Machine Learning Data Processing

Data Types

- Nominal - is for mutual exclusive, but not ordered, categories.
- Ordinal - is one where the order matters but not the difference between values.
- Ratio - has all the properties of an interval variable, and also has a clear definition of 0.0.
- Interval - is a measurement where the difference between two values is meaningful.

	Nominal	Ordinal	Interval	Ratio
Counts / Distribution	✔	✔	✔	✔
Minimum, Maximum		✔	✔	✔
Range		✔	✔	✔
Percentiles		✔	✔	✔
Standard deviation, Variance			✔	✔

	Nominal	Ordinal	Interval	Ratio
Mode	✔	✔	✔	✔
Median		✔	✔	✔
Mean			✔	✔

	Nominal	Ordinal	Interval	Ratio
Countable	✔	✔	✔	✔
Order defined		✔	✔	✔
Difference defined (addition, subtraction)			✔	✔
Zero defined (multiplication, division)				✔

Data Exploration

- Variable Identification
  - Identify Predictor (Input) and Target (output) variables. Next, identify the data type and category of the variables.
- Univariate Analysis
  - Continuous Features
    - Mean, Median, Mode, Min, Max, Range, Quartile, IQR, Variance, Standard Deviation, Skewness, Histogram, Box Plot
  - Categorical Features
    - Frequency, Histogram
- Bi-variate Analysis
  - Finds out the relationship between two variables.
  - Scatter Plot
  - Correlation Plot - Heatmap
  - Two-way table
    - We can start analyzing the relationship by creating a two-way table of count and count%.
  - Stacked Column Chart
  - Chi-Square Test
    - This test is used to derive the statistical significance of relationship between the variables.
  - Z-Test/ T-Test
  - ANOVA

Feature Cleaning

- Missing values
  - One may choose to either omit elements from a dataset that contain missing values or to impute a value
- Special values
  - Numeric variables are endowed with several formalized special values including ±Inf, NA and NaN. Calculations involving special values often result in special values, and need to be handled/cleaned
- Outliers
  - They should be detected, but not necessarily removed. Their inclusion in the analysis is a statistical decision.
- Obvious inconsistencies
  - A person's age cannot be negative, a man cannot be pregnant and an under-aged person cannot possess a drivers license.

Feature Imputation

- Hot-Deck
  - The technique then finds the first missing value and uses the cell value immediately prior to the data that are missing to impute the missing value.
- Cold-Deck
  - Selects donors from another dataset to complete missing data.
- Mean-substitution
  - Another imputation technique involves replacing any missing value with the mean of that variable for all other cases, which has the benefit of not changing the sample mean for that variable.
- Regression
  - A regression model is estimated to predict observed values of a variable based on other variables, and that model is then used to impute values in cases where that variable is missing
- Some Libraries...

Package	Numeric		Kind data			Longitudinal	
	mean	med.	dep.	seq.	gener.	DBM	TSK.
Amelia II[1]				✓	✓		
bnpsd			✓				
CrossClassify R package							
dalex	✓						
fancyimpute	✓		✓		✓	✓	
gain	✓				✓		
hdmpy	✓				✓		
Hmisc[2][3]	✓	✓					
imputeTS[4]					✓		
mice[5]	✓	✓	✓		✓		
missForest							
mlr	✓	✓	✓		✓		
missingness							
missingness.shiny[6]	✓	✓	✓		✓		
missMDA[7]							
missRanger[8]			✓		✓		
pcaMethods[9]					✓		
softImpute[10]					✓		
vim[11]	✓					✓	✓

Feature Engineering

- Decompose
  - Converting 2014-09-20T20:45:40Z into categorical attributes like hour_of_the_day, part_of_day, etc.
- Discretization
  - Continuous Features
    - Typically data is discretized into partitions of K equal lengths/width (equal intervals) or K% of the total data (equal frequencies).
  - Categorical Features
    - Values for categorical features may be combined, particularly when there's few samples for some categories.
- Reframe Numerical Quantities
  - Changing from grams to kg, and losing detail might be both wanted and efficient for calculation
- Crossing
  - Creating new features as a combination of existing features. Could be multiplying numerical features, or combining categorical variables. This is a great way to add domain expertise knowledge to the dataset.

Feature Selection

- Correlation
  - Features should be uncorrelated with each other and highly correlated to the feature we're trying to predict.
  - $$\mathrm{corr}(X,Y) = \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y}$$
  - Covariance
    - $$\mathrm{cov}(x,y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$
    - A measure of how much two random variables change together. Math: dot(de_mean(x), de_mean(y)) / (n - 1)
- Dimensionality Reduction
  - Principal Component Analysis (PCA)
    - Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components.
    - Plot the variance per feature and select the features with the largest variance.
  - Singular Value Decomposition (SVD)
    - SVD is a factorization of a real or complex matrix. It is the generalization of the eigendecomposition of a positive semidefinite normal matrix (for example, a symmetric matrix with positive eigenvalues) to any $m \times n$ matrix via an extension of the polar decomposition. It has many useful applications in signal processing and statistics.
    - $$\mathbf{M} = \mathbf{U\Sigma V}^*$$
      where
      - $\mathbf{U}$ is an $m \times m$ unitary matrix (if $K = \mathbb{R}$, unitary matrices are orthogonal matrices),
      - Σ is a diagonal $m \times n$ matrix with non-negative real numbers on the diagonal,
      - $\mathbf{V}$ is an $n \times n$ unitary matrix over K, and
      - $\mathbf{V}^*$ is the conjugate transpose of $\mathbf{V}$.
- Importance
  - Filter Methods
    - Filter type methods select features based only on general metrics like the correlation with the variable to predict. Filter methods suppress the least interesting variables. The other variables will be part of a classification or a regression model used to classify or to predict data. These methods are particularly effective in computation time and robust to overfitting.
    - Correlation
    - Linear Discriminant Analysis
    - ANOVA: Analysis of Variance
    - Chi-Square
  - Wrapper Methods
    - Wrapper methods evaluate subsets of variables which allows, unlike filter approaches, to detect the possible interactions between variables. The two main disadvantages of these methods are : The increasing overfitting risk when the number of observations is insufficient. AND. The significant computation time when the number of variables is large.
    - Forward Selection
    - Backward Elimination
    - Recursive Feature Ellimination
    - Genetic Algorithms
  - Embedded Methods
    - Embedded methods try to combine the advantages of both previous methods. A learning algorithm takes advantage of its own variable selection process and performs feature selection and classification simultaneously.
    - Lasso regression performs L1 regularization which adds penalty equivalent to absolute value of the magnitude of coefficients.
    - Ridge regression performs L2 regularization which adds penalty equivalent to square of the magnitude of coefficients.

Feature Encoding

- Machine Learning algorithms perform Linear Algebra on Matrices, which means all features must be numeric. Encoding helps us do this.
- Label Encoding

Sample	Category	Numerical
1	Human	5
2	Human	5
3	Penguin	2
4	Octopus	3
5	Alien	4
6	Octopus	3
7	Alien	4

- One Hot Encoding

Human	Penguin	Octopus	Alien
1	0	0	0
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
0	0	1	0
0	0	0	1

- In One Hot Encoding, make sure the encodings are done in a way that all features are linearly independent.

Feature Normalisation or Scaling

- Since the range of values of raw data varies widely, in some machine learning algorithms, objective functions will not work properly without normalization. Another reason why feature scaling is applied is that gradient descent converges much faster with feature scaling than without it.
- Methods
  - Rescaling
    - The simplest method is rescaling the range of features to scale the range in [0, 1] or [−1, 1].
    - $$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$
  - Standardization
    - Feature standardization makes the values of each feature in the data have zero-mean (when subtracting the mean in the numerator) and unit-variance.
    - $$x' = \frac{x - \bar{x}}{\sigma}$$
  - Scaling to unit length
    - To scale the components of a feature vector such that the complete vector has length one.
    - $$x' = \frac{x}{||x||}$$

Dataset Construction

- Training Dataset
  - A set of examples used for learning
  - To fit the parameters of the classifier in the Multilayer Perceptron, for instance, we would use the training set to find the "optimal" weights when using back-progapation.
- Test Dataset
  - A set of examples used only to assess the performance of a fully-trained classifier
  - In the Multilayer Perceptron case, we would use the test to estimate the error rate after we have chosen the final model (MLP size and actual weights) After assessing the final model on the test set, YOU MUST NOT tune the model any further.
- Validation Dataset
  - A set of examples used to tune the parameters of a classifier
  - In the Multilayer Perceptron case, we would use the validation set to find the "optimal" number of hidden units or determine a stopping point for the back-propagation algorithm
- Cross Validation
  - One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (called the validation set or testing set). To reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds.

Machine Learning Concepts

Types

- **Regression** — A supervised problem, the outputs are continuous rather than discrete.
- **Classification** — Inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more (multi-label classification) of these classes. This is typically tackled in a supervised way.
- **Clustering** — A set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.
- **Density Estimation** — Finds the distribution of inputs in some space.
- **Dimensionality Reduction** — Simplifies inputs by mapping them into a lower-dimensional space.

Kind

- **Parametric**
  - Step 1: Making an assumption about the functional form or shape of our function (f), i.e.: f is linear, thus we will select a linear model.
  - Step 2: Selecting a procedure to fit or train our model. This means estimating the Beta parameters in the linear function. A common approach is the (ordinary) least squares, amongst others.
- **Non-Parametric** — When we do not make assumptions about the form of our function (f). However, since these methods do not reduce the problem of estimating f to a small number of parameters, a large number of observations is required in order to obtain an accurate estimate for f. An example would be the thin-plate spline model.

Categories

- **Supervised** — The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.
- **Unsupervised** — No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).
- **Reinforcement Learning** — A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). The program is provided feedback in terms of rewards and punishments as it navigates its problem space.

Approaches

- Decision tree learning
- Association rule learning
- Artificial neural networks
- Deep learning
- Inductive logic programming
- Support vector machines
- Clustering
- Bayesian networks
- Reinforcement learning
- Representation learning
- Similarity and metric learning
- Sparse dictionary learning
- Genetic algorithms
- Rule-based machine learning
- Learning classifier systems

Taxonomy

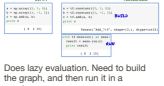
- **Generative Methods**
  - **Popular models**
    - Mixtures of Gaussians, Mixtures of experts, Hidden Markov Models (HMM)
    - Gaussians, Naïve Bayes, Mixtures of multinomials
    - Sigmoidal belief networks, Bayesian networks, Markov random fields
  - Model class-conditional pdfs and prior probabilities. "Generative" since sampling can generate synthetic data points.
- **Discriminative Methods**
  - Directly estimate posterior probabilities. No attempt to model underlying probability distributions. Focus computational resources on given task– better performance
  - **Popular Models**
    - Logistic regression, SVMs
    - Traditional neural networks, Nearest neighbor
    - Conditional Random Fields (CRF)

Selection Criteria

- **Prediction Accuracy vs Model Interpretability** — There is an inherent tradeoff between Prediction Accuracy and Model Interpretability, that is to say that as the model get more flexible in the way the function (f) is selected, they get obscured, and are hard to interpret. Flexible methods are better for inference, and inflexible methods are preferable for prediction.

Libraries

- **Python**
  - **Numpy** — Adds support for large, multi-dimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays
  - **Pandas** — Offers data structures and operations for manipulating numerical tables and time series
  - **Scikit-Learn** — It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.
  - **Tensorflow**
    - **Components**



Does lazy evaluation. Need to build the graph, and then run it in a session.

  - **MXNet** — Is an modern open-source deep learning framework used to train, and deploy deep neural networks. MXNet library is portable and can scale to multiple GPUs and multiple machines. MXNet is supported by major Public Cloud providers including AWS and Azure. Amazon has chosen MXNet as its deep learning framework of choice at AWS.
  - **Keras** — Is an open source neural network library written in Python. It is capable of running on top of MXNet, Deeplearning4j, Tensorflow, CNTK or Theano. Designed to enable fast experimentation with deep neural networks, it focuses on being minimal, modular and extensible.
  - **Torch** — Torch is an open source machine learning library, a scientific computing framework, and a script language based on the Lua programming language. It provides a wide range of algorithms for deep machine learning, and uses the scripting language LuaJIT, and an underlying C implementation.
  - **Microsoft Cognitive Toolkit** — Previously known as CNTK and sometimes styled as The Microsoft Cognitive Toolkit, is a deep learning framework developed by Microsoft Research. Microsoft Cognitive Toolkit describes neural networks as a series of computational steps via a directed graph.

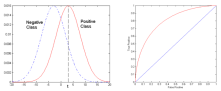
Motivation

- **Prediction** — When we are interested mainly in the predicted variable as a result of the inputs, but not on the each way of the inputs affect the prediction. In a real estate example, Prediction would answer the question of: Is my house over or under valued? Non-linear models are very good at these sort of predictions, but not great for inference because the models are much less interpretable.
- **Inference** — When we are interested in the way each one of the inputs affect the prediction. In a real estate example, Inference would answer the question of: How much would my house cost if it had a view of the sea? Linear models are more suited for inference because the models themselves are easier to understand than their non-linear counterparts.

Performance Analysis

- **Confusion Matrix**

- **Accuracy** — Fraction of correct predictions, not reliable as skewed when the data set is unbalanced (that is, when the number of samples in different classes vary greatly)
- **f1 score**
  - **Precision**
    - $(TP)/(TP + FP)$
      which tells us what proportion of patients we diagnosed as having cancer *actually* had cancer. In other words, proportion of TP in the set of positive cancer diagnoses. This is given by the *rightmost* column in the confusion matrix.
    - Out of all the examples the classifier labeled as positive, what fraction were correct?
  - **Recall**
    - $(TP)/(TP + FN)$
      which tells us what proportion of patients that *actually* had cancer were diagnosed by us as having cancer. In other words, proportion of TP in the set of true cancer states. This is given by the *bottom row* in the confusion matrix.
    - Out of all the positive examples there were, what fraction did the classifier pick up?
  - Harmonic Mean of Precision and Recall: (2 * p * r / (p + r))
- **ROC Curve - Receiver Operating Characteristics**



  - True Positive Rate (Recall / Sensitivity) vs False Positive Rate (1-Specificity)
- **Bias-Variance Tradeoff**
  - Bias refers to the amount of error that is introduced by approximating a real-life problem, which may be extremely complicated, by a simple model. If Bias is high, and/or if the algorithm performs poorly even on your training data, try adding more features, or a more flexible model.
  - Variance is the amount our model's prediction would change when using a different training data set. High: Remove features, or obtain more data.
- **Goodness of Fit = R^2** — 1.0 - sum_of_squared_errors / total_sum_of_squares(y)
- **Mean Squared Error (MSE)**
  - The mean squared error (MSE) or mean squared deviation (MSD) of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors or deviations—that is, the difference between the estimator and what is estimated.

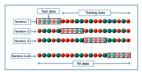
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2,$$

- **Error Rate**
  - The proportion of mistakes made if we apply out estimate model function the training observations in a classification setting.

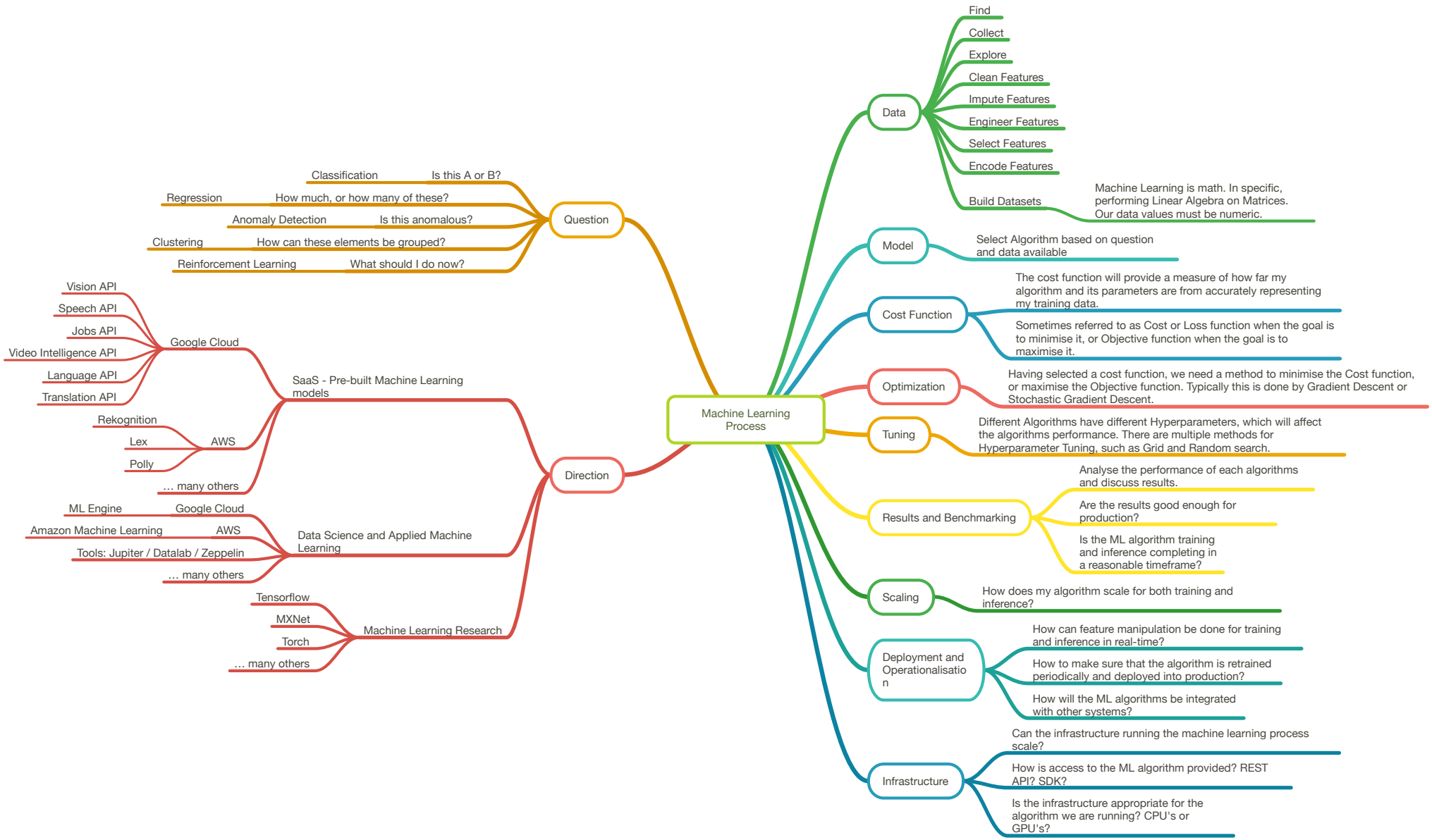
$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i).$$

Tuning

- **Cross-validation**
  - One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (called the validation set or testing set). To reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds.



  - **Methods**
    - Leave-p-out cross-validation
    - Leave-one-out cross-validation
    - k-fold cross-validation
    - Holdout method
    - Repeated random sub-sampling validation
- **Hyperparameters**
  - **Grid Search** — The traditional way of performing hyperparameter optimization has been grid search, or a parameter sweep, which is simply an exhaustive searching through a manually specified subset of the hyperparameter space of a learning algorithm. A grid search algorithm must be guided by some performance metric, typically measured by cross-validation on the training set or evaluation on a held-out validation set.
  - **Random Search** — Since grid searching is an exhaustive and therefore potentially expensive method, several alternatives have been proposed. In particular, a randomized search that simply samples parameter settings a fixed number of times has been found to be more effective in high-dimensional spaces than exhaustive search.
  - **Gradient-based optimization** — For specific learning algorithms, it is possible to compute the gradient with respect to hyperparameters and then optimize the hyperparameters using gradient descent. The first usage of these techniques was focused on neural networks. Since then, these methods have been extended to other models such as support vector machines or logistic regression.
- **Early Stopping (Regularization)** — Early stopping rules provide guidance as to how many iterations can be run before the learner begins to over-fit, and stop the algorithm then.
- **Overfitting** — When a given method yields a small training MSE (or cost), but a large test MSE (or cost), we are said to be overfitting the data. This happens because our statistical learning procedure is trying too hard to find patters in the data, that might be due to random chance, rather than a property of our function. In other words, the algorithms may be learning the training data too well. If model overfits, try removing some features, decreasing degrees of freedom, or adding more data.
- **Underfitting** — Opposite of Overfitting. Underfitting occurs when a statistical model or machine learning algorithm cannot capture the underlying trend of the data. It occurs when the model or algorithm does not fit the data enough. Underfitting occurs if the model or algorithm shows low variance but high bias (to contrast the opposite, overfitting from high variance and low bias). It is often a result of an excessively simple model.
- **Bootstrap** — Test that applies Random Sampling with Replacement of the available data, and assigns measures of accuracy (bias, variance, etc.) to sample estimates.
- **Bagging** — An approach to ensemble learning that is based on bootstrapping. Shortly, given a training set, we produce multiple different training sets (called bootstrap samples), by sampling with replacement from the original dataset. Then, for each bootstrap sample, we build a model. The results in an ensemble of models, where each model votes with the equal weight. Typically, the goal of this procedure is to reduce the variance of the model of interest (e.g. decision trees).


Machine Learning Process
Question
Classification
Is this A or B?
Regression
How much, or how many of these?
Anomaly Detection
Is this anomalous?
Clustering
How can these elements be grouped?
Reinforcement Learning
What should I do now?
Direction
SaaS - Pre-built Machine Learning models
Google Cloud
Vision API
Speech API
Jobs API
Video Intelligence API
Language API
Translation API
AWS
Rekognition
Lex
Polly
… many others
Data Science and Applied Machine Learning
Google Cloud
ML Engine
AWS
Amazon Machine Learning
Tools: Jupiter / Datalab / Zeppelin
… many others
Machine Learning Research
Tensorflow
MXNet
Torch
… many others
Data
Find
Collect
Explore
Clean Features
Impute Features
Engineer Features
Select Features
Encode Features
Build Datasets
Machine Learning is math. In specific, performing Linear Algebra on Matrices. Our data values must be numeric.
Model
Select Algorithm based on question and data available
Cost Function
The cost function will provide a measure of how far my algorithm and its parameters are from accurately representing my training data.
Sometimes referred to as Cost or Loss function when the goal is to minimise it, or Objective function when the goal is to maximise it.
Optimization
Having selected a cost function, we need a method to minimise the Cost function, or maximise the Objective function. Typically this is done by Gradient Descent or Stochastic Gradient Descent.
Tuning
Different Algorithms have different Hyperparameters, which will affect the algorithms performance. There are multiple methods for Hyperparameter Tuning, such as Grid and Random search.
Results and Benchmarking
Analyse the performance of each algorithms and discuss results.
Are the results good enough for production?
Is the ML algorithm training and inference completing in a reasonable timeframe?
Scaling
How does my algorithm scale for both training and inference?
Deployment and Operationalisation
How can feature manipulation be done for training and inference in real-time?
How to make sure that the algorithm is retrained periodically and deployed into production?
How will the ML algorithms be integrated with other systems?
Infrastructure
Can the infrastructure running the machine learning process scale?
How is access to the ML algorithm provided? REST API? SDK?
Is the infrastructure appropriate for the algorithm we are running? CPU's or GPU's?

Machine Learning Mathematics

Linear Algebra

Matrices

Almost all Machine-Learning algorithms use Matrix algebra in one way or another. This is a broad subject, too large to be included here in its full length. Heere's a start: https://en.wikipedia.org/wiki/Matrix_(mathematics)

- Basic Operations: Addition, Multiplication, Transposition
- Transformations
- Trace, Rank, Determinante, Inverse

Eigenvectors and Eigenvalues

$$Ax = \begin{pmatrix} 1 & 2 \\ 4 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} = 5 \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \lambda x.$$

In linear algebra, an eigenvector or characteristic vector of a linear transformation T from a vector space V over a field F into itself is a non-zero vector that does not change its direction when that linear transformation is applied to it.

http://setosa.io/ev/eigenvectors-and-eigenvalues/

Relatives Chain Rule

Rule

$$F'(x) = f'(g(x))g'(x).$$

Leibniz Notation

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$$

Jacobian Matrix

The matrix of all first-order partial derivatives of a vector-valued function. When the matrix is a square matrix, both the matrix and its determinant are referred to as the Jacobian in literature.

$$J = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial \mathbf{x}}, & \frac{\partial \mathbf{f}}{\partial \mathbf{y}}, & \frac{\partial \mathbf{f}}{\partial \mathbf{z}} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_2}{\partial x_n} \end{bmatrix}$$

Gradient

The gradient is a multi-variable generalization of the derivative. The gradient is a vector-valued function, as opposed to a derivative, which is scalar-valued.

$$\nabla f = \frac{\partial f}{\partial x}\mathbf{i} + \frac{\partial f}{\partial y}\mathbf{j} + \frac{\partial f}{\partial z}\mathbf{k}$$

Tensors

For Machine Learning purposes, a Tensor can be described as a Multidimentional Matrix. Depending on the dimensions, the Tensor can be a Scalar, a Vector, a Matrix, or a Multidimentional Matrix.

When measuring the forces applied to an object, the forces and the resulting force values in a multidimensional matrix.

Curse of Dimensionality

When the dimensionality increases, the volume of the space increases so fast that the available data become sparse. This sparsity is problematic for any method that requires statistical significance. In order for a statistical result to be supported by data, the amount of data needed to support the result often grows exponentially with the dimensionality.

Statistics

Measures of Central Tendency

Mean

Median

Value in the middle or an ordered list, or average of two in middle.

Mode

Most Frequent Value

Quantile

Division of probability distributions based on contiguous intervals with equal probabilities. In short: Dividing observations numbers in a sample list equally.

Dispersion

Range

Medium Absolute Deviation (MAD)

The average of the absolute value of the deviation of each value from the mean.

Inter-quartile Range (IQR)

Three quarties divide the data in approximately four equally divided parts.

Variance

Definition

The average of the squared differences from the Mean. Formally, is the expectation of the squared deviation of a random variable from its mean, and it informally measures how far a set (random) numbers are spread out from their mean.

$$\mathrm{Var}(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2.$$

where μ is the expected value, i.e.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Discrete

Types

$$\mathrm{Var}(X) = \sigma^2 = \int (x-\mu)^2 f(x)\,dx = \int x^2 f(x)\,dx - \mu^2$$

Continuous

Standard Deviation

The signed number of standard deviations an observation or datum is above the mean.

z-score/value/factor

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

sqr(variance)

Correlation

Covariance

A measure of how much two random variables change together. http://stats.stackexchange.com/questions/18058/how-would-you-explain-covariance-to-someone-who-understands-only-the-mean

$$\mathrm{cov}(x,y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

```
double mean(x), dx, mean(y) // (n - 1)
```

Relationship

Pearson

$$\rho_{X,Y} = \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y}$$

Benchmarke linear relationship, most appropriate for measurements taken from an interval scale, is a measure of the linear dependence between two variables.

Spearman

Benchmarke monotonic relationship (whether linear or not). Spearman's coefficient is appropriate for both continuous and discrete variables, including ordinal variables.

Kendall

Is a statistic used to measure the ordinal association between two measured quantities.

Contrary to the Spearman correlation, the Kendall correlation is not affected by how far from each other ranks are but only by whether the ranks between observations are equal or not, and is thus only appropriate for discrete variables but not defined for continuous variables.

Co-occurence

The results are presented in a matrix format, where the cross tabulation of two fields is a cell value. The cell value represents the percentage of times that the two fields exist in the same events.

Null Hypothesis

Is a general statement or default position that there is no relationship between two measured phenomena, or no association among groups. The null hypothesis is generally assumed to be true until evidence indicates otherwise.

Techniques

p-value

Five heads in a row Example

This demonstrates that specifying a direction (on a symmetric test statistic) halves the p-value (increases the significance and can mean the difference between data being considered significant or not).

Suppose a researcher flips a coin five times in a row and assumes a null hypothesis that the coin is fair. The test statistic of "total number of heads" can be one-tailed or two-tailed: a one-tailed test corresponds to seeing if the coin is biased towards heads, but a two-tailed test corresponds to seeing if the coin is biased either way. The researcher flips the coin five times and observes heads each time (HHHHH), yielding a test statistic of 5. In a one-tailed test, this is the upper extreme of all possible outcomes, and yields a p-value of $1/2^5 = 1/32 = 0.03$. If assuming a significance level of 0.05, then this result is sufficient to indicate statistical significance, meaning that the coin is fair would be rejected. In a two-tailed test, a test statistic of zero heads (TTTTT) is just as extreme and thus the data of HHHHH would yield a p-value of $2\times(1/2^5) = 1/16 = 0.06$, which is not significant at the 0.05 level.

If a p-value includes plain area in the probability of an observed result or more extreme results, then it is significant.

In this method, as part of experimental design, before performing the experiment, one first chooses a model (the null hypothesis) and a threshold value for p, called the "significance level", traditionally 5% or 1% and denoted as α. If the p-value is less than the chosen significance level (α), that suggests that the observed data is sufficiently inconsistent with the null hypothesis that the null hypothesis may be rejected. However, that does not prove that the tested hypothesis is true. For typical analysis, using the standard $\alpha = 0.05$ cutoff, the null hypothesis is rejected when $p < .05$ and not rejected when $p > .05$. The p-value is not the probability that a null hypothesis is true or the probability of the alternative hypothesis being true (in frequentist inference), but it is only a tool for deciding whether to reject the null hypothesis.

p-hacking

The process of data mining involves automatically testing huge numbers of hypotheses about a single data set by exhaustively searching for combinations of variables that might show a correlation. Conventional tests of statistical significance are based on the probability that an observation arose by chance, and necessarily accept some risk of mistaken test results, called the significance.

Central Limit Theorem

http://blog.xcln.me/posts/central-limit-theorem.html

States that a random variable defined as the average of a large number of independent and identically distributed random variables is itself approximately normally distributed.

Optimization

Gradient Descent

It is a first-order iterative optimization algorithm for finding the minimum of a function. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or of the approximate gradient) of the function at the current point. Instead one takes steps proportional to the positive of the gradient, and approaches a local maximum of that function; the procedure is then known as gradient ascent.

$$\theta^{new} = \theta^{old} - \alpha \nabla_\theta J_i(\theta)$$

Stochastic Gradient Descent (SGD)

Gradient descent uses total gradient over all examples per update, SGD updates after only 1 or few examples.

Mini-batch Stochastic Gradient Descent (MSGD)

Gradient descent uses total gradient over all examples per update, MSGD updates after only 1 example.

Momentum

Idea: Add a fraction γ of previous update to the current one. When the gradient is pointing in the same direction, this will increase the size of the steps taken towards the minimum.

Adagrad

Adaptive learning rates for each parameter

Regularization

L1 norm

L1-norm is also known as least absolute deviations (LAD), least absolute errors (LAE). It is basically minimizing the sum of the absolute differences (S) between the target value and the estimated values.

$$S = \sum_{i=1}^{n}|y_i - f(x_i)|.$$

Manhattan Distance

L2 norm

L2-norm is also known as least squares. It is basically minimizing the sum of the square of the differences (S) between the target value and the estimated values:

$$S = \sum_{i=1}^{n}(y_i - f(x_i))^2$$

Euclidean Distance

Early Stopping

Early stopping rules provide guidance as to how many iterations can be run before the learner begins to over-fit, and stop the algorithm then.

Dropout

Is a regularization technique for reducing overfitting in neural networks by preventing complex co-adaptations on training data. It is a very efficient way of performing model averaging with neural networks. The term "dropout" refers to dropping out units (both hidden and visible) in a neural network.

Sparse regularizer on columns

$$R(\mathbf{w}) = \sum_{i=1}^{d}|W|_{2,1}$$

This regularizer defines an L2 norm on each column and an L-1 norm over all columns. It can be solved by proximal methods.

Nuclear norm regularization

$R(\mathbf{w}) = \|\sigma(W)\|_1$ where $\sigma(W)$ is the eigenvalues in the singular value decomposition of W.

Mean-constrained regularization

$$\mathbf{w}_t - \eta(r - \frac{1}{T}\sum_{t'=1}^{T}\mathbf{w}_{t'})$$

This regularizer constrains the functions learned for each task to be similar to the overall average of the functions across all tasks. This is useful for expressing prior information that each task is expected to share similarities with each other task. An example is predicting blood iron levels measured at different times of the day, where each task represents a different person.

Clustered mean-constrained regularization

$$R(A - \bar{A}) + \sum_{i=1}^{k}\sum_{j \in I_i}(A - \bar{A}_i)^2 \text{ where } \{I_i\} \text{ is a cluster of tasks.}$$

This regularizer is similar to the mean-constrained regularizer, but instead enforces similarity between tasks within the same cluster. This can capture more complex prior information. This technique has been used to predict Netflix recommendations.

Graph-based similarity

$$R(\mathbf{w}) = \frac{1}{2}\sum_{s,t=1}^{T}(\mathbf{w}_s - \mathbf{w}_t)^2 \mathbf{M}_{st} \text{ is a pre-specified similarity matrix.}$$

More general than above, similarity between tasks can be defined by a function. The regularizer encourages the model to learn similar functions for similar tasks.

Probability

Concepts

Frequentist vs Bayesian Probability

- Frequentist — Basic notion of probability: # Results / # Attempts
- Bayesian — The probability is not a number, but a distribution itself.

http://www.behind-the-enemy-lines.com/2008/01/are-you-bayesian-or-frequentist-or.html

Random Variable

In probability and statistics, a random variable, random quantity, aleatory variable or stochastic variable is a variable whose value is subject to variations due to chance (i.e. randomness, in a mathematical sense). A random variable can take on a set of possible different values (similarly to other mathematical variables), each with an associated probability, in contrast to other mathematical variables.

Expectation (Expected Value) of a Random Variable

$$\mathbf{E}[X] = \sum_{i=1}^{n} x_i p_i,$$

Discrete

$$\mathbf{E}[X] = \int_{-\infty}^{\infty} x f(x)\,dx$$

Same, for continuous variables

Independence

Two events are independent, statistically independent, or stochastically independent if the occurrence of one does not affect the probability of the other.

$$p(x, y) = p(x) \cdot p(y),$$

Conditionality

$$P(A \cap B) = P(A|B)P(B)$$

Bayes Theorem (rule, law)

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Simple Form

$$P(B) = \sum_i P(B \mid A_i)P(A_i),$$

$$\Rightarrow P(A_i \mid B) = \frac{P(B \mid A_i)P(A_i)}{\sum_i P(B \mid A_i)\,P(A_i)}$$

With Law of Total probability

Marginalisation

The marginal distribution of a subset of a collection of random variables is the probability distribution of the variables contained in the subset. It gives the probabilities of various values of the variables in the subset without reference to the values of the other variables.

Continuous

$$p(x) = \int_y p(x, y)\,dy.$$

Discrete

$$p(x) = sum_y p(x, y).$$

$$\Pr(X = x) = \sum_y \Pr(X = x, Y = y) = \sum_y \Pr(X = x | Y = y) \Pr(Y = y).$$

Law of Total Probability

Is a fundamental rule relating marginal probabilities to conditional probabilities. It expresses the total probability of an outcome which can be reached via several distinct events - hence the name.

$$\Pr(A) = \sum_n \Pr(A \cap B_n)$$

or, alternatively[1],

$$\Pr(A) = \sum_n \Pr(A \mid B_n)\Pr(B_n),$$

Chain Rule

$$\Pr\left(\bigcap_{k=1}^{n} A_k\right) = \prod_{k=1}^{n} \Pr\left(A_k \middle| \bigcap_{j=1}^{k-1} A_j\right)$$

With four variables, the chain rule produces the product of conditional probabilities:

$$\Pr(A_4, A_3, A_2, A_1) = \Pr(A_4 \mid A_3, A_2, A_1) \cdot \Pr(A_3 \mid A_2, A_1) \cdot \Pr(A_2 \mid A_1) \cdot \Pr(A_1)$$

Permits the calculation of any member of the joint distribution of a set of random variables using only conditional probabilities.

Bayesian Inference

$$P(H \mid E) = \frac{P(E \mid H) \cdot P(H)}{P(E)}$$

Bayesian inference derives the posterior probability as a consequence of two antecedents: a prior probability and a "likelihood function" derived from a statistical model for the observed data. Bayesian inference computes the posterior probability according to Bayes' theorem. It can be applied iteratively so to update the confidence on out hypothesis.

Distributions

Definition

Is a table or an equation that ties each outcome of a statistical experiment with the probability of occurrence. When Continuous, is is described by the Probability Density Function.

Types (Density Function)

Normal (Gaussian)

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Where:

- μ is mean or expectation of the distribution (and also its median and mode),
- σ is standard deviation
- σ^2 is variance

Bernoulli

$$f(k; p) = \begin{cases} p & \text{if } k = 1, \\ 1 - p & \text{if } k = 0. \end{cases}$$

This can also be expressed as

$$f(k; p) = p^k(1-p)^{1-k} \quad \text{for } k \in \{0, 1\}.$$

Gamma

$$f(x; \alpha, \beta) = \text{constant} \cdot x^{\alpha-1}(1-x)^{\beta-1}$$

$$= \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\int_0^1 u^{\alpha-1}(1-u)^{\beta-1}\,\mathrm{d}u}$$

$$= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}$$

$$= \frac{1}{\mathrm{B}(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1}$$

Binomial

$$p(k) = \binom{n}{k} p^k (1-p)^{n-k},$$

Poisson

$$f(k; \lambda) = \Pr(X = k) = \frac{\lambda^k e^{-\lambda}}{k!},$$

where:

- e is Euler's number ($e \approx 2.718281...$)
- $k!$ is the factorial of k.

Uniform

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } a \le x \le b, \\ 0 & \text{for } x < a \text{ or } x > b \end{cases}$$

Cumulative Distribution Function (CDF)

$$F_X(x) = \int_{-\infty}^{x} f_X(t)\,dt.$$

Density Estimation

Methods

Kernel Density Estimation

Kernel	Equation
Uniform	$k_0(u) = \frac{1}{2}\mathbb{1}\{|u| \le 1\}$
Epanechnikov	$k_1(u) = \frac{3}{4}\left(1 - u^2\right)\mathbb{1}\{|u| \le 1\}$
Biweight	$k_2(u) = \frac{15}{16}\left(1 - u^2\right)^2\mathbb{1}\{|u| \le 1\}$
Triweight	$k_3(u) = \frac{35}{32}\left(1 - u^2\right)^3\mathbb{1}\{|u| \le 1\}$
Gaussian	$k_4(u) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{u^2}{2}\right)$

- non-negative
- it's a type of PDF that it is symmetric
- real-valued
- symmetric
- integral over function is equal to 1
- non-parametric
- calculates kernel distributions for every sample point, and then adds all the distributions
- Uniform, Triangle, Quartic, Triweight, Gaussian, Cosine, others...

Cubic Spline

A cubic spline is a function created from cubic polynomials on each between-knot interval by pasting them together twice continuously differentiable at the knots.

Information Theory

Entropy

$$H = -\sum_x p_i \log_2(p_i)$$

Entropy is a measure of unpredictability of information content.

$$H(p, q) = -\sum_x p(x) \log q(x).$$

Cross entropy between two probability distributions p and q, over the same underlying set of events measures the average number of bits needed to identify an event drawn from the set, if a coding scheme is used that is optimized for an "unnatural" probability distribution q, rather than the "true" distribution p.

Cross Entropy

$$H(X, Y) = \mathbb{E}_{X,Y}[-\log p(x, y)] = -\sum_{x,y} p(x, y) \log p(x, y)$$

Joint Entropy

$$H(X|Y) = \mathbb{E}_Y[H(X|y)] = -\sum_y p(y) \sum_x p(x|y) \log p(x|y) = -\sum_{x,y} p(x, y) \log \frac{p(x, y)}{p(y)},$$

Conditional Entropy

$$I(X; Y) = \mathbb{E}_{X,Y}[SI(x, y)] = \sum_{x,y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$$

Mutual Information

$$D_{\mathrm{KL}}(p(X) \| q(X)) = \sum_{x \in X} -p(x) \log q(x) - \sum_{x \in X} -p(x) \log p(x) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}$$

Kullback-Leibler Divergence

Mostly Non-Parametric. Parametric makes assumptions on my data/random-variables, for instance, that they are normally distributed. Non-parametric does not.

The methods are generally intended for description rather than formal inference

Loss Functions (Min Objective/Max Functions)

Maximum Likelihood Estimation (MLE)

Many cost functions are the result of applying Maximum Likelihood. For instance, the Least Squares cost function can be obtained via Maximum Likelihood. Cross Entropy is another example.

The likelihood of a parameter value (or vector of parameter values), θ, given outcomes x, is equal to the probability (density) assumed for those observed outcomes given those parameter values, that is

$$\mathcal{L}(\theta \mid x) = P(x \mid \theta).$$

The natural logarithm of the likelihood function, called the log-likelihood, is more convenient to work with. Because the logarithm is a monotonically increasing function, the logarithm of a function achieves its maximum value at the same points as the function itself, and hence the log-likelihood can be used in place of the likelihood in maximum likelihood estimation and related techniques.

$$f(x_1, x_2, \ldots, x_n \mid \theta) = f(x_1 \mid \theta) \times f(x_2 \mid \theta) \times \cdots \times f(x_n \mid \theta).$$

In general, for a fixed set of data and underlying statistical model, the method of maximum likelihood selects the set of values of the model parameters that maximizes the likelihood function. Intuitively, this maximizes the "agreement" of the selected model with the observed data, and for discrete random variables it maximizes the probability of the observed data under the resulting distribution. Maximum-likelihood estimation gives a unified approach to estimation, which is well-defined in the case of the normal distribution and many other problems.

$$\mathcal{L}(\theta; x_1, \ldots, x_n) = f(x_1, x_2, \ldots, x_n \mid \theta) = \prod_{i=1}^{n} f(x_i \mid \theta).$$

$$\ln \mathcal{L}(\theta; x_1, \ldots, x_n) = \sum_{i=1}^{n} \ln f(x_i \mid \theta).$$

$$\hat{\ell}(\theta; x) = \frac{1}{n}\sum_{i=1}^{n} \ln f(x_i \mid \theta),$$

$$\{\hat{\theta}_{\max}\} \subseteq \{\arg\max_{\theta \in \Theta} \hat{\ell}(\theta; x_1, \ldots, x_n)\},$$

Cross-Entropy

Cross entropy can be used to define the loss function in machine learning and optimization. The true probability pi is the true label, and the given distribution qi is the predicted value of the current model.

$$H(p, q) = \mathbb{E}_p[\log \frac{1}{q(x_i)}]$$

$$H(p, q) = \sum_i p(x_i) \log \frac{1}{q(x_i)}$$

$$H(p, q) = -\sum_x p(x) \log q(x).$$

$$L(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log \hat{y}_n + (1 - y_n) \log(1 - \hat{y}_n)\right]$$

Cross-entropy error function and logistic regression

Logistic

The logistic loss function is defined as:

$$V(f(\vec{x}), y) = \frac{1}{\ln 2}\ln(1 + e^{-y f(\vec{x})})$$

Quadratic

The use of a quadratic loss function is common, for example when using least squares techniques. It is often more mathematically tractable than other loss functions because of the properties of variances, as well as being symmetric: an error above the target causes the same loss as the same magnitude of error below the target. If the target is t, then a quadratic loss function is:

$$\lambda(x) = C(t - x)^2$$

0-1 Loss

In statistics and decision theory, a frequently-used loss function is the 0-1 loss function

$$L(\hat{y}, y) = I[\hat{y} \neq y],$$

Hinge Loss

The hinge loss is a loss function used for training classifiers. For an intended output $t = \pm 1$ and a classifier score y, the hinge loss of the prediction y is defined as:

$$\ell(y) = \max(0, 1 - t \cdot y)$$

Exponential

$$L_2(y, y) = \exp[\frac{1}{2}\sum_y(y_i - \hat{y}_i)^2].$$

Hellinger Distance

It is used to quantify the similarity between two probability distributions. It is a type of f-divergence.

To define the Hellinger distance in terms of measure theory, let P and Q denote two probability measures that are absolutely continuous with respect to a third probability measure λ. The square of the Hellinger distance between P and Q is defined as the quantity

$$H^2(P, Q) = \frac{1}{2}\int\left(\sqrt{\frac{\mathrm{d}P}{\mathrm{d}\lambda}} - \sqrt{\frac{\mathrm{d}Q}{\mathrm{d}\lambda}}\right)^2 \mathrm{d}\lambda.$$

Kullback-Leibler Divergence

Is a measure of how one probability distribution diverges from a second expected probability distribution. Applications include characterizing the relative (Shannon) entropy in information systems, randomness in continuous time-series, and information gain when comparing statistical models of inference.

Discrete

$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

Continuous

$$D_{\mathrm{KL}}(P\|Q) = \int_x p \log \frac{p}{q}\,\mathrm{d}x,$$

Itakura-Saito distance

is a measure of the difference between an original spectrum $P(\omega)$ and an approximation $P_2(\omega)$ of that spectrum. Although it is not a periodogram measure, it is intended to reflect perceptual (dis)similarity.

$$D_{IS}(P(\omega), P_2(\omega)) = \frac{1}{2\pi}\int_{-\pi}^{\pi}\left[\frac{P(\omega)}{P_2(\omega)} - \log\frac{P(\omega)}{P_2(\omega)} - 1\right]\mathrm{d}\omega$$

Hinge Loss

$$\ell(y, t) = \max(0, 1 - t \cdot y)$$

Logistic Loss

$$\ell(y, t) = \frac{1}{\log_2(2)}\log(1 + e^{-t \cdot y})$$

Exponential Loss

$$\ell(y, t) = e^{-t \cdot y}$$

Savage Loss

$$\ell(y, t) = (1 - t \cdot y)^2$$

Tangent Loss

$$\ell(y, t) = (2\arctan(t \cdot y) - 1)^2$$

Cosine Loss

$$\ell(y, t) = 1 - \cos(t \cdot y)$$

Perceptron Loss

$$\ell(y, t) = \max(0, -t \cdot y)$$

Scaled Hinge Loss

$$\ell(y, t) = \max(0, 1 - t \cdot y)$$

Scaled Logistic Loss

$$\ell(y, t) = \frac{1}{\log_2(2)}\log(1 + e^{-t \cdot y})$$

Scaled Exponential Loss

$$\ell(y, t) = e^{-t \cdot y}$$

Scaled Savage Loss

$$\ell(y, t) = (1 - t \cdot y)^2$$

Scaled Tangent Loss

$$\ell(y, t) = (2\arctan(t \cdot y) - 1)^2$$

Scaled Cosine Loss

$$\ell(y, t) = 1 - \cos(t \cdot y)$$

Scaled Perceptron Loss

$$\ell(y, t) = \max(0, -t \cdot y)$$

https://stats.stackexchange.com/questions/154879/a-list-of-cost-functions-used-in-neural-networks-alongside-applications

https://en.wikipedia.org/wiki/Loss_function#For_classification

Language Model

To evaluate a language model, we should measure how much surprise it gives us for real sequences in that languages. For each real word encountered, the language model will give a probability p. And we use $\log_2(p)$ to quantify the surprise. And we average the total surprise over a long enough sequence. So, in case of a 1000-letter sequence with 500 A and 500 B, the surprise given by the 1/2-2/3 model will be: $1\cdot500\log_2(1/2) - 500\cdot\log_2(2/3)\cdot1000 = 1/2 - 1.42\log_2(2/3)$

Write the correct 1/2-1/3 model will give:

$1\cdot500\log_2(1/2) - 500\cdot\log_2(1/3)\cdot1000 = 1/2 - 1.58\log_2(1/3)$

So, we can see, the 1/3, 2/3 model gives more surprise, which indicates it is worse than the correct model. Only when the sequence is long enough, the average effect will mimic the expectation over the 1/2-1/2 distribution. If the sequence is short, it won't give a convincing result.

Machine Learning Models

Regression

- Linear Regression
  - $Y = \beta_0 + \beta_1 X + \epsilon$
  - $y_i = \beta_0 + \beta_1 x_i + e_i \quad \{\text{data}; i = 1 \ldots n\}$
- Generalised Linear Models (GLMs)
  - Is a flexible generalization of ordinary linear regression that allows for response variables that have error distribution models other than a normal distribution. The GLM generalizes linear regression by allowing the linear model to be related to the response variable via a link function and by allowing the magnitude of the variance of each measurement to be a function of its predicted value.
  - Link Function
    - Identity: $\mu = \mathbf{X}\boldsymbol{\beta}$
    - Inverse: $\mu = (\mathbf{X}\boldsymbol{\beta})^{-1}$
    - Logit: $\mu = \dfrac{\exp(\mathbf{X}\boldsymbol{\beta})}{1+\exp(\mathbf{X}\boldsymbol{\beta})} = \dfrac{1}{1+\exp(-\mathbf{X}\boldsymbol{\beta})}$
  - Cost Function is found via Maximum Likelihood Estimation
- Locally Estimated Scatterplot Smoothing (LOESS)
- Ridge Regression
- Least Absolute Shrinkage and Selection Operator (LASSO)
- Logistic Regression
  - $p(X) = \dfrac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}.$
  - Logistic Function

Bayesian

- $p(C_k \mid \mathbf{x}) = \dfrac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$
- $\hat{y} = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}}\ p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k).$
- Naive Bayes
  - Naive Bayes Classifier. We neglect the denominator as we calculate for every class and pick the max of the numerator
- Multinomial Naive Bayes
- Bayesian Belief Network (BBN)

Dimensionality Reduction

- Principal Component Analysis (PCA)
- Partial Least Squares Regression (PLSR)
- Principal Component Regression (PCR)
- Partial Least Squares Discriminant Analysis
- Quadratic Discriminant Analysis (QDA)
- Linear Discriminant Analysis (LDA)

Instance Based

- k-nearest Neighbour (kNN)
- Learning Vector Quantization (LVQ)
- Self-Organising Map (SOM)
- Locally Weighted Learning (LWL)

Decision Tree

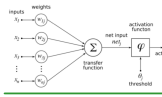
- Random Forest
- Classification and Regression Tree (CART)
- Gradient Boosting Machines (GBM)
- Conditional Decision Trees
- Gradient Boosted Regression Trees (GBRT)

Clustering

- Algorithms
  - Hierarchical Clustering
    - Linkage
      - complete
      - single
      - average
      - centroid
    - Dissimilarity Measure
      - Euclidean
        - Euclidean distance or Euclidean metric is the "ordinary" straight-line distance between two points in Euclidean space.
      - Manhattan
        - The distance between two points measured along axes at right angles.
  - k-Means
    - How many clusters do we select?
  - k-Medians
  - Fuzzy C-Means
  - Self-Organising Maps (SOM)
  - Expectation Maximization
  - DBSCAN
- Validation
  - Data Structure Metrics
    - Dunn Index
    - Connectivity
    - Silhouette Width
  - Stability Metrics
    - Non-overlap APN
    - Average Distance AD
    - Figure of Merit FOM
    - Average Distance Between Means ADM

Neural Networks

Unit (Neurons)

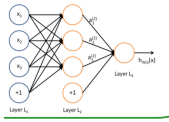


A unit often refers to the activation function in a layer by which the inputs are transformed via a nonlinear activation function (for example by the logistic sigmoid function). Usually, a unit has several incoming connections and several outgoing connections.

Input Layer

Comprised of multiple Real-Valued inputs. Each input must be linearly independent from each other.

Hidden Layers



Layers other than the input and output layers. A layer is the highest-level building block in deep learning. A layer is a container that usually receives weighted input, transforms it with a set of mostly non-linear functions and then passes these values as output to the next layer.

Batch Normalization

Using mini-batches of examples, as opposed to one example at a time, is helpful in several ways. First, the gradient of the loss over a mini-batch is an estimate of the gradient over the training set, whose quality improves as the batch size increases. Second, computation over a batch can be much more efficient than m computations for individual examples, due to the parallelism afforded by the modern computing platforms.

With SGD, the training proceeds in steps, and at each step we consider a mini-batch x1...m of size m. The mini-batch is used to approx- imate the gradient of the loss function with respect to the parameters.

Learning Rate

Neural networks are often trained by gradient descent on the weights. This means at each iteration we use backpropagation to calculate the derivative of the loss function with respect to each weight and subtract it from that weight.

However, if you actually try that, the weights will change far too much each iteration, which will make them "overcorrect" and the loss will actually increase/diverge. So in practice, people usually multiply each derivative by a small value called the "learning rate" before they subtract it from its corresponding weight.

- Tricks
  - Simplest recipe: keep it fixed and use the same for all parameters.
    - $\theta^{new} = \theta^{old} - \alpha \nabla_\theta J_t(\theta)$
  - Better results by allowing learning rates to decrease
    Options:
    - Reduce by 0.5 when validation error stops improving
    - Reduction by O(1/t) because of theoretical convergence guarantees, with hyper-parameters ε_0 and τ and t is iteration numbers.
      - $\alpha = \dfrac{\alpha_0 \tau}{\max(t, \tau)}$
    - Better yet: No hand-set learning of rates by using AdaGrad

Weight Initialization

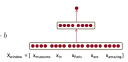
- All Zero Initialization
  - In the ideal situation, with proper data normalization it is reasonable to assume that approximately half of the weights will be positive and half of them will be negative. A reasonable-sounding idea then might be to set all the initial weights to zero, which you expect to be the "best guess" in expectation.
  - But, this turns out to be a mistake, because if every neuron in the network computes the same output, then they will also all compute the same gradients during back-propagation and undergo the exact same parameter updates. In other words, there is no source of asymmetry between neurons if their weights are initialized to be the same.
- Initialization with Small Random Numbers
  - Thus, you still want the weights to be very close to zero, but not identically zero. In this way, you can random these neurons to small numbers which are very close to zero, and it is treated as symmetry breaking. The idea is that the neurons are all random and unique in the beginning, so they will compute distinct updates and integrate themselves as diverse parts of the full network.
  - The implementation for weights might simply drawing values from a normal distribution with zero mean, and unit standard deviation. It is also possible to use small numbers drawn from a uniform distribution, but this seems to have relatively little impact on the final performance in practice.
- Calibrating the Variances
  - One problem with the above suggestion is that the distribution of the outputs from a randomly initialized neuron has a variance that grows with the number of inputs. It turns out that you can normalize the variance of each neuron's output to 1 by scaling its weight vector by the square root of its fan-in (i.e., its number of inputs)
  - This ensures that all neurons in the network initially have approximately the same output distribution and empirically improves the rate of convergence. The detailed derivations can be found from Page. 18 to 23 of the slides. Please note that, in the derivations, it does not consider the influence of ReLU neurons.

Backpropagation

- Is a method used in artificial neural networks to calculate the error contribution of each neuron after a batch of data. It calculates the gradient of the loss function. It is commonly used in the gradient descent optimization algorithm. It is also called backward propagation of errors, because the error is calculated at the output and distributed back through the network layers.
- Neural Network taking 4 dimension vector representation of words.

$$s = U^T f(Wx + b) \qquad x \in \mathbb{R}^{20\times 1}, W \in \mathbb{R}^{8\times 20}, U \in \mathbb{R}^{8\times 1}$$

$$\begin{aligned} s &= U^T a \\ a &= f(z) \\ z &= Wx + b \end{aligned}$$



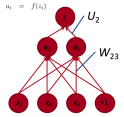
$$J = \max(0, 1 - s + s_c)$$

$$\begin{aligned} s &= U^T f(Wx + b) \\ s_c &= U^T f(Wx_c + b) \end{aligned}$$

$$\frac{\partial s}{\partial x} = W^T \delta$$

$$\begin{aligned} \frac{\partial s}{\partial x_j} &= \sum_{i=1}^{2} \frac{\partial s}{\partial a_i} \frac{\partial a_i}{\partial x_j} \\ &= \sum_{i=1}^{2} \frac{\partial U^T a}{\partial a_i} \frac{\partial a_i}{\partial x_j} \\ &= \sum_{i=1}^{2} U_i \frac{\partial f(W_i x + b)}{\partial x_j} \\ &= \sum_{i=1}^{2} U_i f'(W_i x + b) \frac{\partial W_i x}{\partial x_j} \\ &= \sum_{i=1}^{2} \delta_i W_{ij} \\ &= \delta^T W_j \end{aligned}$$

$$\begin{aligned} z_i &= W_i x + b_i = \sum_{j=1}^{8} W_{ij} x_j + b_i \\ a_i &= f(z_i) \end{aligned}$$



$$\frac{\partial J}{\partial U} = 1\{1 - s + s_c > 0\}(-f(Wx + b) + f(Wx_c + b))$$

$$\frac{\partial J}{\partial U} = 1\{1 - s + s_c > 0\}(-a + a_c)$$

- In this method, we reuse partial derivatives computed for higher layers in lower layers, for efficiency.

Activation Functions

- Defines the output of that node given an input or set of inputs.
- Types
  - ReLU
    - $f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$
  - Sigmoid / Logistic
    - $f(x) = \dfrac{1}{1+e^{-x}}$
  - Binary
    - $f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
  - Tanh
    - $f(x) = \tanh(x) = \dfrac{2}{1+e^{-2x}} - 1$
  - Softplus
    - $f(x) = \ln(1 + e^x)$
  - Softmax
    - $f_i(\vec{x}) = \dfrac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}} \quad \text{for } i = 1, \ldots, J$
  - Maxout
    - $f(\vec{x}) = \max_i x_i$
  - Leaky ReLU, PReLU, RReLU, ELU, SELU, and others.