, unstable (sensitive to minor changes in the model)
 - Unreliable coefficients & p-value: 可能change sign, 不知道individual contribution of a variable (partially explain 为啥coefficient不能直接被认为是feature importance)
 - Observe & solve
 - Observe: correlation matrix & Variance inflation factor VIF
 - <https://towardsdatascience.com/how-to-detect-and-deal-with-multicollinearity-9e02b18695f1>
 - identifies correlation between independent variables and the strength
 - 1 no correlation; 1-5 moderate; >10 strong
 - In the VIF method, we pick each feature and regress it against all of the other features. For each regression, the factor is calculated as $\frac{1}{1-R^2}$, greater the value of R-squared, greater is the VIF
 - Solve:
 - 什么时候不用处理:
 - a. Only exist among a few variables, 并且这几个变量不太影响target
 - b. Multicollinearity 只影响coefficient和p-value, 对prediction和goodness of fit影响不太大 → 如果目标是make prediction, 只要保证precision/recall足够高就行
 - drop highly-correlated variables which are not highly correlated with the target
 - linear transformation (例如加起来)
 - PCA/PLS(partial least squares)
 - Regularization: 适用于变量数多, 并且不确定最终模型有哪些变量的时候, 通过减小 coefficients来减少multicollinearity的影响
 - <https://www.linkedin.com/pulse/what-multicollinearity-how-affects-model-performance-machine-cheruku/>
- Imbalance
 - When observation in one class is higher than the observation in other classes

- 问题: Most machine learning algorithms work best when the number of samples in each class are about equal. This is because most algorithms are designed to maximize accuracy and reduce errors. However, if the data set is imbalanced then In such cases, you get a pretty high accuracy just by predicting the majority class, but you fail to capture the minority class.
- **Handle**
 - 从数据出发: 如果不是事件本身, 而是收集不均匀的话, 就collect new data;
 - 或者Resampling: Random undersampling (lose information → bias), Random oversampling (overfit), Smote Synthetic Minority Oversampling Technique, generates synthetic data for the minority class, randomly picks a point from the minority class and finds k-nearest neighbors for this point → generates a synthetic point between the chosen point and its neighbors.
 - 或者manually generate new data: 根据已有data, 人工加random error → generate new data
 - 从model出发: Metric: precision, recall, f1-score, auc
 - Improve model:logistic (调整threshold 根据0和1的比例), rf可以给class加weight, 例如给10%的class加90%的权重, ensemble模型第一步就是resample, robust to imbalance
 - Penalize the prediction error on infrequent classes (increase the cost of classification mistakes on the minority class)
- <https://www.analyticsvidhya.com/blog/2020/07/10-techniques-to-deal-with-class-imbalance-in-machine-learning/>

Feature engineering

- Quantitative data
 - Transformations
 - Log, capping, flooring to more standard distribution
 - Binning
 - Break down continuous variable into discrete bins
 - Dimensionality Reduction
 - Generate a reduced set of uncorrelated features
 - Normalize data => min/max scaling 让data的值在0-1
 - Standardize feature => z score (mean=0, std=1)
- Categorical data
 - One-hot encoding: turns each category into a vector of all zeros
 - Hashing: turns data into fixed dimensional vector using hashing function
- Scale, Standardize, Normalize
 - Scale generally means to change the range of the values
 - Shape of the distribution doesn't change. The range is often set at 0 to 1

- Standardize generally means changing the values so that the distribution's standard deviation equals one => z score (mean=0, std=1)
- Normalize can be used to mean either of the above things (and more!)
- StandardScaler
 - industry's go-to algorithm
 - standardizes a feature by subtracting the mean and dividing by the standard deviation. StandardScaler does not meet the strict definition of scale I introduced earlier
 - Std = 1, mean closes to 0
- Need to perform Feature Scaling when we are dealing with
 - Gradient Descent Based algorithms (Linear and Logistic Regression, Neural Network)
 - Distance-based algorithms (KNN, K-means, SVM)
 - as these are very sensitive to the range of the data points
- Why
 - Many machine learning algorithms perform better or converge faster when features are on a relatively similar scale and/or close to normally distributed
 - In gradient descent based: Feature value can affect step size of the gradient descent. If different ranges → different step size; To ensure gradient descent moves smoothly toward minimum and steps for gradient descent get updated at the same rate for each feature → scale
- 总结
 - Make the flow of gradient descent smoothly and help reach the minimum quickly
 - Without scaling, the algorithm may be biased toward features that have values of higher magnitude
- <https://www.enjoyalgorithms.com/blog/need-of-feature-scaling-in-machine-learning>
- Feature selection 特征选择
 - We can summarize feature selection as follows:
 - Feature Selection: Select a subset of input features from the dataset.
 - **Unsupervised:** Do not use the target variable (e.g. remove redundant variables). Correlation
 - **Supervised:** Use the target variable (e.g. remove irrelevant variables)
 - **Wrapper:** Search for well-performing subsets of features - RFE
 - **Filter:** Select subsets of features based on their relationship with the target
 - Statistical Methods / Feature Importance Methods
 - **Intrinsic:** Algorithms that perform automatic feature selection during training
 - Decision Trees
 - Dimensionality Reduction: Project input data into a lower-dimensional feature space
 - <https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/>
 - Filter 过滤法

- ANOVA: (numerical input, categorical output / reverse): Test at least one pair is different

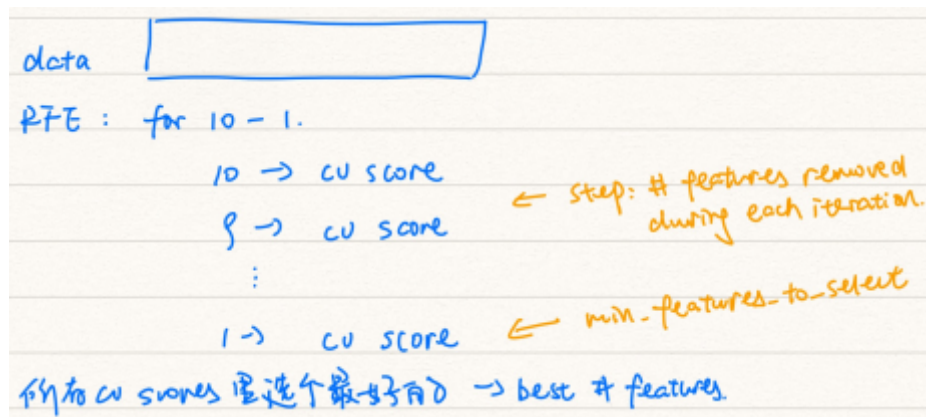
$$H_0 : \mu_1 = \mu_2 = \mu_3$$

$H_a : \text{At least 1 pair are different}$

- 比如 height → 能不能 gold, silver, blonde 看 gold 的平均 height, silver class 的平均身高, blonde 的 有没有 statistical difference → 有, reject null → 可做 feature
- Chi-square: Test whether a predictor is independent with the target, if yes, that variable should not be a feature for training

○ **Wrapper 包装法**: Narrow down # features and only include important features

- RFE
 - Select features by recursively considering smaller and smaller sets of features
 - the estimator is trained on the initial set of features
 - Obtain the importance of each feature through any specific attribute or callable
 - Tree: feature importance
 - Regression: least absolute coefficient value
 - Remove the least important features from the current set of features
 - Refit → remove: recursively repeated on the remaining set of features until the desired number of features is reached
 - <https://machinelearningmastery.com/rfe-feature-selection-in-python/>
- RFECV recursive feature elimination with cross validation



- Stepwise selection (greedy algorithm)
 - Forward Selection: start with a null model, iteratively add one best predictor everytime, until all predictors have been used; then select a single best number of predictors to get the best model.
 - Backward Selection
 - Forward-Backward Selection: alternates between forward and backward, bringing in and removing variables that meet the criteria for entry or removal, until a stable set of variables is attained

- Aim to find models with high R^2 and low RSS, 用AIC, adjusted R^2 等来考虑要多少个predictors
 - Best-subset: for each number of predictors, try all combinations to find the best; then select a single best number of predictors to get the ideal model
 - p features $\rightarrow k = 1, 2, \dots, p \rightarrow$ try each model with k predictors \rightarrow choose best subset model using a regression metric like R^2
 - Computationally expensive
 - Try every option in a large search space \rightarrow 很可能overfit with a high variance in coefficient estimates
 - **Shrinkage**: full model, shrink coefficients to zero (similar to regularization)
 - **Dimensionality Reduction**
 - **PCA, LDA**
- Categorical Variables
 - Categorical vs. categorical
 - Contingency table for two variables \rightarrow chi-square test to test the independence between two variables
 - If we assume that two variables are independent, then the values of the contingency table for these variables should be distributed uniformly. And then we check how far away from uniform the actual values are
 - Null hypothesis: they are independent
 - Alternative hypothesis is that they are correlated in some way
 - p value is 0.08 - quite small, but still not enough to reject the hypothesis of independence. So we can say that the "correlation" here is 0.08
 - 例子: 用户地址customer location vs. 产品类型
 - Categorical vs. continuous
 - Logistic: use continuous to predict categorical ones, 如果show relationship, 就 correlated
 - Anova test: 每个category, 画出continuous的分布, 再用anova看几个mean是不是 statistically different
 - <https://datascience.stackexchange.com/questions/893/how-to-get-correlation-between-two-categorical-variable-and-a-categorical-variable>
 - How to encode? 区别是什么, 哪个更好, 为什么
 - Nominal(没顺序), ordinal (有顺序)
 - Ordinal就可以用labeling encoding
 - Dummy
 - One-hot encoding
 - Group into communities, clustering...
 - Categorical variable with too many distinct values

- Reducing Cardinality by using a simple Aggregate function (classification)
 - 把frequency很高的category留着, 其他的minority合并成“other”
- Target encoding (both regression & classification)
 - Replace each category in the variable → with the mean response value given that category
 - Binary classification
 1. Find the conditional probability of the response variable being one, given that the categorical column takes a particular value
 2. Example: if you have a categorical column of city in predicting loan defaults, and the probability of a person who lives in SF defaults is 0.4, then replace “SF” with 0.4
- Binary classification
- Louvain community detection algorithm
 - A method to extract communities from large networks without setting a predetermined number of clusters like k-means

Model selection

- Training & prediction speed
- Budget
- Volume & Dimensionality of Data
- Categorical vs. Numerical features
- Explainability

Model training & evaluation

- Train-validation-test split
- Cross-validation
- Hyper-parameter tuning
 - finding the best combination of hyperparameters that gives the best performance
 - HT impacts a model's training time, compute resources needed (hence cost) and performance
 - <https://towardsdatascience.com/bayesian-optimization-for-hyperparameter-tuning-how-and-why-655b0ee0b399>
 - Grid search:
 - Form a grid == cartesian product of those parameters → then sequentially try all combinations 看which yields best results
 - Random search
 - Instead of a discrete set of values, we provide a statistical distribution for each hyperparameter → randomly sampled from the distribution

- 面试：不是特别经常问到。可能会问到in context of neural network, random forest, XGBoost...
要能list a couple of the hyperparameters for your favorite modeling technique, along with what impacts they have on generalization
- Metrics
 - Confusion matrix

		Predicted		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) Type 2 Error	Sensitivity $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) Type 1 Error	True Negative (TN)	Specificity $\frac{TN}{(TN + FP)}$
		Precision $\frac{TP}{(TP + FP)}$	Negative Predictive Value $\frac{TN}{(TN + FN)}$	Accuracy $\frac{TP + TN}{(TP + TN + FP + FN)}$

- Accuracy: Well balanced, not skewed, no class imbalance
- Precision
 - $TP/(TP+FP)$, want to be sure of our prediction
 - Cost of false positive 假阳性 is high → restaurants wants good wine
- Recall (sensitivity) 多少1中被正确预测 TPR
 - $TP/(TP+FN)$, want to capture as many positives as possible
 - Cost of FN 假阴 is high → covid这种传染病, 找出尽可能多的感染者
- Recall & Precision tradeoff
 - 高 recall => misdiagnosing who didn't have disease
 - 高 precision => missing truly disease people
- Specificity: $TN / \text{all } N$ 多少0中被正确预测
- F1 score 讨论business & product impact of false positive or false negative
 - 当precision & recall 都重要是, 可以optimize F1 score
 - Balance between precision and recall
$$F1 = \frac{2 * \text{prec} * \text{recall}}{\text{prec} + \text{recall}} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$
- Type I II Error
 - ○ I: False positive rate: fp / n
 - ○ II: False negative rate: fn / p
- When is FP more serious than FN
 - Give a \$1000 coupon to customers who are assumed to spend \$10000. If a customer cannot actually spend that much, and you give the coupon → 损失
- When is FN more serious than FP
 - 尽量提高recall, 找出每一个正样本, Security check → 放过一个恐怖分子都很dangerous
- Are FN and FP equally serious

- It's equally bad for banks to lose good customers and have bad customers
- How to choose an evaluation metric for an imbalance classification
 - First: talk to stakeholders and figure out what is important about the particular model.
 - Second: perform a literature review and discover what metrics are most commonly used by other practitioners or academics working on the same general type of problem



- ROC curve /AUC

- Roc shows the performance of a classifier at all classification thresholds
- a graphic that shows the tradeoff between the rate at which you can correctly predict something with the rate of incorrectly predicting something
 - TPR: proportion of positive data points that are correctly considered as positive $TP/all\ P$
== recall
 - FPR: proportion of negative data points that are mistakenly considered as positive $FP/all\ N$



- Upper-right 最右上方, threshold=0, all points are classified as positive
 $TP/all\ p = 1$ all positive cases are correctly considered as positive
 $FP/all\ N = 1$ all negative case are mistakenly considered as positive
- Move left: threshold increase
- lower-left 最左下方: threshold = 1, all points are classified as negative
 $Tpr = 0$ no positive case is correctly considered as positive
 $Fpr = 0$ no negative case are mistakenly considered as positive
- If negative >> positive, precision is better than FPR, because precision does not consider # of pred_n, so it isn't affected by imbalance
- Model Drift
 - 一段时间后发现模型不work了, 发生了什么, 怎么办
 - What it essentially means is that the relationship between the target variable and the independent variables changes with time. Due to this drift, the model keeps becoming unstable and the predictions keep on becoming erroneous with time
 - **Concept drift:** properties of target variable changes


- 例如原来demand是根据sales (existing)来预测的, 现在要考虑potential / new customers
 - What is considered as “fraudulent” change
- **Data drift:** properties of predictors variables changes
 - changes in the data due to seasonality, changes in consumer preferences, the addition of new products
- Model 出现的问题
 - Overfit → cannot generalize
 - Incomplete / outdated features
- 怎么办: Monitor performance and regularly refit models to proactively re-developed so as to mitigate the risks associated with drift
 - Monitor the performance of the model over time
 1. Input, output
 1. Know data issue or model issue
 2. data issue: look into what changes are causing this shift (data collection method or a genuine shift in the trend)
 3. model issue: look at what features of your model may be causing this change in distribution
 2. Features
 1. monitor a few critical features whose change in data distribution might skew the model results terribly
 2. Add additional features that might improve the model's performance and make it more solid and accurate
 3. Prediction quality (different metrics)
 - Regularly refit models using new data to maintain accuracy
 - E-commerce: weekly
 - Use an ensemble of models
 - use multiple algorithms simultaneously and combine their predictions into one final prediction that can be more accurate
- For time series:
 - For situations where the data changes with time, weighing data can be a good option → give more weight to most recent data and less weight to historical data
- How to maintain a deployed model?
 - Monitor(changes), Evaluate (metrics), Rebuild
- <https://towardsdatascience.com/why-machine-learning-models-degrade-in-production-d0f2108e9214>

1.10 生成模型与判别模型

- 监督学习的任务就是从数据中学习一个模型（也叫分类器），应用这一模型，对给定的输入 X 预测相应的输出 Y 。
 - 这个模型的一般形式为**决策函数** $Y=f(X)$ 或者**条件概率分布** $P(Y|X)$ 。
- 决策函数 $Y=f(X)$: X 对应 Y ，比较 Y 与阈值，根据结果判定 X 属于哪个类别。

条件概率分布 $P(Y|X)$: 你输入一个 X ，它通过比较它属于所有类的概率，然后输出概率最大的那个作为该 X 对应的类别。
- 监督学习方法分 **生成方法 (Generative approach)** 和 **判别方法 (Discriminative approach)**
所学到的模型分别称为 **生成模型 (Generative Model)** 和 **判别模型 (Discriminative Model)**
- Discriminative models 判别模型
 - 由数据直接学习决策函数 $Y=f(X)$ 或者条件概率分布 $P(Y|X)$ 作为预测的模型，即**判别模型 (Discriminative model)**
 - 判别模型寻找不同类别之间的最优分类面，反映的是异类数据之间的差异
 - $y = \operatorname{argmax} P(T = k|x)$
 - **常见判别模型有** logistic regression, SVMs, traditional neural networks, Nearest neighbor, Conditional random fields(CRF)
- Generative models 生成模型
 - 由数据学习联合概率密度分布 $P(X,Y)$ ，然后求出条件概率分布 $P(Y|X)$ 作为预测的模型，即**生成模型 (Generative Model)**
 - Joint distribution of X & Y : $P(Y|X) = P(X,Y) / P(X)$
 - 对后验概率建模，从统计的角度表示数据的分布情况，能够反映同类数据本身的相似度
 - 基本思想是首先建立样本的联合概率密度模型 $P(X,Y)$ ，然后再得到后验概率 $P(Y|X)$ ，再利用它进行分类
 - **常见生成模型有** Gaussians, Naive Bayes, Mixtures of Gaussians, Mixtures of experts, Hidden Markov Models (HMM), Sigmoidal belief networks, Bayesian networks, Markov random fields
- 由生成模型可以得到判别模型，但由判别模型得不到生成模型
- $P(X)$ 称为**先验概率**， $P(Y|X)$ 称为**后验概率**
- <https://blog.csdn.net/zouxy09/article/details/8195017>
<https://www.cnblogs.com/zeze/p/7047630.html>

1.11 Linear regression 线性回归

-  Quick runtime, Easy to Interpret
 - X : matrix of predictor variables
 - β : vector of parameters that determines the weight of each variable
- Assumptions

- Linearity: relationship between target variable and feature set is linear
- Homoscedasticity 同方差性: variance of residuals is constant (points are about the same distance from the line)
- Independence: all observations are independent of one another
- Normality: Y is normally distributed
- Evaluating LR
 - Evaluation of LR based on residuals: distance between predicted and actual
 - Residual sum of squares (RSS)
 - LR estimates β by 最小化 (RSS)

$$RSS = \sum_{i=1}^n (y_i - f(x_i))^2$$
 - TSS = RSS + ESS(explained)
 - (Adjusted) R^2
 - measures how much variability in dependent variable can be explained by the model

$$R^2 = 1 - \frac{SS_{Regression}}{SS_{Total}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$
 - R^2 does not consider the overfitting problem \rightarrow adjusted $R^2 \rightarrow$ penalizes additional independent variables added to the model and adjusts the metric to prevent overfitting

$$(Adjusted)R^2 = 1 - \frac{(1-R^2)(N-1)}{N-p-1}$$
 - MSE, RMSE, MAE

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$
 - Root RMSE \rightarrow used more commonly because sometimes MSE is too big to compare easily
 - MAE (mean absolute error): more direct representation of error
 - MSE measures the variance of the residuals, MAE measures the average of residuals
 - MSE gives larger penalization to big prediction errors by squaring it while MAE treats all errors the same
 - Loss function
 - Mean squared error $MSE = \frac{1}{N} \sum_{i=1}^N (y_i - y_{\hat{a}ti})^2$
 - Gradient descent 过程 / Implementation
 1. Initially let $m = 0$ and $c = 0$. L = learning rate (which controls how much the value of m changes with each step)
 2. Calculate the partial derivative of the loss function with respect to m , and plug in the current values of x , y , m and c in it to obtain the derivative value D
 3. update the current value of m and c using the following equation

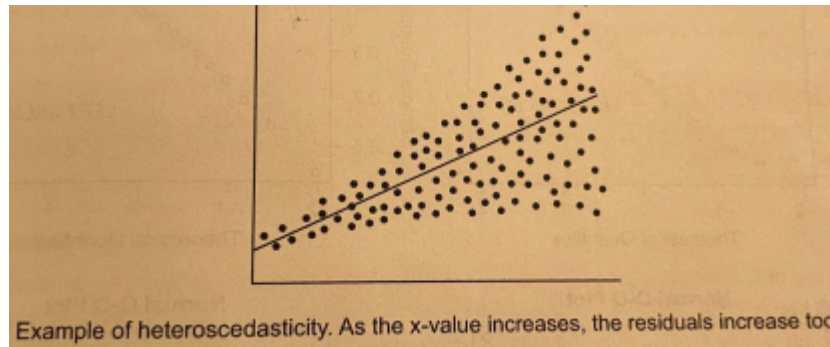
$$m = m - L \times D_m$$

$$c = c - L \times D_c$$

- Avoid LR Pitfalls

- Heteroscedasticity: residuals are not identically distributed

- Plot: residuals vs fitted values



- Normality: residuals are normally distributed

- Test through QQ (quantile) plot: standardized residuals vs. theoretical quantiles

- <https://towardsdatascience.com/q-q-plots-explained-5aa8495426c0>

- Outliers

- Identify outliers → Cook's distance (cd)

- Cd is an estimate of the influence of a data point, by 考虑 residual and leverage (how far away the X value differs from that of other observations) of every point. → remove points when cd > threshold

- <https://towardsdatascience.com/identifying-outliers-in-linear-regression-cooks-distance-9e212e9136ae-9e212e9136ae>

- Multicollinearity

- Predictors are correlated

- Observe: Variance inflation factor VIF

- Solve: Remove correlated variables, linear combination, PCA/PLS(partial least squares)

- <https://towardsdatascience.com/how-to-detect-and-deal-with-multicollinearity-9e02b18695f1>

- Confounding Variables

- Extraneous variable in a statistical model that correlates directly or inversely with both the dependent variable and independent variable. The estimate fails to account for the confounding factor

- Multicollinearity is an extreme case. Confound的方式有很多

- Selection bias: data are biased due to the way they were collected

- Omitted variable bias: 删掉了重要的variables

- Handle:

- Stratification 分层: create multiple categories or subgroups in which confounding var do not vary much → test significance and strength of associations using chi-square
- <https://sphweb.bumc.bu.edu/otlt/MPH-Modules/PH717-QuantCore/PH717-Module11-Confounding-EMM/PH717-Module11-Confounding-EMM6.html>
- Generalized Linear Models
 - Allows for residuals not normally distributed
 - Can use weights and predictors nonlinearly, ex. Polynomial regression...
- <https://towardsdatascience.com/linear-regression-using-gradient-descent-97a6c8700931>

1.12 Logistic regression 逻辑回归

- logistic回归, 又叫对数几率回归, 是一个分类模型
- logistic回归和线性回归的关系
 - 线性回归模型为 $f(x) = w_0x_0 + w_1x_1 + \dots + w_nx_n + b$
 - 向量形式为 $f(x) = w^T x + b$
 - 广义线性回归模型为 $y = g(w^T x + b)$
 - 我们可以从线性回归的回归模型引出logistic回归的分类模型: 利用广义线性回归“模型
只需找一个单调可微函数将分类任务的真实标记y与线性回归模型的预测值联系起来便可以
 - logistic回归是处理二分类问题的, 所以输出的标记y={0,1}; 线性回归模型产生的预测值z=wx+b是一个实值, 所以我们将实值z转化成0/1值便可
 - 引入**sigmoid**函数: $g(x) = \frac{1}{1+e^{-x}}$, which 取值在[0, 1]之间, 在远离0的地方函数的值会很快接近0/1
 - 在原来的线性回归模型外套上sigmoid函数便形成了logistic回归模型的预测函数: $y = \frac{1}{1+e^{-(w^T x + b)}}$
- 梯度上升算法求解logistic回归模型参数w
 - 变换预测函数为对数几率: y视为样本x正例可能性, 1-y是反例可能性

$$\ln \frac{y}{1-y} = w^T x + b$$

- 将式子中的y视为类后验概率估计 $p(y=1|x)$:

$$\ln \frac{p(y=1|x)}{p(y=0|x)} = w^T x + b$$

$$p(y=1|x) = \frac{e^{w^T x + b}}{1 + e^{w^T x + b}} = h_w(x)$$

$$p(y=0|x) = \frac{1}{1 + e^{w^T x + b}} = 1 - h_w(x)$$

- 定义准则函数:

$$J(w) = \frac{1}{n} \sum_{i=1}^m (y^i \ln h_w(x^i) + (1 - y^i) \ln (1 - h_w(x^i)))$$

- 使用梯度上升算法求解参数w, 因此参数w的迭代式为

$$w_{j+1} = w_j + \alpha \nabla J(w_j)$$

- 梯度上升算法：
 - 梯度上升算法和我们平时用的梯度下降算法思想类似，梯度上升算法基于的思想是：要找到某个函数的最大值，最好的方法是沿着这个函数的梯度方向探寻！直到达到停止条件为止
- 随机梯度上升算法：
 - 梯度上升算法在每次更新回归系数时都需要遍历整个数据集，该方法在处理小数据时还尚可，但如果具有数十亿样本和成千上万的特征，那么该方法的计算复杂度太高了，改进方法便是一次仅用一个数据点来更新回归系数，此方法便称为随机梯度上升算法
 - 由于可以在更新样本到来时对分类器进行增量式更新，因而随机梯度上升算法是一个“在线学习算法”。而梯度上升算法便是“批处理算法”
- https://tech.meituan.com/intro_to_logistic_regression.html
https://blog.csdn.net/jediael_lu/article/details/77852060
http://blog.csdn.net/feilong_csdn/article/details/64128443

1.12 Logistic regression 逻辑回归

- Logistic regression is a binary classifier that models the conditional probability that Y belongs to a particular category given X (output is in the range of 0 and 1)
- learns a linear relationship and then introduces a non-linearity in the form of the Sigmoid function (natural logarithm of the “odds” of the target variable)
- **Logistic function:** the probability of being class 1 given X

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$

- Odds: probability of an event divided by the probability of its complement

$$\frac{p(X)}{1-p(X)} = 1 + e^{\beta_0 + \beta_1 X}$$

- $0 \rightarrow \text{inf}$: low \rightarrow high probability

- Log-odds / logit

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X$$

- Logit is linear in X

- Odds: number of true / number of false \rightarrow Once it has the log(odds), we convert that value to a probability by using a logistic function in order to make predictions

- Fit / Estimate coefficients $(\beta_0, \beta_1 \dots) \rightarrow$ Maximum likelihood

- Likelihood function:



$$l(\beta_0, \beta_1) = \prod_{i: y_i=1} p(x_i) \prod_{i': y_{i'}=0} (1 - p(x_{i'}))$$

- Joint probability (product of probabilities)
- Want to maximize the probability of predicted values being close to the actual ones
 - When class = 0, we want probability to be as small as possible
 - When class = 1, want prob as large as possible
 - Ideally, Product of probability of class 1 and 1-prob of class 0 should be maximized


- Maximize likelihood → maximize log likelihood → minimize the negative log likelihood (log loss)
- Method of maximization
 - Gradient methods
 - Minimize log loss / binary cross-entropy loss function

$$J(\beta) = -\frac{1}{m} \sum_{i=1}^m [y_i \log(p(x_i)) + (1 - y_i) \log(1 - p(x_i))]$$
 - Start with random guess → compute gradient and update betas → until converge

$$\frac{\delta}{\delta \beta_j} J(\beta) = \frac{1}{m} \sum_{i=1}^m (p(x_i) - y_i) x_{ij}$$
 - i: index of data points; j: index of features
 - Expectation maximization
- Evaluate: Modify loss function to handle imbalance
 - Add class weight: $\text{logloss} = \frac{1}{N} \sum_{i=1}^N [-(w_0(y_i \log(\hat{y}_i))) + (w_1((1 - y_i) \log(1 - \hat{y}_i)))]$
 - $w_j = n_{\text{samples}} / (n_{\text{classes}} * n_{\text{samples}})$
 - w_j is the weight for each class (j signifies the class)
 - N_{samples} is the total number of samples or rows in the dataset
 - N_{classes} is the total number of unique classes in the target
 - N_{samples}_j is the total number of rows of the respective class
- Implementation



- <https://medium.com/analytics-vidhya/logistic-regression-with-gradient-descent-explained-machine-learning-a9a12b38d710>

Multiclass Logistic regression

- Multinomial logistic regression
 - extension to the logistic regression model that involves changing the loss function to cross-entropy loss and predicting probability distribution to a multinomial probability distribution to natively support multi-class classification problems
 - loss function: log loss → cross entropy loss
 - Output: a single probability value → one probability for each class label
- 

 - Evaluate: stratified k-fold cross validation
- Other extensions:
 - split the multi-class classification dataset into multiple binary classification datasets and fit a binary classification model on each. Two different examples of this approach are the One-vs-Rest and One-vs-One strategies.
 - One-vs-Rest
 - One-vs-One

- <https://machinelearningmastery.com/multinomial-logistic-regression-with-python/>

Cross Entropy

- Usually used as a loss function when optimizing classification models
 - Softmax converts logits into probabilities. Cross-Entropy takes the output probabilities (P) and measures the distance from the truth values
- Entropy: Entropy of a random variable X is the level of uncertainty inherent in the variable's possible outcome





- Reason for negative sign: $\log(p(x)) < 0$ for all $p(x)$ in $(0,1)$
 - Larger entropy \rightarrow greater uncertainty for probability distribution
- Log loss, binary cross entropy
 - Compare each predicted class probability with the actual class (0 or 1)
 - Calculate a score that penalizes the probability based on how far it is from the actual expected value
 - The penalty is $\log p(x) \rightarrow$ large score for large differences with 1. 距离1越远, $\log p(x)$ 越大
- Cross-entropy
- <https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e>

面试问题

- Logistic loss 损失函数
 - $\frac{1}{n} \sum_{i=1}^n \ln(1 + e^{-y_i w^t x_i}) = \frac{1}{n} \sum_{i=1}^n -y \log(y') - (1 - y) \log(1 - y')$

1.13 SVM 支持向量机

- SVM is a supervised ml algorithm that is mostly used in classification problems. It performs classification by finding the hyper-plane that separates the two classes in feature space
- If I cannot \rightarrow 1). Soften what we mean by “separates”; 2) enrich and enlarge the feature space so that separation is possible
 - Support Vectors are simply the coordinates of individual observation
 - An SVM classifier is a frontier that best segregates the two classes (hyper-plane/ line)
- The SVM kernel is a function that takes a low-dimensional input space and transforms it to a higher dimensional space (i.e it converts a non-separable problem to a separable problem)
 - Simply put, it does some extremely complex data transformations, then finds out the process to separate the data based on the labels or outputs you’ve defined
 - RBF
 - Gaussian kernels
 - SVR: Our best fit line is the hyperplane that has a maximum number of points
- Loss function
 - Hinge loss: loss + penalty

- gives some punishment to both incorrect predictions and those close to decision boundary ($0 < \theta^T x < 1$)
- 
- 
- C bounds the sum of slack variable ε_i which tells us where the i th observation is located, relative to the margin and hyperplane
 - if i th observation on the wrong side of the margin, $\varepsilon_i > 0$
 - wrong side of hyperplane, $\varepsilon_i > 1$
- If C is large \rightarrow more tolerance to the violation to the margin \rightarrow margin widen
- <https://towardsdatascience.com/optimization-loss-function-under-the-hood-part-iii-5dff33fa015d>
- Hyperparameters
 - C and Gamma (for RBF kernel)
 - <https://towardsdatascience.com/hyperparameter-tuning-for-support-vector-machines-c-and-gamma-parameters-6a5097416167>
- When to use:
 - Nonlinear decision boundaries
 - Smaller amount of data
 - Interpretability is not important
- 优缺点
 - Pros
 - Perform well in high dimension ($n > p$)
 - Best when classes are separable
 - Outliers have less impact
 - It uses a subset of training points in the decision function (support vectors) \rightarrow memory efficiency.
 - Cons
 - Slow for large dataset
 - Poor performance with overlapped classes
 - Select appropriate hyperparameters and kernel function is important

1.14 Naïve Bayes 朴素贝叶斯

- Naive Bayes is a classification technique based on Bayes' Theorem with an assumption of independence among predictors
 - assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature



- It works by calculating the posterior probability, which is the conditional probability for an observation belonging to a class given X , and then classifying the observation to the class with highest posterior probability
 - $P(c|x)$ is the posterior probability of class c given predictors.
 - $P(c)$ is the prior probability of the class.
 - $P(x|c)$ is the likelihood which is the probability of predictors given a class.
 - $P(x)$ is the prior probability of predictors
- 优缺点
 - Pros
 - Fast → real-time predictions
 - Scalable with large datasets and high dimension ($p > n$)
 - Multiclass prediction
 - Insensitive to irrelevant features:
 - If X_i is an irrelevant attribute then $P(X_i/Y)$ becomes almost uniformly distributed.
 - The class conditional probability for X_i has no impact on overall computation of posterior probability.
 - Cons
 - Assume independence of features
 - Bad estimator → predict_proba output should not be taken seriously
 - Training data should represent population well; if you have no occurrence of a class label or certain attribute value → 0
- Application:
 - Text classification
 - Spam filtering
 - Sentiment analysis
 - Recommendation (what will the user buy next)

面试问题

- 解释贝叶斯深度网络，并说明其优缺点
- 贝叶斯分类的前提假设

1.15 TREE-BASED 树相关

Classification tree

- Impure: gender
 - Work alcoholic (50,50) → impure
 - Pregnant (0, 80) → pure
- To quantify the impurity of the leaves: (最pure最能区分的在root)
 - **Entropy, information gain**
 - Entropy
 - measure of uncertainty and randomness (impurity)
 - how much variance the data has
 - For a discrete variable Y:
High entropy => distribution closer to uniform than a skewed one
Information Gain $IG(Y, X) = H(Y) - H(Y|X)$
 - 表现好 entropy低
 - Information Gain: concept of a decrease in entropy after splitting the data on a feature.
The greater the information gain, the greater the decrease in entropy or uncertainty
 - **Gini impurity**
 - For categorical $GI = 1 - [P_+^2 + P_-^2]$
 - Total gini impurity for a variable = weighted average for each leaves
 - $1 - (\text{probability of yes})^2 - (\text{probability of no})^2$
 - For numerical
 - Sort the column, from lowest to highest → calculate average for all adjacent values → calculate gini for each average → choose the average with lowest gini → get gini for this variable
 - 所有variable一起比较, gini impurity最低的做root

Decision tree

- a type of supervised ml used to categorize or make predictions based on how a previous set of questions were answered
 - Core idea: Divide the feature space into several regions → have an average value or a same class for data in that area
 - 步骤: Search for a split that maximizes the “separation” of the classes → Stop when you meet some stopping criteria → Clean up the tree when you went too far doing splits (Pruning)
- Node Splitting
 - **A process of dividing a node into multiple sub-nodes** to create relatively pure nodes
 - Continuous target variable → Reduction in variance
 - For each split, 计算variance of each child node

- Calculate the variance of each split as the weighted average variance of child nodes
 - Select the split with the lowest variance
 - Perform steps 1-3 until completely homogeneous (variance=0) nodes are achieved
- Categorical target → Information gain (1-Entropy)
 - Entropy ~ purity of a node. Lower entropy, high purity. Homogeneous (entropy=0)
 - For each split, calculate the entropy of each child node
 - Calculate the entropy of each split as the weighted average entropy of child nodes
 - Select the split **with the lowest entropy** (highest information gain)
 - Until you achieve homogeneous nodes, repeat steps 1-3
- Prevent overfitting
 - Pre-pruning
 - Post-pruning
- Outliers
 - Outliers literally don't matter to trees because what makes an outlier an outlier (using the common meaning) is the distance from other data points.
 - Trees do not care about distance. They only care if the data is on one side or the other of a split. It doesn't matter at all how far from the split the data point is.
- 优缺点
 - 优点
 - Easy to interpret and visualize
 - Less efforts on feature engineering: Can handle missing values and outliers; Irrelevant features won't affect
 - 缺点
 - Prone to overfit
 - Sensitive to data (if data changes slightly, outcome can change to a large extent)
- Application:
 - Identify buyers for products, features that are most important to retain customers, fault diagnosis of machines

Random Forest

- It is a group of classification or regression trees randomly selected from a training set of the data
 - Prediction: majority votes in classification problems and averaging in regression problems
- Trees的缺点是inaccuracy. They work great with the data used to create them, but not flexible when it comes to classifying new samples → RF combines the simplicity of decision trees with flexibility
- 非专业解释: 是否能被hire → 需要several interviewer decide together
 - Step 1: Create a bootstrapped dataset, 从original dataset里选出一样size的dataset, rows可重复

- Step 2: create a decision tree using the bootstrapped dataset, but only use a random subset of variables at each step
- Aggregate the result: Bootstrapping the data plus using the aggregate to make a decision is called bagging
- Evaluate:
 - 没被bootstrap选中的entries是out-of-bag data, 把out-of-bag sample用所有没有用到他们的tree里run
 - Measure the accuracy by "the proportion of out-of-bag samples that were correctly classified by the RF"
 - Out-of-bag error: proportion of out-of-bag samples that were incorrectly classified (1-accuracy)
- Summary
 - 怎么决定每轮用多少个variable, 用accuracy来比较
 - Typically, $\sqrt{\text{numberVariables}}$, and then try a few settings above and below that value
- Feature importance: how much each feature contributes to decreasing the weighted impurity by averaging the decrease in impurity over trees
- RF Hyperparameters
 - n_estimators : number of decision trees to be created. Higher → perform better → computation cost
 - max_features: max number of features to be considered while bootstrapping.
 - max_depth: maximum extent to which our trees can split. High: overfit. Low: hamper the training process.
 - Ccp_alpha: complexity parameter for cost complexity pruning $\alpha|T|$

面试问题

- 具体说明一下决策树如何划分, 写出相应的公式
 - Gini Impurity $(1-p^2)$, Information Gain $(-p \log p)$
- 决策树将一个特征全部乘以2会有什么影响
 - 如果特征是连续的, 将其全部乘以2会改变特征的值范围。这可能导致划分点的位置发生变化, 因为决策树选择划分点时通常基于某个阈值
 - 如果特征是离散的, 将其全部乘以2会导致特征值发生改变, 可能使得原本不同的特征值变得相同, 或者原本相同的特征值变得不同
- 决策树的分裂方式
 - 基于二元划分, 多叉划分, 基于信息增益或基尼不纯度, CART算法
 - id3, gini, gdbt, xgboost

1.16 常见聚类方法

- 聚类就是按照某个特定标准(如距离准则, 即数据点之间的距离)把一个数据集分割成不同的类或簇, 使得同一个簇内的数据对象的相似性尽可能大, 同时不在同一个簇中的数据对象的差异性也尽可能地大。聚类后同一类的数据尽可能聚集到一起, 不同类数据尽量分离。
- k-means聚类算法
 - k-means算法目标是, 以k为参数, 把n个对象分成k个簇, 使簇内具有较高的相似度, 而簇间的相似度较低。
 - k-means算法的处理过程如下: 首先, 随机地选择k个对象, 每个对象初始地代表了一个簇的平均值或中心;对剩余的每个对象, 根据其与各簇中心的距离, 将它赋给最近的簇;然后重新计算每个簇的平均值。这个过程不断重复, 直到准则函数收敛
 - 采用平方误差准则, 其定义如下: $E = \sum_{i=1}^k \sum_{p \in C_i} (p - m_i)^2$
 - 这里E是数据库中所有对象的平方误差的总和, p是空间中的点, m_i 是簇 C_i 的平均值[9]。该目标函数使生成的簇尽可能紧凑独立, 使用的距离度量是欧几里得距离
 - 步骤:
 - 任意选择k个对象作为初始的簇中心;
 - repeat根据簇中对象的平均值, 将每个对象(重新)赋予最类似的簇;
 - 更新簇的平均值, 即计算每个簇中对象的平均值;
 - until不再发生变化。
 - 优点: 简单直接, 在低维数据集上有不错的效果
 - 缺点: 对于高维数据, 其计算速度十分慢, 主要是慢在计算距离上; 它的另外一个缺点就是它需要我们设定希望得到的聚类数k, 若我们对于数据没有很好的理解, 那么设置k值就成了一种估计性的工作。
- 层次聚类算法
 - 根据层次分解的顺序是自底向上的还是自上向下的, 层次聚类算法分为凝聚的层次聚类算法和分裂的层次聚类算法
 - 凝聚型层次聚类的策略是先将每个对象作为一个簇, 然后合并这些原子簇为越来越大的簇, 直到所有对象都在一个簇中, 或者某个终结条件被满足。绝大多数层次聚类属于凝聚型层次聚类, 它们只是在簇间相似度的定义上有所不同。
 - 四种广泛采用的簇间距离度量方法: 最大距离, 最小距离, 平均值距离, 平均距离
 - 最小距离的凝聚层次聚类算法流程:
 - 将每个对象看作一类, 计算两两之间的最小距离;
 - 将距离最小的两个类合并成一个新类;
 - 重新计算新类与所有类之间的距离;
 - 重复(2)、(3), 直到所有类最后合并成一类。
 - 优点: 距离和规则的相似度容易定义, 限制少; 不需要预先制定聚类数; 可以发现类的层次关系(在一些特定领域如生物有很大作用);
 - 缺点: 计算复杂度太高(考虑并行化); 奇异值也能产生很大影响; 算法很可能聚类成链状(一层包含着一层);

- SOM聚类算法
 - SOM神经网络是由芬兰神经网络专家Kohonen教授提出的
 - 该算法假设在输入对象中存在一些拓扑结构或顺序，可以实现从输入空间(n维)到输出平面(2维)的降维映射，其映射具有拓扑特征保持性质,与实际的大脑处理有很强的理论联系。
 - SOM网络包含输入层和输出层。输入层对应一个高维的输入向量，输出层由一系列组织在2维网格上的有序节点构成，输入节点与输出节点通过权重向量连接。学习过程中，找到与之距离最短的输出层单元，即获胜单元，对其更新。同时，将邻近区域的权值更新，使输出节点保持输入向量的拓扑特征。
 - SOM算法流程：
 - 网络初始化，对输出层每个节点权重赋初值；
 - 将输入样本中随机选取输入向量，找到与输入向量距离最小的权重向量；
 - 定义获胜单元，在获胜单元的邻近区域调整权重使其向输入向量靠拢；
 - 提供新样本、进行训练；
 - 收缩邻域半径、减小学习率、重复，直到小于允许值，输出聚类结果。
- FCM聚类算法
 - FCM算法是一种无监督的模糊聚类方法，在算法实现过程中不需要人为的干预。
 - FCM算法是一种以隶属度来确定每个数据点属于某个聚类程度的算法。该聚类算法是传统硬聚类算法的一种改进。
 - FCM算法流程：
 - 标准化数据矩阵；
 - 建立模糊相似矩阵，初始化隶属矩阵；
 - 算法开始迭代，直到目标函数收敛到极小值；
 - 根据迭代结果，由最后的隶属矩阵确定数据所属的类，显示最后的聚类结果。
- 优点：相比起前面的“硬聚类”，FCM方法会计算每个样本对所有类的隶属度，若某样本对某类的隶属度在所有类的隶属度中具有绝对优势，则该样本分到这个类是一个十分保险的做法，反之若该样本在所有类的隶属度相对平均，则需要其他辅助手段来进行分类。
- 缺点：KNN的缺点基本它都有。
- http://blog.csdn.net/alex_luodazhi/article/details/47125149

面试问题

- k-means的k值该如何确定
 - elbow method，x轴为聚类的数量，y轴为WSS

1.17 集成学习

Boosting、Bagging、Stacking

- **集成学习 (Ensemble learning)** 通过组合几种模型来提高机器学习的效果, 与单一模型相比, 该方法可以提供更好的预测结果
- **集成方法**是将几种机器学习技术组合成一个预测模型的元算法, 以达到减小方差 (bagging)、偏差 (boosting) 或改进预测 (stacking) 的效果
- 集合方法可分为两类:
 - **序列集成方法**, 其中参与训练的基础学习器按照顺序生成 (例如 AdaBoost), 序列方法的原理是利用基础学习器之间的依赖关系。通过对之前训练中错误标记的样本赋值较高的权重, 可以提高整体的预测效果
 - **并行集成方法**, 其中参与训练的基础学习器并行生成 (例如 Random Forest), 并行方法的原理是利用基础学习器之间的独立性, 通过平均可以显著降低错误
 - 大多数集成方法使用单一基础学习算法来产生同质的基础学习器, 即同质集成; 还有一些使用异构学习器的方法, 即异构集成
- **Boosting**
 - 个体学习器间存在强依赖关系、必须**串行**生成的序列化方法。主要关注**降低偏差** (bias)。典型算法: AdaBoost
 - 先从初始训练集训练出一个基学习器, 根据基学习器的表现**调整训练样本的权值分布**, 使得先前基学习器做错的训练样本在后续获得更多关注 (更大的权值)
 - each model tries to compensate for the weaknesses of its predecessor (learn from errors), 随着每次新模型迭代, 先前模型中错误分类数据的权重都会增加→帮助算法识别需要关注的参数以提高其性能
 - 使用基于调整后的样本训练下一个基学习器, 如此反复进行, 直至基学习器数目达到事先指定的值T, 最终将这T个基学习器进行加权结合
 - 通过加法模型将弱分类器进行线性组合, 比如**AdaBoost**通过加权多数表决的方式, 即增大错误率小的分类器的权值, 同时减小错误率较大的分类器的权值。而**提升树**通过拟合残差的方式逐步减小残差, 将每一步生成的模型叠加得到最终模型
- **Bagging (Bootstrap Aggregation)**
 - 个体学习器不存在强依赖关系、可同时生成的**并行化**方法
 - 基于**自助采样法 (Bootstrap sampling)**, 从原始样本集中抽取训练集。每轮从原始样本集中使用Bootstrapping的方法抽取n个训练样本 (在训练集中, 有些样本可能被多次抽取到, 而有些样本可能一次都没有被抽中)。共进行k轮抽取, 得到k个训练集。(k个训练集之间是相互独立的)
 - 每次使用一个训练集得到一个模型, k个训练集共得到k个模型
 - 结合方法一般为: 对分类任务使用简单投票法, 对回归任务使用均值作为最后的结果
 - 主要**降低方差** (variance) => 有可能overfit
- **Bagging, Boosting二者之间的区别**
 - 样本选择上
 - Bagging: 训练集是在原始集中有放回选取的, 从原始集中选出的各轮训练集之间是独立的

- Boosting: 每一轮的训练集不变, 只是训练集中每个样例在分类器中的权重发生变化。而权重是根据上一轮的分类结果进行调整
- 样例权重:
 - Bagging: 使用均匀取样, 每个样例的权重相等
 - Boosting: 根据错误率不断调整样例的权值, 错误率越大则权重越大
- 预测函数:
 - Bagging: 所有预测函数的权重相等
 - Boosting: 每个弱分类器都有相应的权重, 对于分类误差小的分类器会有更大的权重
- 并行计算:
 - Bagging: 各个预测函数可以并行生成
 - Boosting: 各个预测函数只能顺序生成, 因为后一个模型参数需要前一轮模型的结果



• Stacking


- Stacking 通过元分类器或元回归聚合多个分类或回归模型
- 基础层次模型 (level model) 基于**完整的训练集**进行训练, 然后元模型基于**基础层次模型的输出**进行训练
- 基础层次通常由不同的学习算法组成, 因此 stacking 集成通常是异构的。
- 例如stacking 集成由 k-NN、随机森林和朴素贝叶斯基基础分类器组成, 它的预测结果由作为元分类器的 Logistic 回归组合。
- stacking 能够实现比单个分类器更高的准确率, 并且没有显示过拟合的迹象
- Reduce variance and bias

• 随机森林

- 随机森林 (Random Forest, RF) 是Bagging的一个扩展变体。RF在以决策树为基学习器构建Bagging集成的基础上, 进一步在决策树的训练过程中引入了随机属性选择
- 比bagging多了一步 random subset of features, 工作: bootstrapped data set → random subset of features → fit model → aggregate the results
- 随机森林是由很多决策树构成的, 不同决策树之间没有关联。当我们进行分类任务时, 新的输入样本进入, 就让森林中的每一棵决策树分别进行判断和分类, 每个决策树会得到一个自己的分类结果, 决策树的分类结果中哪一个分类最多, 那么随机森林就会把这个结果当做最终的结果。
- 随机抽样构造决策树, 随机选取属性构造分裂节点, 建造大量决策树形成森林
- <https://easyai.tech/ai-definition/random-forest/>

Gradient Boosting

- Regression:
 - Start with a leaf that is the avg value of the target variable
 - Add a tree based on the residuals
 - Scale the tree's contribution to the final prediction with a learning rate
 - Keep adding trees based on the errors made by previous tree

- Classification
 - Start with a leaf that is the log(odds) → convert it into probability
 - Calculate the residuals → add a tree to predict residuals
 - Transform residual

 - Scales learning rate → get a new log odds prediction → probability
 - 继续residual → predict on residuals
 - The process repeats until we have made the maximum number of trees specified, or the residuals get super small
- After each iteration, we need to be closer to our final model. In other words, each iteration should reduce the value of our loss function
- <https://blog.paperspace.com/gradient-boosting-for-classification/>

GBDT 梯度提升树

- **GBDT (Gradient Boosted Decision Trees)**
 - GBDT也是集成学习Boosting家族的成员，GBDT也是迭代，使用了前向分布算法（Forward Stagewise Algorithm），但是弱学习器限定了只能使用CART回归树模型，同时迭代思路和Adaboost也有所不同
 - CART是“Classification and Regression Tree”的缩写。“CART回归树”特指一种以二叉树为逻辑结构的，用于完成线性回归任务的决策树
 - 在GBDT的迭代中，假设我们前一轮迭代得到的强学习器是 $f_{t-1}(x)$ ，损失函数是 $L(y, f_{t-1}(x))$ ，我们本轮迭代的目标是找到一个CART回归树模型的弱学习器 $h_t(x)$ ，让本轮的损失函数 $L(y, f_t(x)) = L(y, f_{t-1}(x) + h_t(x))$ 最小。也就是说，本轮迭代找到决策树，要让样本的损失尽量变得更小。
- **GBDT的负梯度拟合**
 - 用损失函数的负梯度来拟合本轮损失的近似值，进而拟合一个CART回归树。第t轮的第i个样本的损失函数的负梯度表示为：

$$r_{ti} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{t-1}(x)}$$
 - 我们可以拟合一颗CART回归树，得到了第t颗回归树，针对每一个叶子节点里的样本，我们求出使损失函数最小，也就是拟合叶子节点最好的的输出值
 - 这样我们就得到了本轮的决策树拟合函数，从而得到本轮最终的强学习器
 - 通过损失函数的负梯度来拟合，我们找到了一种通用的拟合损失误差的办法，这样无论是分类问题还是回归问题，我们通过其损失函数的负梯度的拟合，就可以用GBDT来解决我们的分类回归问题。区别仅仅在于**损失函数不同导致的负梯度不同**而已

XGBoost

- XGboost (Optimized Distributed Gradient Boosting) with improved efficiency, accuracy, and scalability
- Regression
 - Start with initial prediction
 - Residuals → similarity score (if residuals are similar → cannot cancel out → large)
 - $Similarityscore = \frac{SumofResidualSquare}{NumberofResidual+lambda}$
 - Lambda reduces the prediction's sensitivity to individual observations
 - Calculate Gain to determine how to split the data
 - Gain = left similarity + right - root
 - Prune the tree by
 - Gain - gamma (user defined tree complexity parameter)
 - Positive → do not prune
 - If prune → subtract gamma from the next gain
- **XGboost 相比传统gbdt有何不同**
 - 传统GBDT以CART作为基分类器，XGboost **还支持线性分类器**，这个时候XGboost 相当于带L1和L2正则化项的逻辑回归（分类问题）或者线性回归（回归问题）
 - 传统GBDT在优化时只用到一阶导数信息，xgboost则**对代价函数进行了二阶泰勒展开，同时用到了一阶和二阶导数**。顺便提一下，xgboost工具支持自定义代价函数，只要函数可一阶和二阶求导
 - xgboost在**代价函数里加入了正则项，用于控制模型的复杂度，防止过拟合**。正则项里包含了树的叶子节点个数、每个叶子节点上输出的score的L2模的平方和。从Bias-variance tradeoff角度来讲，正则项降低了模型的variance，使学习出来的模型更加简单，防止过拟合，这也是xgboost优于传统GBDT的一个特性。
 - **Shrinkage (缩减)**，相当于学习速率（xgboost中的eta）。xgboost在进行完一次迭代后，会将叶子节点的权重乘上该系数，主要是为了削弱每棵树的影响，让后面有更大的学习空间。实际应用中，一般把eta设置得小一点，然后迭代次数设置得大一点。
 - **列抽样 (column subsampling)**。xgboost借鉴了随机森林的做法，支持列抽样，不仅能降低过拟合，还能减少计算，这也是xgboost异于传统gbdt的一个特性。
 - **对缺失值的处理**: 对于特征的值有缺失的样本，xgboost可以自动学习出它的分裂方向 <https://towardsdatascience.com/xgboost-is-not-black-magic-56ca013144b4>
- **xgboost为什么快**
 - xgboost工具**支持跨集群并行化**。注意xgboost的并行不是tree粒度的并行，xgboost也是一次迭代完才能进行下一次迭代的（第t次迭代的代价函数里包含了前面t-1次迭代的预测值）
 - xgboost的并行是在特征粒度上的。我们知道，决策树的学习最耗时的一个步骤就是对特征的值进行排序（因为要确定最佳分割点），xgboost在训练之前，预先对数据进行了排序，然后保存为block结构，后面的迭代中重复地使用这个结构，大大减小计算量。
 - 这个block结构也使得并行成为了可能，在进行节点的分裂时，需要计算每个特征的增益，最终选增益最大的那个特征去做分裂，那么各个特征的增益计算就可以开多线程进行。
- 当特征较多、数据集较大、存在离群值和缺失值而又不想进行太多特征工程时使用XGboost

- <https://www.cnblogs.com/pinard/p/6140514.html>
<https://www.zhihu.com/question/41354392>
- 总结
 - Bagging + 决策树 = 随机森林
 - AdaBoost + 决策树 = 提升树
 - Gradient Boosting + 决策树 = GBDT

比较

- Bagging vs. Boosting
 - Adaptive boosting or AdaBoost
 - This method iteratively identifies misclassified data points and adjusts their weights to minimize the training error → continues to optimize in a sequential fashion until it yields the strongest predictor
 - Gradient boosting
 - sequentially adding predictors to an ensemble with each one correcting for the errors of its predecessor
 - Instead of changing the weights of data points like AdaBoost, the gradient boosting trains on the residual errors of the previous predictor. The name, gradient boosting, is used since it combines the gradient descent algorithm and boosting method
 - Extreme gradient boosting or XGBoost
 - an implementation of gradient boosting that's designed for computational speed and scale
 - XGBoost allows for learning to occur in parallel during training
- Gradient boosting (GB) vs. Random Forest (RF)
 - Similarity
 - An ensemble of decision trees are used
 - Flexible and don't need much data preprocessing
 - Difference
 - In GB, trees are built one at a time, such that successive weak learners learn from the mistakes of preceding weak learners. In RF, trees are built independently at the same time.
 - Output: GB combines the results of weak learners with each successive iteration; in RF, trees are combined at the end (average or majority)
 - GB is more prone to overfitting due to their focus on mistakes over training iterations and the lack of independence in tree building.
 - GB hyperparameters are harder to tune
 - GB may take longer to train because the trees of the latter are built sequentially.
 - In real life: GB excels when used on unbalanced datasets (ex. Fraud detection); RF excels at multi-class object detection with noisy data (ex. Computer vision)

- Gradient boosting 的 tree depth会小一点，因为random forest用的都是fully grown trees

面试问题

- boosting和bagging在不同情况下的选用
 - Bagging典型算法有随机森林，主要目的就是降低方差(variance)，但是该算法会导致偏差(bias)比较高
 - Boosting典型算法有AdaBoosting, XGBoosting, LightGBM, GBDT，该类型算法一般可以有较低的偏差，但同样会导致有较高的方差，另外也容易产生过拟合
 - 对于一般比较宽泛的需求预测，但是要求方差波动较小的，可以考虑使用Bagging类型的算法
 - 对于银行对账这类需求，对于模型精度要求高，那么多考虑使用Boosting算法
- Xgboost和随机森林的区别
 - 当降低模型成本时，XGBoost总是更加重视功能空间，而随机森林则试图为超参数提供更多偏好来优化模型

1.18 Dimensionality Reduction 降维

PCA

- 将高维的特征向量合并称为低纬度的特征属性，是一种无监督的降维方法
 - 算法目标是通过某种线性投影，将高维的数据映射到低维的空间中表示，并且期望在所投影的维度上数据的方差最大（最大方差理论），以此使用较少的数据维度，同时保留较多的原数据点的特性
- PCA produces a low-dimensional representation of a dataset. It finds a sequence of linear combinations of the variables that have maximal variance, and are mutually uncorrelated
- 主成分选择
 - 假设原来的特征数据是n维数据，首先选着方差最大方向为第一维数据
 - 第二个坐标轴选择和第一个坐标轴垂直或者正交 的方向；第三个坐标轴选择和第一个、第二个坐标轴都垂直或者正交的方向
 - 该过程一直重复，直到新坐标系的维度和达到给定的值。而这些方向所表示的数据特征就被称为“主成分 principal component”
- 为什么要特征降维
 - 特征属性之间存在着相互关联关系。多重共线性会导致解的空间不稳定，从而导致模型的泛化能力弱
 - 高维空间样本具有稀疏性，导致模型比较难找到数据特征
 - Helps in compressing data and reducing storage space
 - Reduce computation time
 - Removes redundant features
- 维度诅咒？它如何影响距离和相似性度量？
 - 指在高维空间中分析和组织数据时出现问题的情况

- 共同主题：当维数增加时，空间体积增加得很快，导致可用数据变得稀疏
- 问题：支持结果所需的数据量随着维度的增加呈指数增长，需要更多的数据来支持分析
- Apriori Algorithm
 - 购物篮分析中的基础算法 → 目标是找到经常一起购买的产品组合，我们称之为frequent itemsets
 - Step 1. Computing the support for each individual item
 - Step 2. Deciding on the support threshold
 - Step 3. Selecting the frequent items
 - Step 4. Finding the support of the frequent itemsets
 - Step 5. Repeat for larger sets
 - Step 6. Generate Association Rules and compute confidence
 - <https://towardsdatascience.com/the-apriori-algorithm-5da3db9aea95>

LDA

- 线性判断分析(LDA)
 - LDA是一种基于分类模型进行特征属性合并的操作，是一种有监督的降维方法
 - LDA的原理是，将带上标签的数据点，通过投影的方法，投影到维度更低的空间中，使得投影后的点，会形成按类别区分，一簇一簇的情况，相同类别的点，将会在投影后的空间中更接近
 - 用一句话概括就是：“投影后类内方差最小，类间方差最大”
- Fisher's LDA
 - The goal is to find a projection onto a line where the points corresponding to k_1 and k_2 are well separated
 - The best projection makes the difference between the means as large as possible, and with small variance. (class 之间的mean越大越好, within-class variance小)
- Classification
 - LDA is often the bench-marking method before other more complicated and flexible ones are employed
 - generic classification problem
 - A random variable X comes from one of K classes, with some class-specific probability densities $f(x)$.
 - A discriminant rule tries to divide the data space into K disjoint regions that represent all the classes.
 - With these regions, classification by discriminant analysis simply means that we allocate x to class j if x is in region j
 - How do we know which region the data x falls in? two allocation rules:
 - Maximum likelihood rule:
 - assume that each class could occur with equal probability, then allocate x to class j if $j = \operatorname{argmax}_i f_i(x)$

- Bayesian rule: know the class prior probabilities, π , then allocate x to class j if $j = \operatorname{argmax}_i \pi_i f_i(x)$
 - Without the equal covariance assumption, the quadratic term in the likelihood does not cancel out → QDA
 - LDA is used when n is small, classes are well separated, and Gaussian assumptions are reasonable
 - LDA vs. QDA
 - The number of parameters estimated in LDA increases linearly with p while that of QDA increases quadratically with p
 - when the problem dimension is large → expect QDA to have worse performance than LDA
 - Compromise: LDA with regularization
 - When $n_{\text{feature}} > n_{\text{sample}}$, LDA with shrinkage performs better
- PCA和LDA比较
 - 相同点:
 - 两者均可以对数据完成降维操作
 - 两者在降维时候均使用矩阵分解的思想
 - 两者都假设数据符合高斯分布
 - 不同点:
 - LDA是监督降维算法, PCA是无监督降维算法
 - LDA降维最多降到类别数目 $k-1$ 的维数, 而PCA没有限制
 - LDA除了降维外, 还可以应用于分类
 - LDA选择的是分类性能最好的投影, 而PCA选择样本点投影具有最大方差的方向
 - LDA 有labels, PCA 只是combine features

面试问题

- 特征值与特征向量物理意义
 - 特征值, 表示一个矩阵的向量被拉伸或压缩的程度

经验风险最小化 (ERM) 与结构风险最小化 (SRM)

极大似然估计 (MLE) 与最大后验概率估计 (MAP)

2. 线性神经网络

3. 卷积神经网络

2.1 卷积相关定义

- 卷积神经网络 - CNN 最擅长的就是图片的处理。它受到人类视觉神经系统的启发
- CNN 有2大特点：
 - 能够有效的将大数据量的图片降维成小数据量
 - 能够有效的保留图片特征，符合图片处理的原则
- CNN 解决了什么问题
 - **“将复杂问题简化”，把大量参数降维成少量参数，再做处理**
 - **用类似视觉的方式保留了图像的特征，当图像做翻转，旋转或者变换位置时，它也能有效的识别出来是类似的图像。**
- 人类的视觉原理
 - 从原始信号摄入开始（瞳孔摄入像素 Pixels）接着做初步处理（大脑皮层某些细胞发现边缘和方向）
然后抽象（大脑判定，眼前的物体的形状，是圆形的）然后进一步抽象（大脑进一步判定该物体是只气球）
 - CNN模仿人类大脑的这个特点，构造多层的神经网络，较低层的识别初级的图像特征
若干底层特征组成更上一层特征，最终通过多个层级的组合，最终在顶层做出分类
- CNN 由3个部分构成：卷积层, 池化层, 全连接层
- 卷积层
 - 卷积层负责提取图像中的局部特征
 - 卷积具有“权值共享”这样的特性，可以降低参数数量，达到降低计算开销，防止由于参数过多而造成过拟合。
 - 卷积层是构建卷积神经网络的**核心层**，它产生了网络中大部分的**计算量**
 - 输入 $X = n_h * n_w$ ，卷积核 kernel $W = k_h * k_w$ ，偏差参数 $b \in \mathbb{R}$
输出 $Y = X * W + b$ ，输出为 $(n_h - k_h + 1) * (n_w - k_w + 1)$
 - 填充 padding：
 - 给定 $32 * 32$ 图像和 $5 * 5$ 卷积核，第一层会得到 $28 * 28$ 输出 $(n-k+1)$
 - padding 在周围添加额外的行或列
 - 输出为 $(n_h - k_h + p_h + 1) * (n_w - k_w + p_w + 1)$
 - 步幅
 - 给定 $224 * 224$ 图像和 $5 * 5$ 卷积核，需要55层才能降到 $4 * 4$ ，需要大量计算才能得到较小输出
 - 步幅可以使得卷积核每次滑动多个元素
 - 输出为 $[(n_h - k_h + p_h + 1) / s_h] * [(n_w - k_w + p_w + 1) / s_w]$

- ```
构造一个二维卷积层，它具有3个输入通道，3个输出通道和形状为（2，2）的卷积核
每次横移2格，竖移两格，上下左右各加一行padding
conv2d = nn.Conv2d(in_channels=3, out_channels=3, kernel_size=(2, 2),
 stride=(2, 2), padding=(1,1), bias=False)

多输出通道
conv2d = nn.Conv2d(in_channels=3, out_channels=3, kernel_size=(2, 2))
```

- 池化层 (Pooling)

- 池化层可以说是下采样，用来大幅降低参数量级(降维)，防止过拟合
- 池化的过程中基本不会进行另补充；其次池化前后深度不变, 输出通道等于输出通道
- 最大池化的效果比平均池化
- 最大池化层 (Max Pooling): 返回滑动窗口中的最大值

- ```
# 二维最大池化层,window size为3，步幅为2
maxpool = nn.MaxPool2d((3, 3), stride=(2, 2))

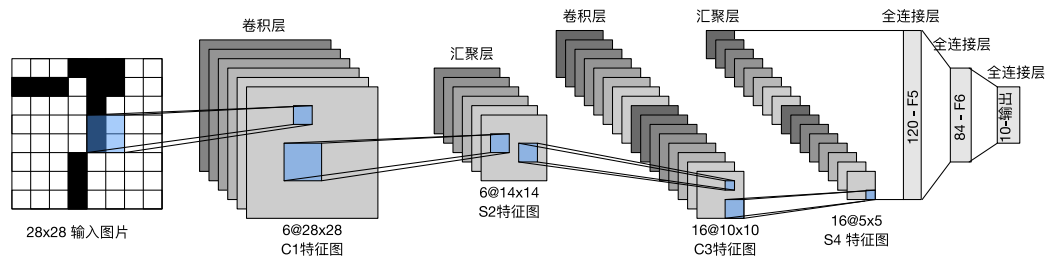
# 非正方形窗口
maxpool = nn.MaxPool2d((3, 2), stride=(2, 1))
```

- 全连接层

- 全连接层类似神经网络的部分，用来输出想要的结果。
- ```
fc2 = nn.Linear(in_features=500, out_features=128)
```

## 2.2 LeNet卷积神经网络

- 第一个成功的卷积神经网络应用，是Yann LeCun在上世纪90年代实现的。
- LeNet (LeNet-5) 由两个部分组成
  - 卷积编码器：由两个卷积层组成
  - 全连接层密集块：由三个全连接层组成



- ```
class LeNet(nn.Module):
    def __init__(self, inputChannels, num_classes):
        super(LeNet, self).__init__()

        # inputChannels*32*32 -> 6*28*28
```

```

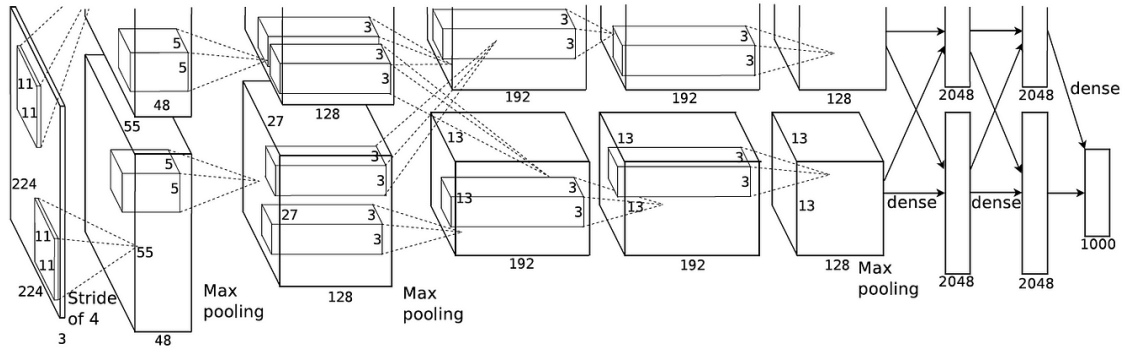
        self.conv1 = nn.Conv2d(in_channels = inputChannels, out_channels =
6, kernel_size = (5, 5))
        self.relu1 = nn.ReLU()
        # 6*28*28 -> 6*14*14
        self.maxpool1 = nn.MaxPool2d(kernel_size = 2, stride = 2)
        # 6*14*14 -> 16*10*10
        self.conv2 = nn.Conv2d(in_channels = 6, out_channels = 16,
kernel_size = (5, 5))
        self.relu2 = nn.ReLU()
        # 16*10*10 -> 16*5*5
        self.maxpool2 = nn.MaxPool2d(kernel_size = 2, stride = 2)
        # 16*5*5 -> 120 -> 84 -> num_classes
        self.fc1 = nn.Linear(in_features=16*5*5, out_features=120)
        self.sgm1 = nn.Sigmoid()
        self.fc2 = nn.Linear(in_features=120, out_features=84)
        self.sgm2 = nn.Sigmoid()
        self.fc3 = nn.Linear(in_features=84, out_features=num_classes)
        self.softmax = nn.Softmax(dim=1)
        self.logSoftmax = nn.LogSoftmax(dim=1)

    def forward(self, x):
        x = self.relu1(self.conv1(x))
        x = self.maxpool1(x)
        x = self.relu2(self.conv2(x))
        x = self.maxpool2(x)
        x = torch.flatten(x, 1)
        x = self.sgm1(self.fc1(x))
        x = self.sgm2(self.fc2(x))
        x = self.fc3(x)
        return x

```

2.3 AlexNet 深度卷积神经网络

- AlexNet和LeNet的设计理念非常相似，但也存在显著差异
- AlexNet比相对较小的LeNet5要深得多。AlexNet由八层组成：五个卷积层、两个全连接隐藏层和一个全连接输出层
- AlexNet的卷积通道数目是LeNet的10倍
- AlexNet将sigmoid激活函数改为更简单的ReLU激活函数。
 - 一方面，ReLU激活函数的计算更简单，它不需要如sigmoid激活函数那般复杂的求幂运算。
 - 另一方面，当使用不同的参数初始化方法时，ReLU激活函数使训练模型更加容易。



•

- ```

class AlexNet(nn.Module):
 def __init__(self, num_classes, dropout=0.5):
 super(AlexNet, self).__init__()

 self.features = nn.Sequential(
 # use 11*11 bigger window to capture object, and use less stride to
 # reduce size
 # [3, 224, 224] -> [64, 55, 55] -> [64, 27, 27]
 nn.Conv2D(3, 64, kernel_size = 11, stride = 4, padding = 2),
 nn.ReLU(inplace=True),
 nn.MaxPool2d(kernel_size = 3, stride = 2),
 # [64, 27, 27] -> [192, 27, 27] -> [192, 13, 13]
 nn.Conv2D(64, 192, kernel_size = 5, padding = 2),
 nn.ReLU(inplace=True),
 nn.MaxPool2d(kernel_size=3, stride=2),
 # three conv2d
 # [192, 13, 13] -> [384, 13, 13] -> [256, 13, 13] -> [256, 13, 13]
 nn.Conv2d(192, 384, kernel_size=3, padding=1),
 nn.ReLU(inplace=True),
 nn.Conv2d(384, 256, kernel_size=3, padding=1),
 nn.ReLU(inplace=True),
 nn.Conv2d(256, 256, kernel_size=3, padding=1),
 nn.ReLU(inplace=True),
 # [256, 13, 13] -> [256, 6, 6]
 nn.MaxPool2d(kernel_size=3, stride=2),
)
 self.avgpool = nn.AdaptiveAvgPool2d((6, 6))
 self.classifier = nn.Sequential(
 # use dropout to mitigate overfitting
 # 256*6*6 -> 4096 -> 4096 -> num_classes
 nn.Dropout(p=dropout),
 nn.Linear(256*6*6, 4096),
 nn.ReLU(inplace=True),
 nn.Dropout(p=dropout),
 nn.Linear(4096, 4096),
 nn.ReLU(inplace=True),
 nn.Linear(4096, num_classes),
)

 def forward(self, x):

```

```

x = self.features(x)
x = self.avgpool(x)
x = torch.flatten(x, 1)
x = self.classifier(x)
return x

```

## 2.4 VGG 使用块的网络

- VGG块与之类似，由一系列卷积层组成，将卷积层组合成块，后面再加上用于空间下采样的最大汇聚层
- VGG网络可以分为两部分：第一部分主要由卷积层和汇聚层组成，第二部分由全连接层组成

| ConvNet Configuration       |                        |                        |                                     |                                     |                                                  |
|-----------------------------|------------------------|------------------------|-------------------------------------|-------------------------------------|--------------------------------------------------|
| A                           | A-LRN                  | B                      | C                                   | D                                   | E                                                |
| 11 weight layers            | 11 weight layers       | 13 weight layers       | 16 weight layers                    | 16 weight layers                    | 19 weight layers                                 |
| input (224 × 224 RGB image) |                        |                        |                                     |                                     |                                                  |
| conv3-64                    | conv3-64<br>LRN        | conv3-64<br>conv3-64   | conv3-64<br>conv3-64                | conv3-64<br>conv3-64                | conv3-64<br>conv3-64                             |
| maxpool                     |                        |                        |                                     |                                     |                                                  |
| conv3-128                   | conv3-128              | conv3-128<br>conv3-128 | conv3-128<br>conv3-128              | conv3-128<br>conv3-128              | conv3-128<br>conv3-128                           |
| maxpool                     |                        |                        |                                     |                                     |                                                  |
| conv3-256<br>conv3-256      | conv3-256<br>conv3-256 | conv3-256<br>conv3-256 | conv3-256<br>conv3-256<br>conv1-256 | conv3-256<br>conv3-256<br>conv3-256 | conv3-256<br>conv3-256<br>conv3-256<br>conv3-256 |
| maxpool                     |                        |                        |                                     |                                     |                                                  |
| conv3-512<br>conv3-512      | conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512<br>conv1-512 | conv3-512<br>conv3-512<br>conv3-512 | conv3-512<br>conv3-512<br>conv3-512<br>conv3-512 |
| maxpool                     |                        |                        |                                     |                                     |                                                  |
| conv3-512<br>conv3-512      | conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512<br>conv1-512 | conv3-512<br>conv3-512<br>conv3-512 | conv3-512<br>conv3-512<br>conv3-512<br>conv3-512 |
| maxpool                     |                        |                        |                                     |                                     |                                                  |
| FC-4096                     |                        |                        |                                     |                                     |                                                  |
| FC-4096                     |                        |                        |                                     |                                     |                                                  |
| FC-1000                     |                        |                        |                                     |                                     |                                                  |
| soft-max                    |                        |                        |                                     |                                     |                                                  |

- ```

class VGG(nn.Module):
    def __init__(self, num_classes = 10, in_channels = 3, arch_type = 'vgg_11'):
        super(VGG, self).__init__()

        # 原始VGG网络有5个卷积块，其中前两个块各有一个卷积层，后三个块各包含两个卷积层
        # 由于该网络使用8个卷积层和3个全连接层，因此它通常被称为VGG-11
        self.conv_arch = {
            'vgg_11': ((1, 64), (1, 128), (2, 256), (2, 512), (2, 512)),
            'vgg_13': ((2, 64), (2, 128), (2, 256), (2, 512), (2, 512)),
            'vgg_19': ((2, 64), (1, 128), (4, 256), (4, 512), (4, 512))}

        self.conv_blocks = []
        for (num_conv, out_channels) in self.conv_arch[arch_type]:
            self.conv_blocks.append(self.vgg_block(num_conv, in_channels,
            out_channels))
            in_channels = out_channels

        self.classifier = nn.Sequential(
            nn.Linear(out_channels*7*7, 4096),
            nn.ReLU(inplace=True),
            nn.Dropout(0.5),

```



```

        nn.Linear(4096, 4096),
        nn.ReLU(inplace=True),
        nn.Dropout(0.5),
        nn.Linear(4096, num_classes))

    self.features = nn.Sequential(*self.conv_blocks)

    def vgg_block(self, num_conv, in_channels, out_channels):
        layers = []
        for _ in range(num_conv):
            layers.append(nn.Conv2d(in_channels, out_channels, kernel_size = 3,
padding = 1))
            layers.append(nn.ReLU())
            in_channels = out_channels
        layers.append(nn.MaxPool2d(kernel_size=2, stride=2))
        return nn.Sequential(*layers)

    def forward(self, x):
        x = self.features(x)
        x = torch.flatten(x, 1)
        x = self.classifier(x)
        return x

```

2.5 Batch Normalization 批量规范化

- 训练深层网络可能具有更广的变化范围
- 不论是沿着从输入到输出的层，跨同一层中的单元，或是随着时间的推移，模型参数的随着训练更新变幻莫测
- 更深层的网络很复杂，训练慢收敛很慢，导致多次重新学习，容易过拟合。这意味着正则化变得更加重要
- 批量规范化应用于单个可选层, 在每次训练迭代中,

- 我们首先规范化输入，固定小批量里面的均值和方差

$$\mu_B = \frac{1}{|B|} \sum_{i \in B} x_i \text{ and } \sigma = \frac{1}{|B|} \sum_{i \in B} (x_i - \mu)^2 + \epsilon$$

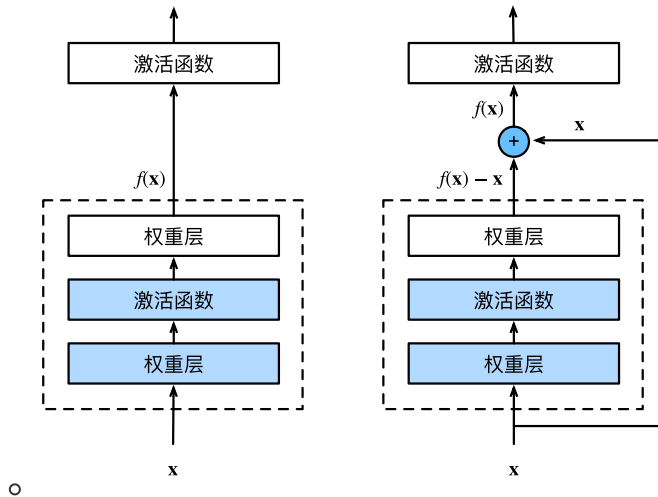
- 即通过减去其均值并除以其标准差，其中两者均基于当前小批量处理 (γ 和 β 是可学习参数)

$$x_{i+1} = \gamma \frac{x_i - \mu_B}{\sigma} + \beta$$

- 批量规范化层可以用在
 - 全连接和卷积层输出上，激活函数之前
 - 全连接和卷积层输入上
- 对于全连接，作用在特征维; 对于卷积层，作用在通道维
- 没必要跟丢弃法(dropout)一起使用
- 加快收敛，但是不改变模型精度

2.6 ResNet 残差网络

- 网络越深效果越不好，但如果加了一个同等映射，效果起码不会比之前差
- 核心思想：每个附加层都应该更容易地包含原始函数作为其元素之一
- 残差块：串联一个层改变函数类，我们希望能扩大函数类，残差块加入快速通道来获得之前的 x
 - $f(x) = x + g(x)$

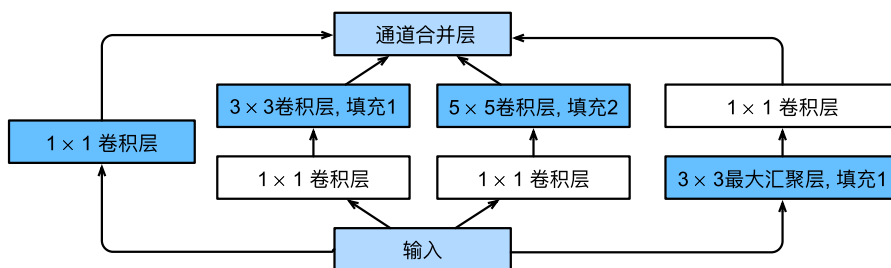


- 残差块细节
 - ResNet沿用了VGG完整的 3×3 卷积层设计
 - 残差块里首先有2个有相同输出通道数的 3×3 卷积层。每个卷积层后接一个批量规范化层和ReLU激活函数
 - 通过跨层数据通路，跳过这2个卷积运算，将输入直接加在最后的ReLU激活函数前
- 残差块使得很深的网络更加容易训练，甚至可以一千层

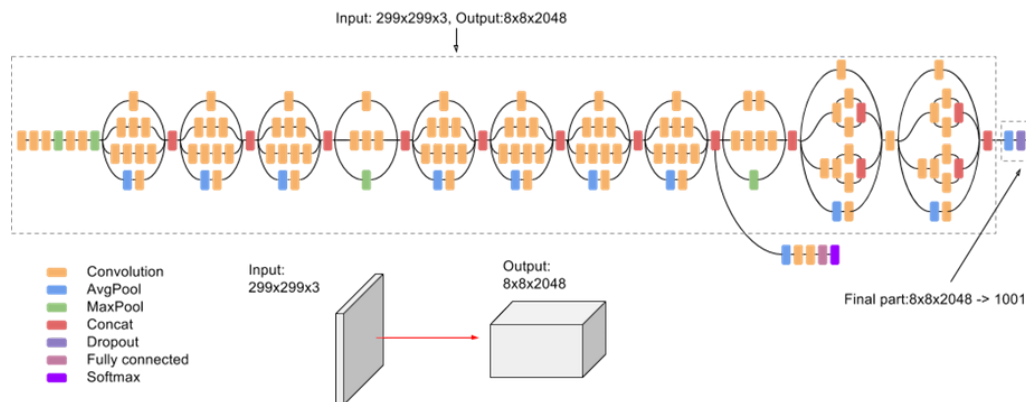
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	$7 \times 7, 64, \text{stride } 2$				
conv2_x	56×56	$3 \times 3 \text{ max pool, stride } 2$				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

2.7 GoogLeNet

- Inception块：基本的卷积块被称为Inception块（Inception block）
- Inception块由四条并行路径组成，从不同空间大小中提取信息，再输出通道上合并



- GoogLeNet一共使用9个Inception块和全局平均汇聚层的堆叠来生成其估计值。



o

3. 优化算法

4. 计算机视觉

4.1 图像增广

- 图像增广意义
 - 对训练图像进行一系列的随机变化之后，生成相似但不同的训练样本，从而扩大了训练集的规模
 - 随机改变训练样本可以减少模型对某些属性的依赖，从而提高模型的泛化能力
- 常用的图像增广方法
 - 翻转和裁剪

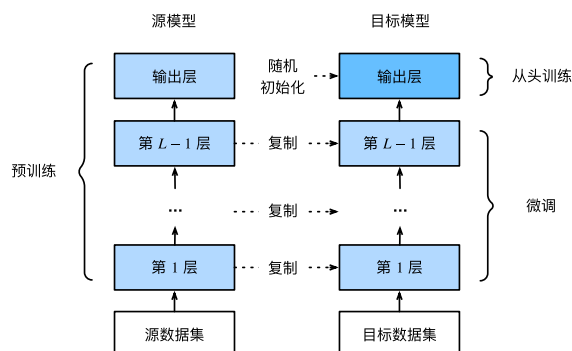
```
torchvision.transforms.RandomHorizontalFlip()
torchvision.transforms.RandomVerticalFlip()
torchvision.transforms.RandomResizedCrop((200, 200), scale=(0.1, 1), ratio=(0.5, 2))
```

- 改变颜色: 亮度、对比度、饱和度和色调

```
torchvision.transforms.ColorJitter(brightness=0.5, contrast=0.5,
saturation=0.5, hue=0.5)
```

4.2 Fine-tuning 微调

- 神经网络分为两类
 - 特征提取将原始像素变成容易线性分割的特征
 - 线性分类器来分类
- 步骤
 - 在源数据集上预训练神经网络模型，即**源模型**
 - 创建一个新的神经网络模型，即**目标模型**。这将复制源模型上的所有模型设计及其参数（输出层除外）
 - 向目标模型添加输出层，其输出数是目标数据集中的类别数。然后随机初始化该层的模型参数
 - 在目标数据集（如椅子数据集）上训练目标模型。输出层将从头开始进行训练，而所有其他层的参数将根据源模型的参数进行微调



```
model = torchvision.models.resnet18(pretrained=True)
model.fc = nn.Linear(finetune_net.fc.in_features, 2) # 改变最后一层输出
nn.init.xavier_uniform_(model.fc.weight) # 随机初始化weight
```

- 可以专门训练最后一层weight

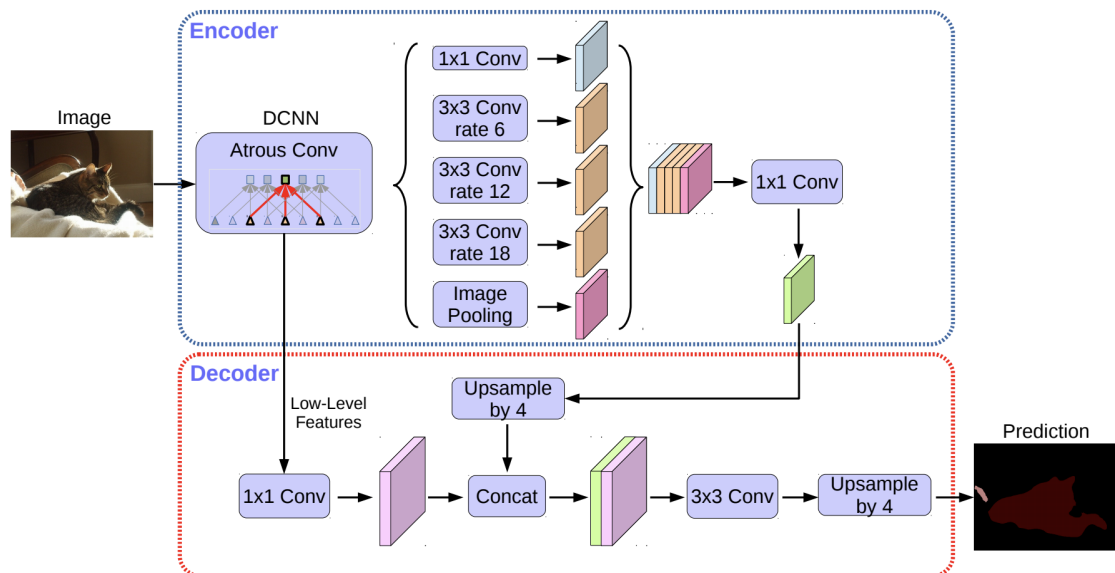
```
# 如果param_group=True, 输出层中的模型参数将使用十倍的学习率
if param_group:
    params_1x = [param for name, param in net.named_parameters()
                  if name not in ["fc.weight", "fc.bias"]]
    trainer = torch.optim.SGD([{'params': params_1x},
                               {'params': net.fc.parameters(),
                                'lr': learning_rate * 10}],
                              lr=learning_rate, weight_decay=0.001)
else:
    trainer = torch.optim.SGD(net.parameters(), lr=learning_rate,
                              weight_decay=0.001)
```

- 总结
 - 微调通过使用在大数据上得到的预训练好的模型来初始化模型权重来完成提升精度
 - 预训练模型质量很重要

- 微调通常速度更快、精度更高

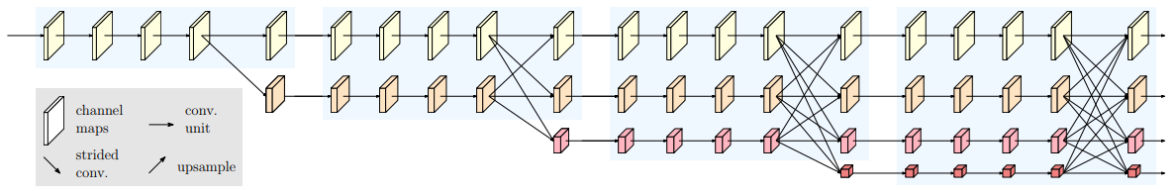
4.3 语义分割

- 语义分割(semantic segmentation)，图像分割 (image segmentation) 和实例分割 (instance segmentation)
 - 语义分割：重点关注于如何将图像分割成属于不同语义类别的区域。
 - 图像分割：将图像划分为若干组成区域，这类问题的方法通常利用图像中像素之间的相关性。
 - 实例分割：同时检测并分割 (simultaneous detection and segmentation)，研究如何识别图像中各个目标实例的像素级区域。与语义分割不同，实例分割不仅需要区分语义，还要区分不同的目标实例。
- DeepLab 模型
 - 空洞卷积 (dilated convolution)：通过引入扩张率 (Dilation Rate) 这一参数使得同样尺寸的卷积核获得更大的感受野
 - 图像分割任务需要较大的感受野来更好的完成任务
 - 通过设置dilated rate来完成空洞卷积，扩大任意倍数，没有额外计算，没有额外参数
 - SPP-Layer：使用不同pooling layer, $\{16, 4, 1\} \times \text{channel}$ ，最后进行特征拼接，
 - ASPP-Layer(atrous convolution SPP): 引入不同倍率的空洞卷积



4.4 HRNet 高分辨网络

- ResNet以及GoogleNet都是提取低分辨率特征，对于位置敏感的任务例如检测和分割来说，高分辨率的空间特征是非常重要的
- HRNet始于一组高分辨率卷积，然后逐步添加低分辨率的卷积分支，并将它们以并行的方式连接起来



- 网络由若干阶段组成，其中第 n 段包含 n 个卷积分支并且具备 n 个不同的分辨率
- **从头到尾保持高分辨率，而不同分支的信息交互是为了补充通道数减少带来的信息损耗**
- 获得的高分辨表征不仅仅习得了健壮的语义特征，在空间维度上的准确性也表现良好
 - 不同分辨率分支并行连接而非串行，因此整个网络都保留了高分辨表征
 - HRNet始终保留了高分辨率与低分辨率特征，并且不断的让两者相融合，相互促进。
- HRNet有两个版本
 - HRNetV1仅仅输出高分辨率卷积分支的结果，用在人类姿态估算任务中
 - HRNetV2则结合了所有并行的分支，用在图像分割中。

5. Transformer 模型

5.1 Transformer

- Transformer 模型中也采用了 encoder-decoder 架构
- Transformer是一个完全基于注意力机制的编解码器模型，它抛弃了之前其它模型引入注意力机制后仍然保留的循环与卷积结构，而采用了自注意力（Self-attention）机制，在任务表现、并行能力和易于训练性方面都有大幅的提高
- 定义输入序列首先经过 word embedding，再和 positional encoding 相加后，输入到 encoder 中。输出序列经过的处理和输入序列一样，然后输入到 decoder，最后，decoder 的输出经过一个线性层，再接 Softmax

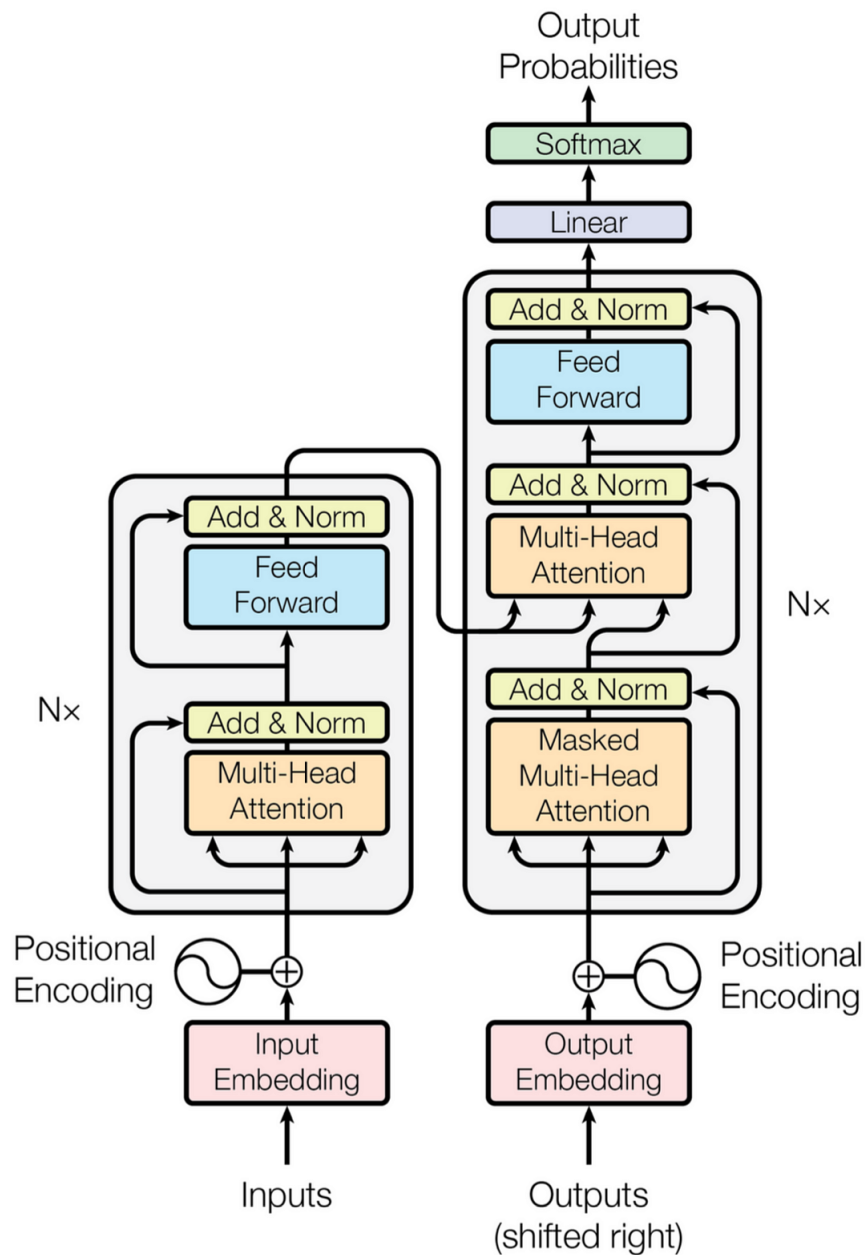
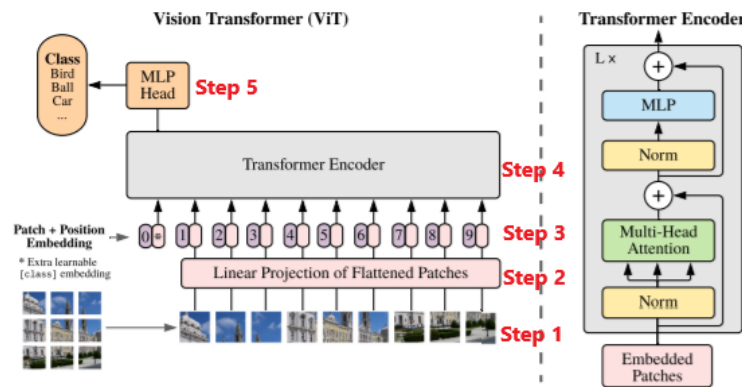


Figure 1: The Transformer - model architecture.

5.2 Vision Transformer

- Transformer最初提出是针对NLP领域的，并且在NLP领域大获成功。ViT也是受到其启发，尝试将Transformer应用到CV领域。
- 模型框架
 - Linear Projection of Flattened Patches(Embedding层)
 - 对于标准的Transformer模块，要求输入的是token（向量）序列，即二维矩阵[num_token, token_dim]，对于图像数据而言，其数据格式为[H, W, C]是三维矩阵明显不是Transformer想要的。所以需要先通过一个Embedding层来对数据做个变换。如下图所示，首先将一张图片按给定大小分成一堆Patches。
 - 在代码实现中，直接通过一个卷积层来实现。以ViT-B/16为例，直接使用一个卷积核大小为16x16，步距为16，卷积核个数为768的卷积来实现。

- 在输入Transformer Encoder之前注意需要加上[class]token以及Position Embedding。



- Transformer Encoder
 - Transformer Encoder其实就是重复堆叠Encoder Block L 次
- MLP Head (最终用于分类的层结构)
 - 上面通过Transformer Encoder后输出的shape和输入的shape是保持不变的, 以ViT-B/16为例, 输入的是[197, 768]输出的还是[197, 768]
 - 这里我们只是需要分类的信息, 所以我们只需要提取出[class]token生成的对应结果就行, 即[197, 768]中抽取出[class]token对应的[1, 768]。接着我们通过MLP Head得到我们最终分类结果。

6. Variational autoencoder(VAE)

- 混合模型 (Mixture Model)
 - 混合模型是一个可以用来表示在总体分布 (distribution) 中含有 K 个子分布的概率模型
 - 混合模型表示了观测数据在总体中的概率分布, 它是一个由 K 个子分布组成的混合分布
 - 混合模型不要求观测数据提供关于子分布的信息, 来计算观测数据在总体分布中的概率
- 单高斯模型
 - 当样本数据 X 是一维数据 (Univariate) 时, 高斯分布遵从下方概率密度函数 (Probability Density Function) :

$$P(x|\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)$$

- 当样本数据 X 是多维数据 (Multivariate) 时, 高斯分布遵从下方概率密度函数, D 为数据维度:

$$P(x|\theta) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{(x - \mu)^T \Sigma^{-1} (x - \mu)}{2}\right)$$

- 高斯混合模型 (Gaussian Mixture Model)
 - 高斯混合模型可以看作是由 K 个单高斯模型组合而成的模型, 这 K 个子模型是混合模型的隐变量 (Hidden variable)
 - 一个混合模型可以使用任何概率分布, 这里使用高斯混合模型是因为高斯分布具备很好的数学性质以及良好的计算性能

- 例如有一组狗的样本数据，不同种类的狗，体型、颜色、长相各不相同，但都属于狗这个种类，此时单高斯模型可能不能很好的来描述这个分布，因为样本数据分布并不是一个单一的椭圆，所以用混合高斯分布可以更好的描述这个问题