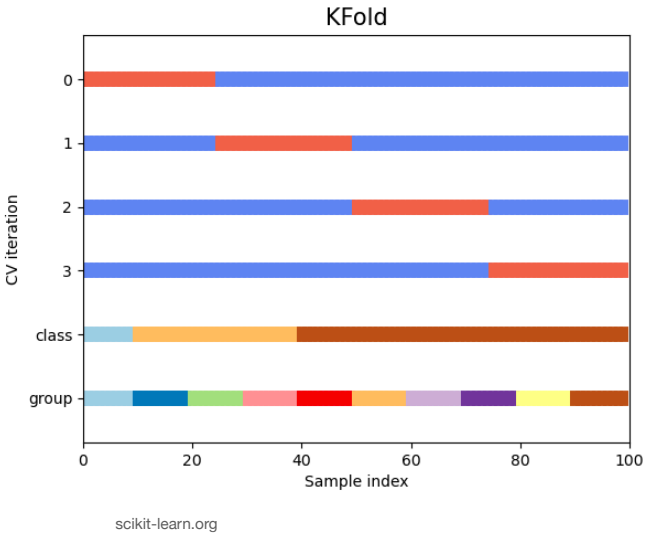
Stratified cross-validation

- The folds in cross-validation are usually selected at random
- This might lead to selecting, by chance, a fold with unbalanced/unrepresentative observations
  - Unbalanced classes
  - Irregular data
- Stratified cross-validation seeks to ensure that each fold represents the strata of the data
  - Performed in a supervised way for classification tasks
- Why and when is stratification important?
  - Classification algorithms tend to weight each observation the same, therefore overrepresented classes are given too much weight in comparison to other classes
    - Algorithms that assign equal weights to each class are not affected by stratification
  - Many algorithms are unable to learn representation for a class that is absent in the dataset

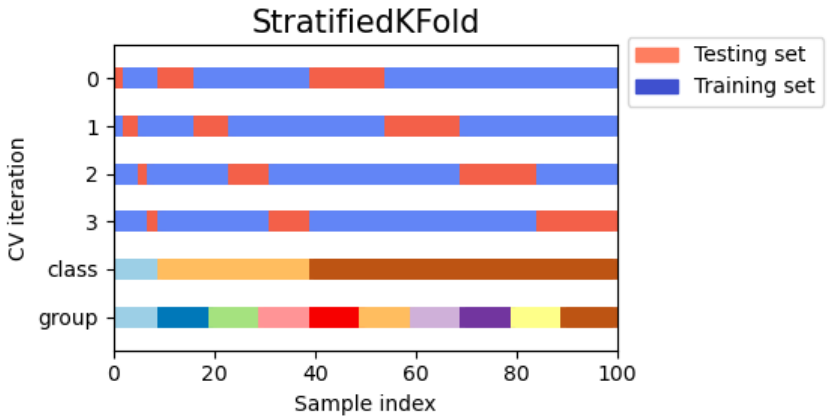
Stratified cross-validation (cont'd)

Demonstrating the problem:
(non-stratified cross-validation)



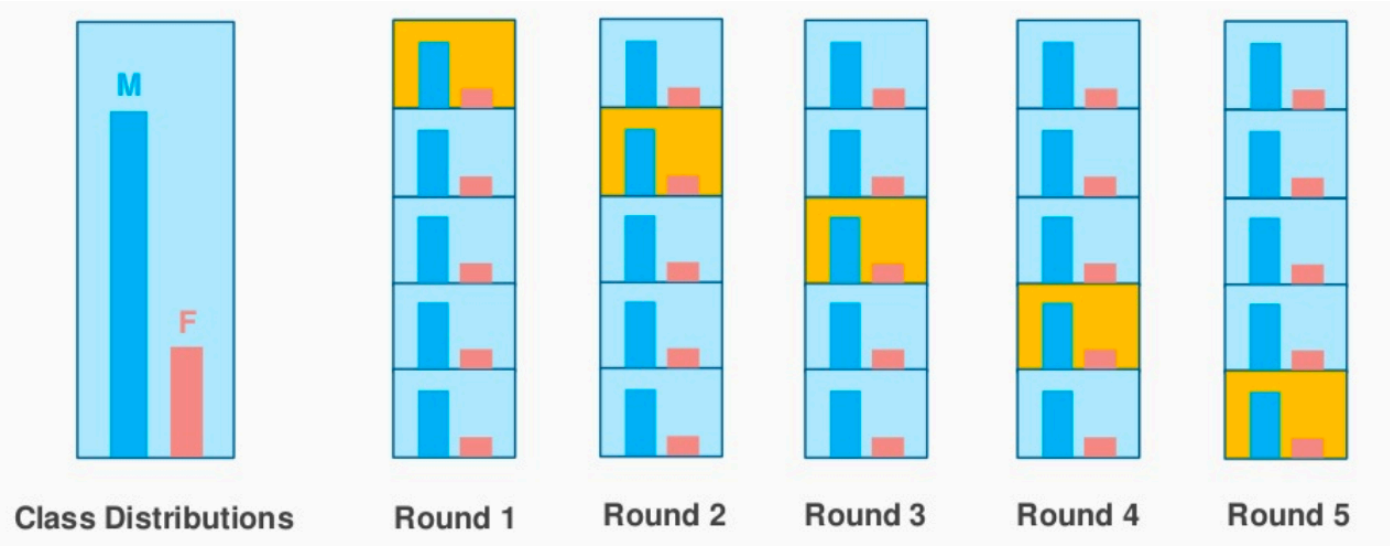
Stratified cross-validation (cont'd)

Demonstrating the solution:
(stratified cross-validation)



scikit-learn.org

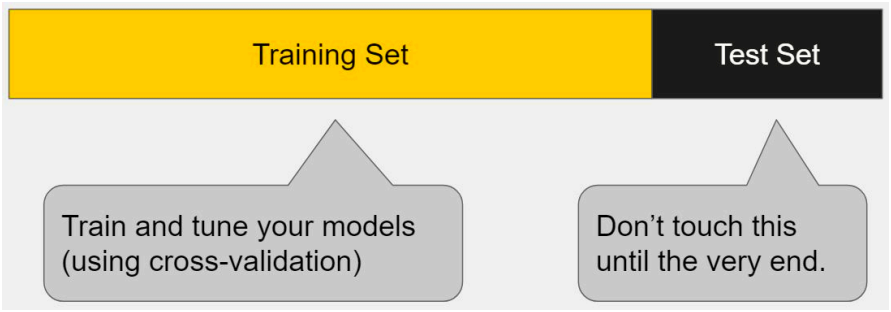
Stratified cross-validation (cont'd)



stats.stackexchange.com

Building the final model after cross-validation

- Because after X-fold cross-validation we have X models
  - E.g. 5-fold cross-validation gives us 5 different models, across which we average the performance
- Normally, if the performance is acceptable, we build the final model from all the folds
- Run the final test on the set-aside test data using the final model



https://elitedatascience.com

Other ways to do validation

- Leave-one-out validation
- Leave-pair-out validation
- …

How do we evaluate our model?

- Train your model, test it on set aside observations with known output variable (supervised learning)
  - Compare estimated/predicted $\hat{y}$ to y
    - In case of classification compare the labels
      - True/false
    - In case of regression compare the response values
      - How far off?

Evaluating the model for classification problems

- Accuracy - how many labels are we predicting correct?

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

https://towardsdatascience.com

Sensitivity

- How many true positives are we predicting?
- How well are we detecting/sensing correct labels
- How many of the positive cases did we detect?
- Often called “true positive rate” or “recall”

- If we are classifying cancer vs. not
  - Rate at which we correctly predict cancer observations from all true cancer cases

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

https://en.wikipedia.org

Specificity

- How well are we correctly predicting that the observation is not of a particular class?
- How many of the negative cases did we detect?
- Often called “true negative rate” or “false positive rate”

- If we are classifying cancer vs. not
  - Rate at which we correctly predict observations as non-cancer from all true non-cancer cases

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{N}} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

https://en.wikipedia.org

Precision

- Rate at which positive cases are being identified correctly within all predicted as positive
- Often called “positive predictive value” or “PPV”

- If we are classifying cancer vs. not
  - Rate at which we correctly predict observations as cancer from all predicted as cancer

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

https://en.wikipedia.org

Confusion matrix

- Often used to demonstrate model sensitivity and specificity in one table
- Sometimes called “error matrix”

		Actual class	
		P	**N**
Predicted class	**P**	**TP**	FP
	N	FN	**TN**

https://en.wikipedia.org

Confusion matrix

- Often used to demonstrate model sensitivity and specificity in one table
- Sometimes called “error matrix”

		Actual class	
		P	**N**
Predicted class	**P**	**TP**	FP
	N	FN	**TN**

https://en.wikipedia.org

	Actual positive (1)	Actual negative (0)
Predicted positive (1)	TP	FP
Predicted negative (0)	FN	TN

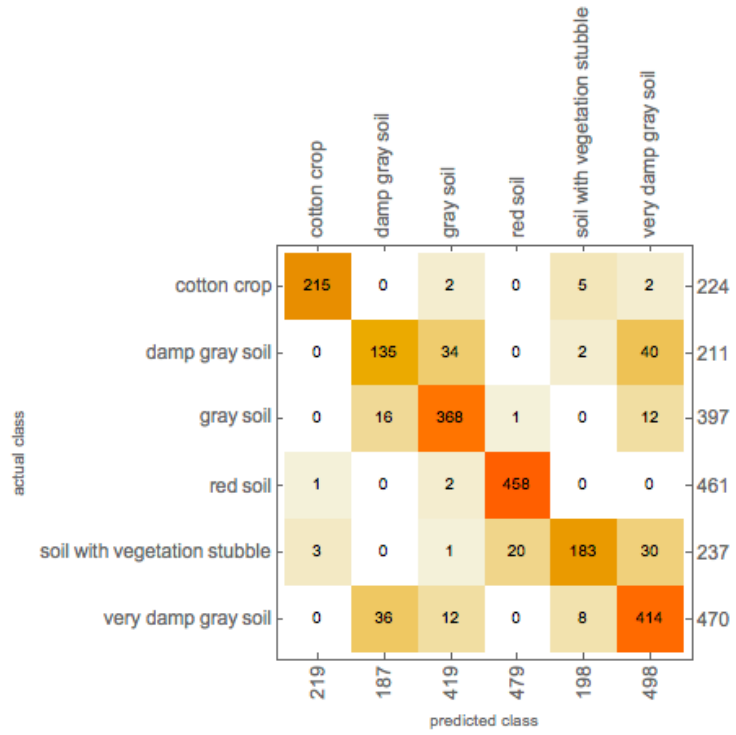
https://statisticaloddsandends.wordpress.com

Confusion matrix (cont’d)

	Actual positive (1)	Actual negative (0)
Predicted positive (1)	5	50
Predicted negative (0)	10	10000

https://statisticaloddsandends.wordpress.com

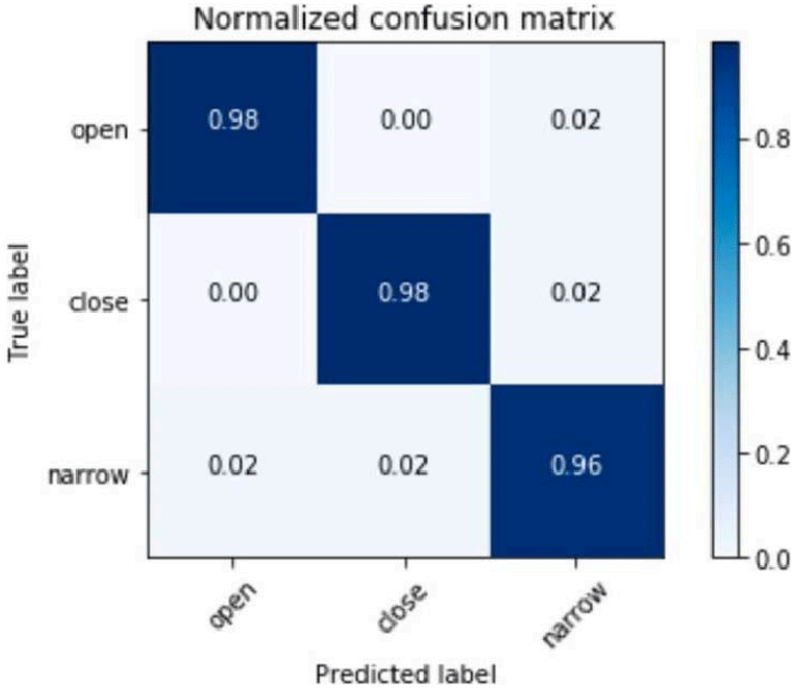
Confusion matrix (cont'd)



www.wolfram.com

A confusion matrix can represent individual classes

Confusion matrix (cont'd)



A confusion matrix can contain normalized values, percentages, or ratios

"Automatic Classification of Anterior Chamber Angle Using Ultrasound Biomicroscopy and Deep Learning" 2019 Translational Vision Science & Technology

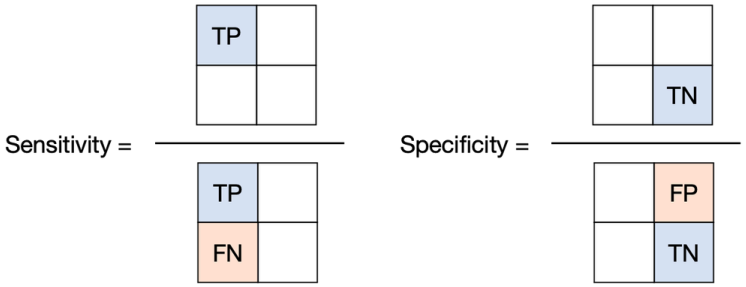
Confusion matrix - deeper look

		True condition				
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$	F_1 score = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$		

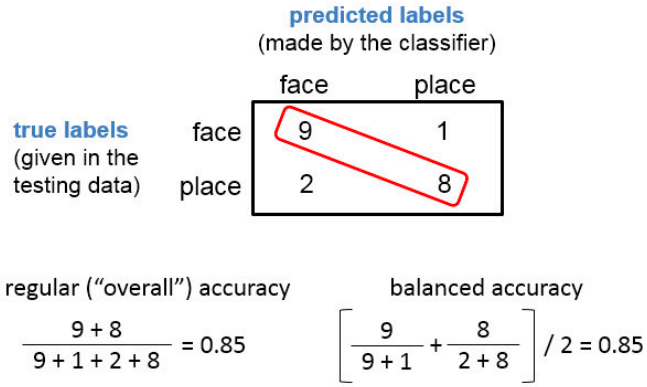
https://en.wikipedia.org

Balanced accuracy

- Balanced accuracy - arithmetic mean of sensitivity and specificity



https://statisticaloddsandends.wordpress.com



regular ("overall") accuracy

$$\frac{9+8}{9+1+2+8} = 0.85$$

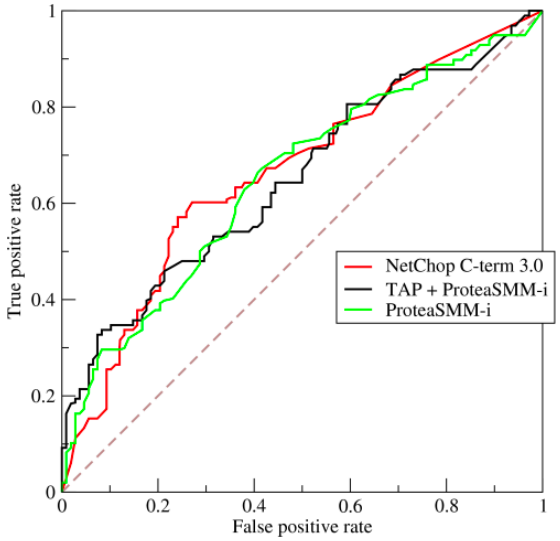
balanced accuracy

$$\left[\frac{9}{9+1} + \frac{8}{2+8}\right] / 2 = 0.85$$

http://mvpa.blogspot.com
Jo Etzel

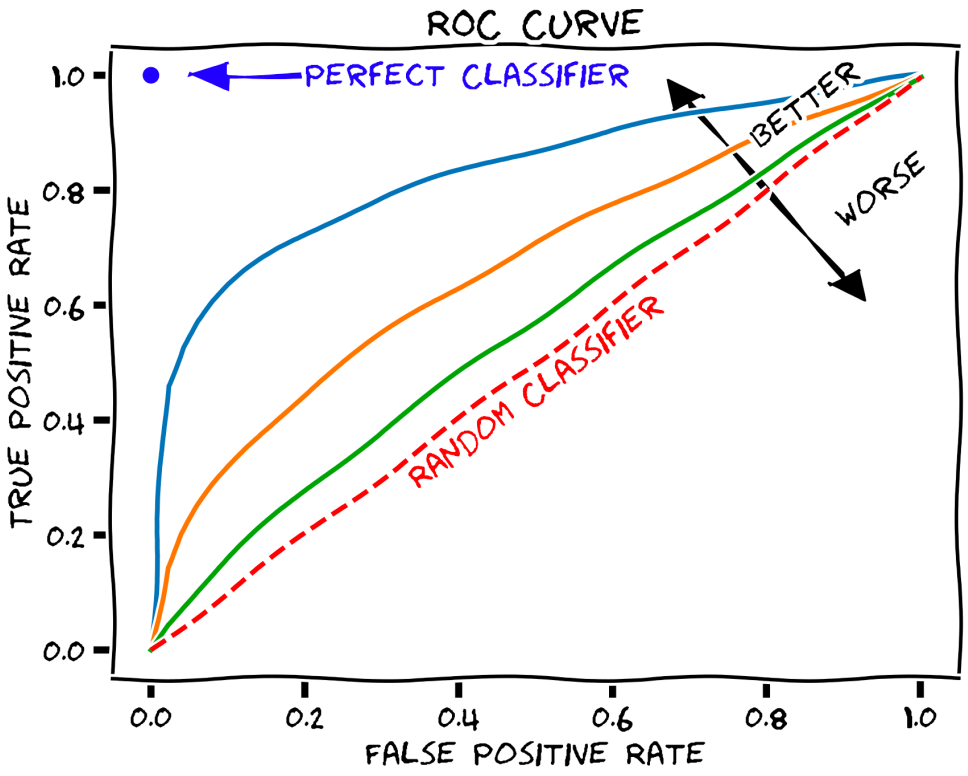
ROC curve

- Receiver operating characteristic (ROC)
- Graphically illustrates the performance of a binary classifier by plotting specificity (false positive rate) vs. sensitivity (true positive rate)
  - Plot specificity vs. sensitivity at different thresholds



https://en.wikipedia.org

ROC curve (cont'd)

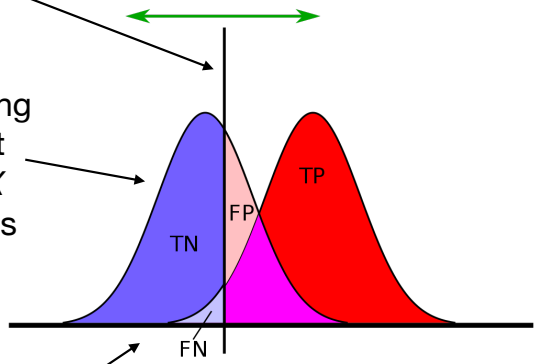


https://en.wikipedia.org

ROC curve (cont'd)

Separation line learned by this classifier

Two distributions representing two classes in this dataset and distribution of some *X* input variable for each class

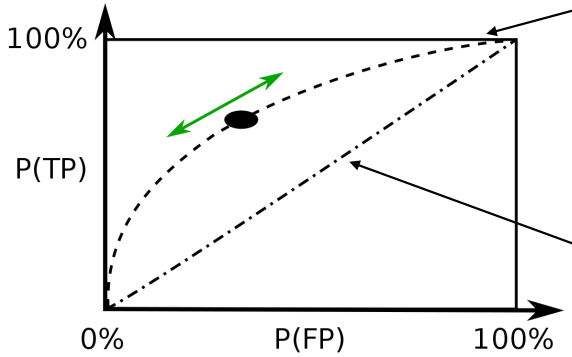


Confusion matrix for this classifier

TP	FP
FN	TN

Different thresholds indicate the positioning of the classifier separation line

No perfect separation is possible (real life data tends to be messy)

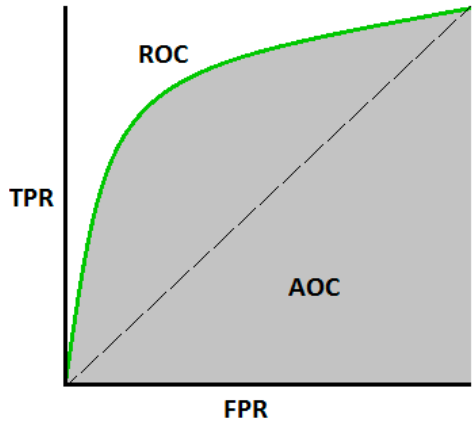


The dashed line shows the performance of a completely random classification (flipping a non-biased coin)

https://en.wikipedia.org

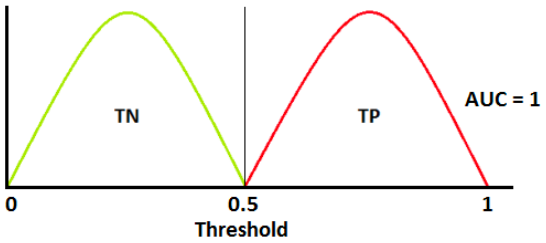
Area under the curve

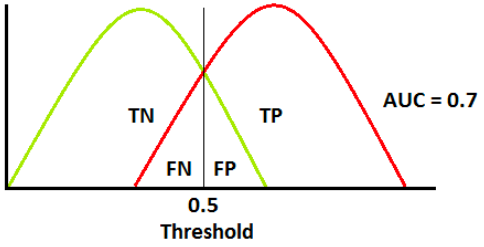
- Area under the curve (AUC)
  - Area under the ROC curve
- One of the most widely used model evaluation metrics in binary classification problems

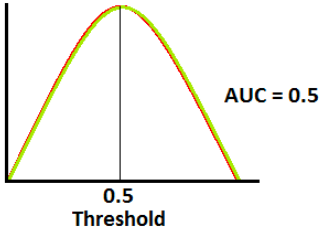


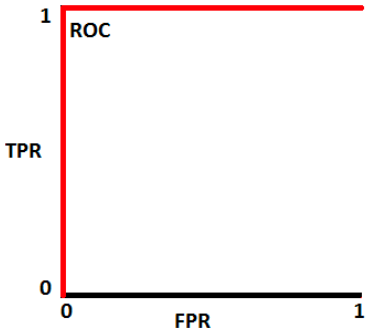
https://towardsdatascience.com/

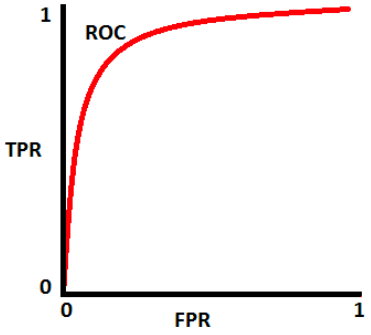
Area under the curve (cont'd)

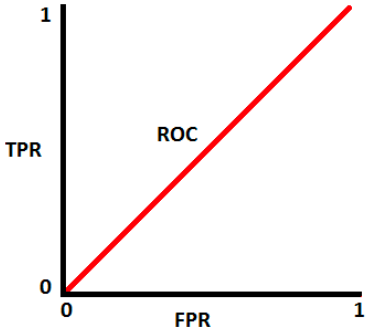












https://towardsdatascience.com/

F1 score

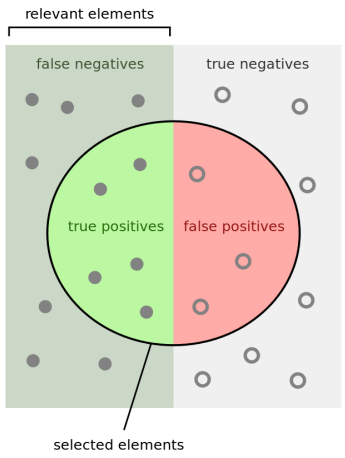
- Harmonic mean between precision and recall (sensitivity)
- Indicates how precise and robust the classifier is
- Sometimes called “F1 measure”

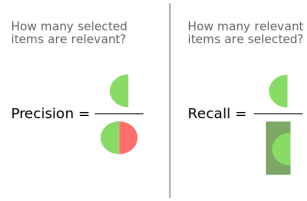
$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

https://towardsdatascience.com

$$\mathrm{F_1} = 2 \cdot \frac{\mathrm{PPV} \cdot \mathrm{TPR}}{\mathrm{PPV} + \mathrm{TPR}}$$

https://en.wikipedia.org





https://en.wikipedia.org

Logarithmic loss

- Useful for multi-class classifiers
- Metric that is based on probabilities
  - Based on negative log-likelihood function
- Lower loss means a better model
- Sometimes called “log loss”, “logistic loss”, or “cross entropy loss”

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

N - number of observations being predicted
y_i - true class for the observation *i*
$p(y_i)$ - probability observation *i* being predicted to be class *y*

https://towardsdatascience.com

Logarithmic loss (cont'd)

Multi-class log-loss:

$$F = -\frac{1}{N}\sum_{i}^{N}\sum_{j}^{M} y_{ij} \cdot Ln(p_{ij})) = \sum_{j}^{M}\left(-\frac{1}{N}\sum_{i}^{N} y_{ij} \cdot Ln(p_{ij}))\right) = \sum_{j}^{M} F_i$$

https://stats.stackexchange.com

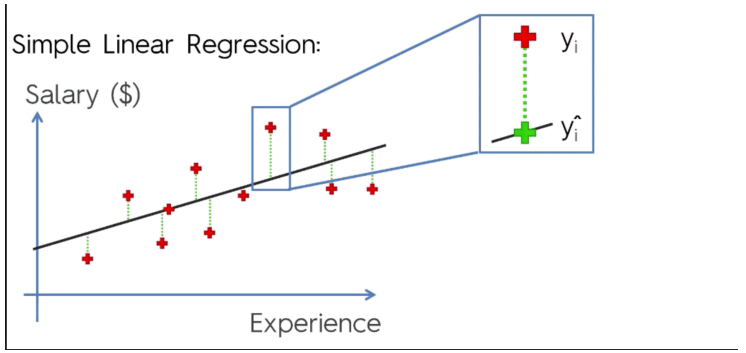
N - number of observations being predicted
M - number of classes represented in the dataset
y_{ij} - expected class assignment for class *j* for observation *i*
p_{ij} - probability observation *i* being predicted to be class *j*

Mean absolute error (MAE)

- Used in regression problems
  - Often called L1 loss
- Average absolute value distance between the predicted response and true response variable

$$MeanAbsoluteError = \frac{1}{N}\sum_{j=1}^{N}|y_j - \hat{y}_j|$$

https://towardsdatascience.com



https://medium.com
Jorge Leonel

Mean squared error (MSE)

- Used in regression problems
  - Often called L2 loss
- Average squared distance between the predicted response and true response variable
- Because the distance is squared, MSE will usually be bigger than MAE for the same model

$$MeanSquaredError = \frac{1}{N}\sum_{j=1}^{N}(y_j - \hat{y}_j)^2$$

https://towardsdatascience.com

Root mean squared error (RMSE)

- Used in regression problems
- Measures standard deviation of the errors produced by the model
- Similar to MSE but we take a root
- Makes this measure be more comparable to MAE

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

includehelp.com

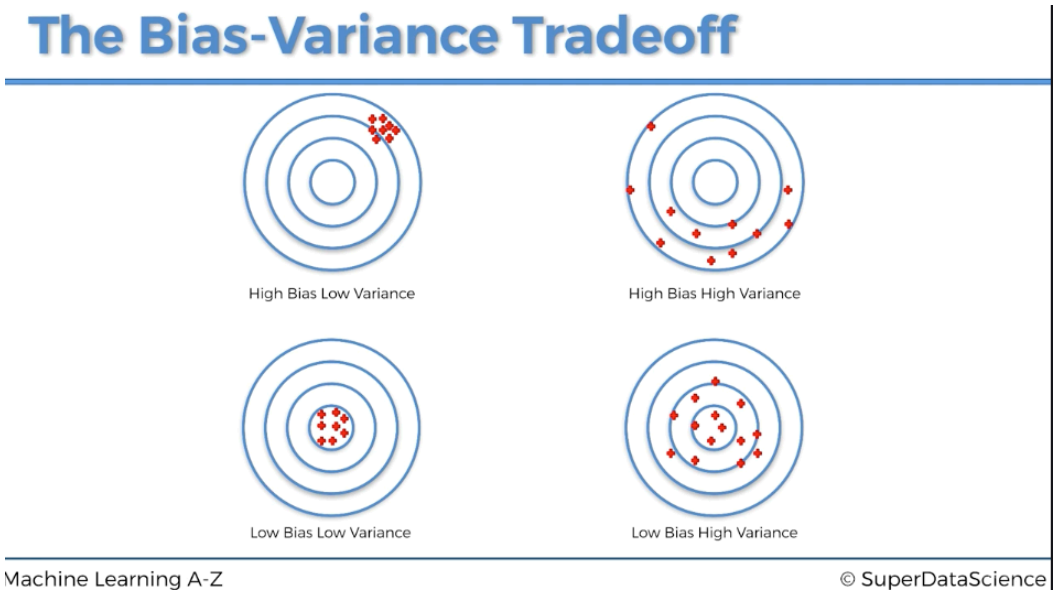
Other variations on measuring regression errors

- Mean absolute percent error (MAPE)
  - Converts MAE to percentages
- Mean percentage error (MPE)
  - Same as MAPE but does not use absolute value
  - Used for estimating if there are more positive or negative errors produced by the model
    - Since these can cancel each other out, cannot use to evaluate overall accuracy of the model

Model bias vs. model variance

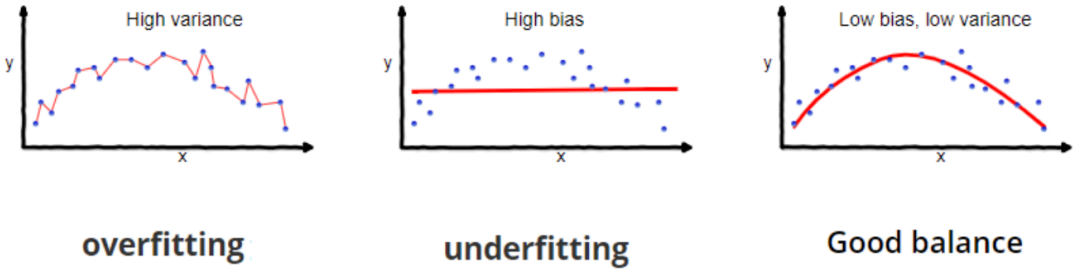
- Model bias - how well the model fits the training dataset
  - High bias model does not describe the rules and patterns of the training data
  - Detected by high error in both training and testing dataset
- Model variance - how much the predictions for a given data point might vary
  - High variance model fits the training data very well but does not generalize to unseen/test data
  - Detected by low training error but high test error
- We can use both to look for signs of model overfitting
- A good model will attempt to minimize both bias and variance

Bias-variance tradeoff



https://rpubs.com

Bias-variance tradeoff (cont'd)



overfitting **underfitting** **Good balance**

https://towardsdatascience.com

Model parameter learning

Model-based learning

Use the input data

$$\begin{bmatrix} x_{1,0} & x_{1,1} & \dots & x_{1,n} \\ x_{2,0} & x_{2,1} & \dots & x_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m,0} & x_{m,1} & \cdots & x_{m,n} \end{bmatrix} \text{ and } \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_m \end{bmatrix}$$

To learn a set of parameters

$$\begin{bmatrix} \theta_0 & \theta_1 & \cdots & \theta_n \end{bmatrix}$$

Which yield a **generalized** function

$$f(x;\theta) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$

Capable of predicting values or classes on new input data

$$f(x_i;\theta) = 39$$

$$f(x_j;\theta) = 1$$

https://www.jeremyjordan.me/hyperparameter-tuning/

Model parameters

- Internal to the model
  - E.g. weights assigned to each feature
    - Weights are learned to optimize performance
- Estimated/learned from the training data
- Saved as a part of the model
  - A model without parameters is just an algorithm that hasn't been trained yet on the data

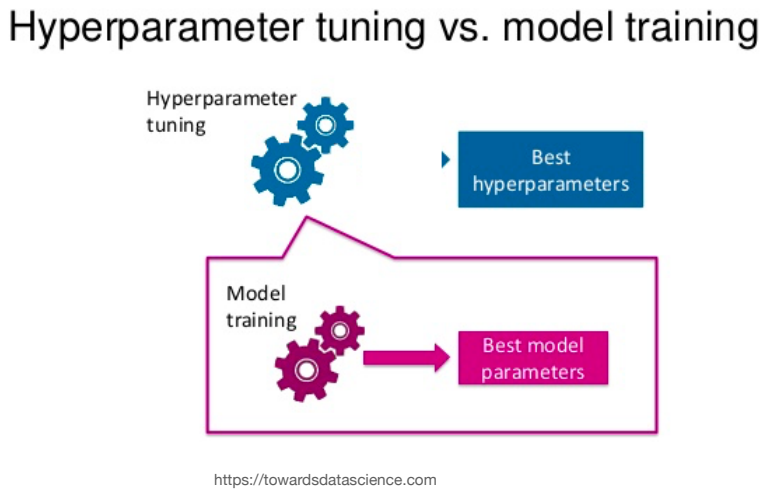
- Examples of model parameters
  - Weights learned by a neural network classifier
  - Support vectors learned by Support Vector Machine (SVM)
  - Coefficients learned by a regression model

Model hyperparameter

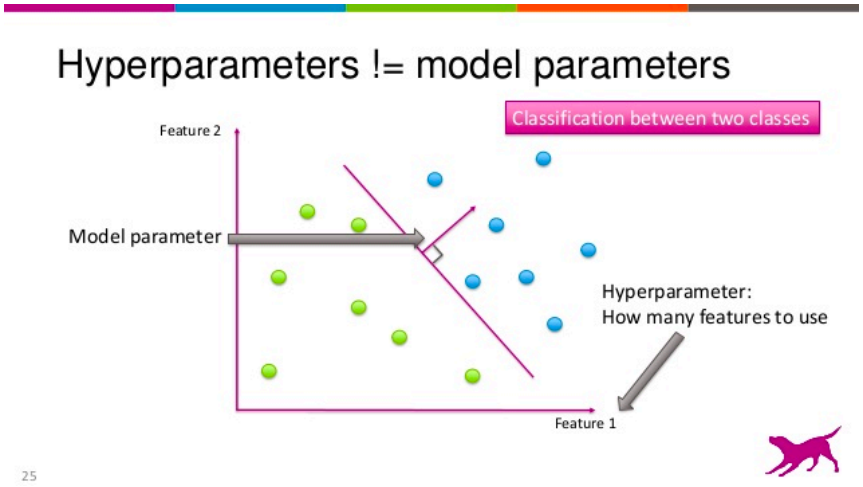
- Configuration of the model
  - Can be thought of as parameters of prior distribution (from statistics)
- Required for estimating model parameters
  - Need to be initialized prior to training the model
- Usually provided manually by domain experts or using heuristics
- Often tuned for each specific predictive task/problem

- Examples of model hyperparameters
  - Number and sizes of hidden layers in neural networks
  - Number of epochs
  - Kernel in a Support Vector Machine (SVM)
  - Number of folds in model cross-validation
  - Regression penalty
  - Learning rate of gradient descent
  - Number of clusters in k-means clustering
  - Depth of decision tree
  - Activation function

Model parameters vs. hyperparameters



Model parameters vs. hyperparameters (cont'd)



"Evaluating Machine Learning Models -- A Beginner's Guide" Alice Zheng

Model parameters vs. hyperparameters (cont’d)

Model parameters

Model parameters are learned attributes that define *individual models*.

- e.g. regression coefficients
- e.g. decision tree split locations
- They can be **learned directly** from the training data

Hyperparameters

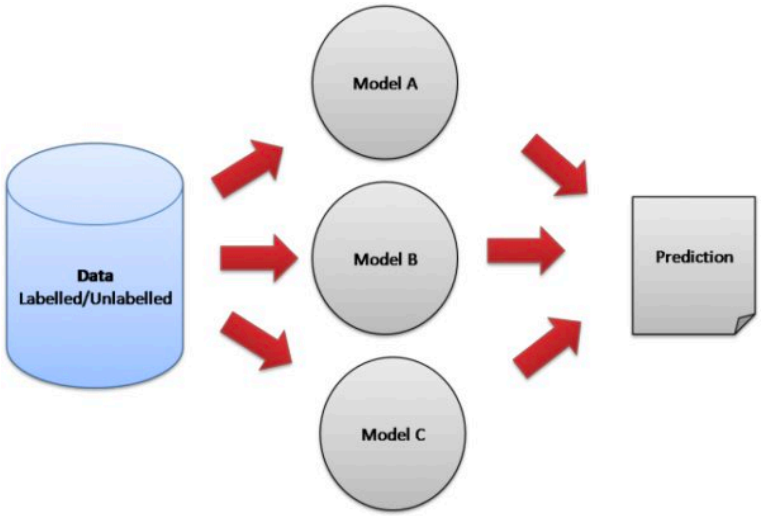
Hyperparameters express "higher-level" *structural settings* for algorithms.

- e.g. strength of the penalty used in regularized regression
- e.g. the number of trees to include in a random forest
- They are **decided** before fitting the model because they can't be learned from the data

Ensemble learning

- Ensemble learning is a technique in ML to obtain better predictive power by aggregating predictions across multiple predictors
  - Wisdom of the crowd concept
  - Big idea: one is weak, together we are strong
- Ensemble predictors consist of a finite set of alternative models representing the training set
  - Predictions are aggregated across multiple predictors
    - E.g. majority classifier, average predicted value, etc.

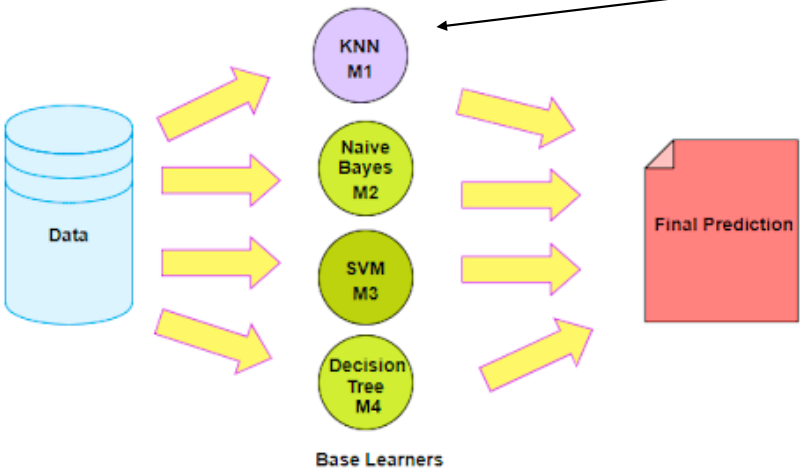
Ensemble learning (cont'd)



www.kdnuggets.com

Ensemble learning (cont'd)

These models can be produced with different algorithms, be different types of models, or be the same type of model but built with subsampling of feature space or training data, or any combination of these



Akshay Chavan 2019, www.aitimejournal.com

Representation learning

- A set of machine learning techniques that automatically learn the best representation of the data
  - Feature learning/detection
- An alternative to “feature engineering”, which requires manual labor
- Examples:
  - Supervised artificial neural networks (ANNs)
  - Supervised and unsupervised dictionary learning (learning sparse representation of the data)
  - Principal component analysis (PCA) and variations of similar data projections (ICA, SVD, etc.)
  - Autoencoders
  - Matrix factorization

Machine learning problems are optimization problems

- Much of machine learning is based on optimization theory
  - Learn solution that minimizes a loss/error or maximizes a reward
  - The process of learning is an optimization task

$$\hat{w} = arg\ max \sum_{i}^{n} L(x_i, y_i; w)$$

optimal model parameters

w - model parameters (e.g. feature weights)
x_i - input features for observation *i*
y_i - output variable for observation *i*
L - loss function (cost function, objective function)

Typical machine learning problem

- Identify and define the problem
- Collect data which will help you solve this problem
- Setup hyperparameters
- Train your model and estimate model parameters
- Evaluate your model
- Deploy your model and apply to new data
- Observe and repeat*

Stochastic gradient descent

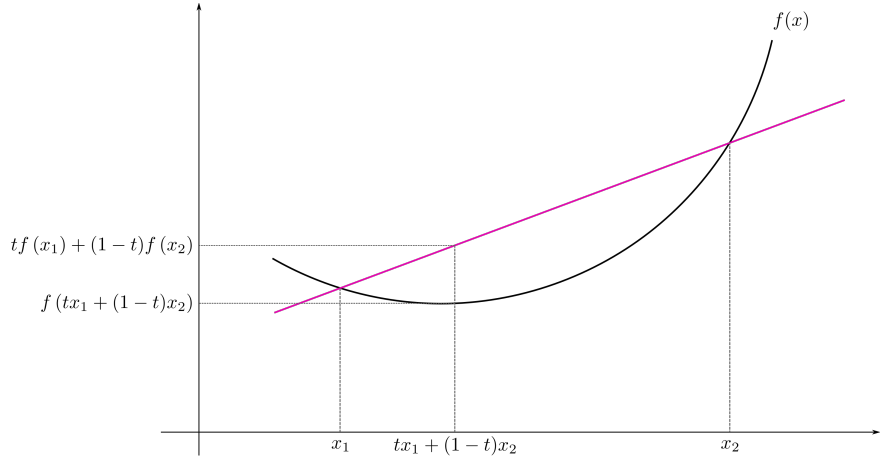
- Remember from calculus that gradient descent is a helpful way to find function minima and maxima?
  - Derivative = slope (rate of change of a function)
  - Gradient = vector pointing in the direction of the steepest ascent
- We want to find the point at which our loss function is at its minimum
- Stochastic gradient descent is used for optimizing the objective function
  - Finds the model parameters at which the value of the objective function is minimum
  - Stochastic refers to the fact that we update the weights after every samples instead of accumulating changes over the entire batch of samples (too costly)
  - Optimization technique used when parameters cannot be found analytically (combinatorial complexity of searching an infinite space)

Objective function

- Remember we discussed the loss/error/cost function for evaluating machine learning models?
- The objective of the **learning** process is to minimize cumulative loss across training samples

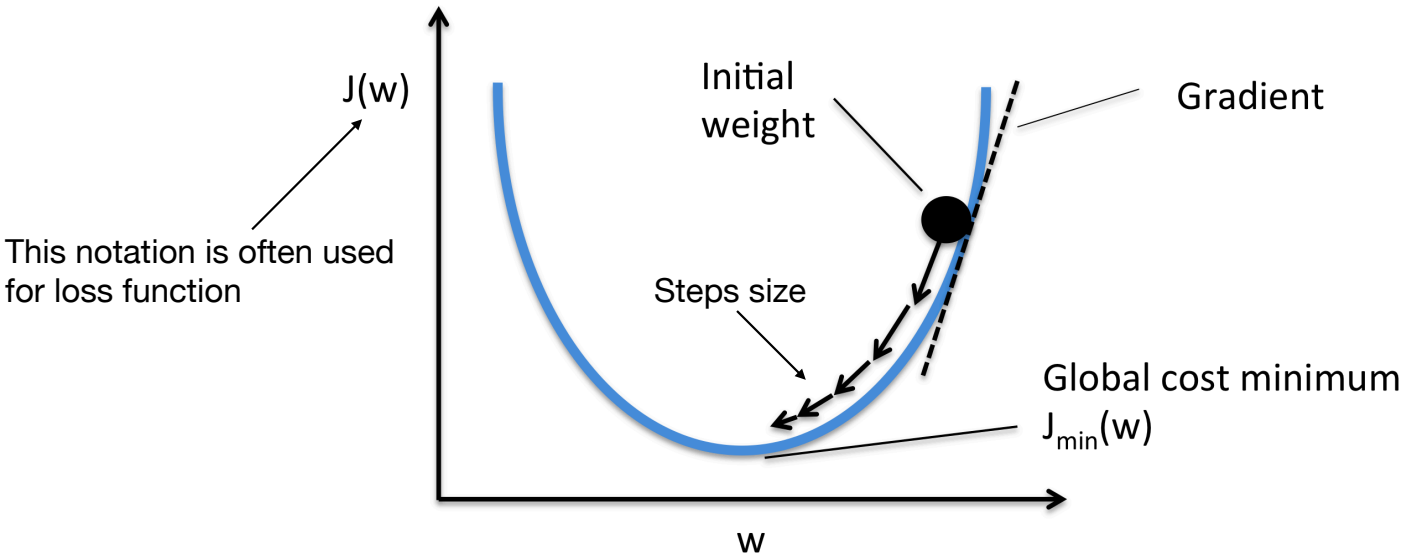
Loss function must be a strictly convex function

- “In mathematics, a real-valued function defined on an n-dimensional interval is called convex if the line segment between any two points on the graph of the function lies above the graph between the two points” - Wikipedia



https://en.wikipedia.org

Cost function optimization - intuition



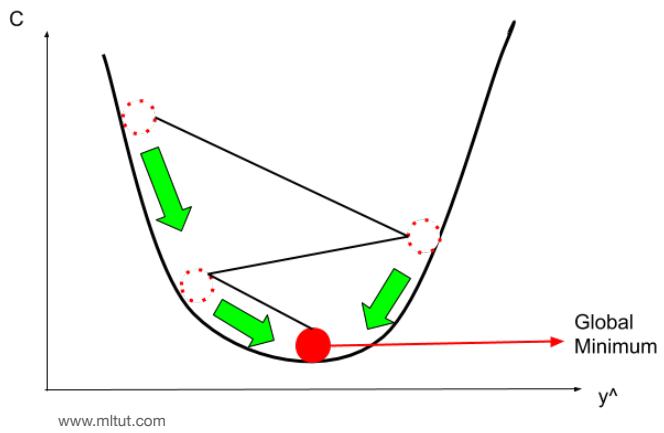
Overview of stochastic gradient descent steps

- Steps:
  - Randomly initialize your model parameters
  - Choose the gradient descent step (hyper-parameter)
  - For each sample
    - Compute loss
    - Adjust weights in the direction opposite to the gradient using the chosen step

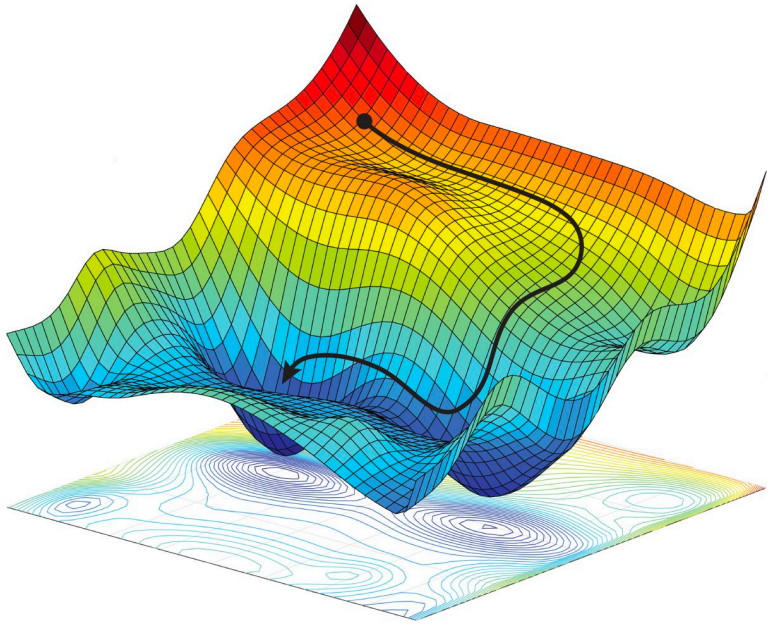
- Choose an initial vector of parameters w and step size η.
- Repeat until an approximate minimum is obtained:
  - Randomly shuffle examples in the training set.
  - For $i = 1, 2, \ldots, n$, do:
    - $w := w - \eta \nabla Q_i(w)$.

https://en.wikipedia.org

Step choice is important

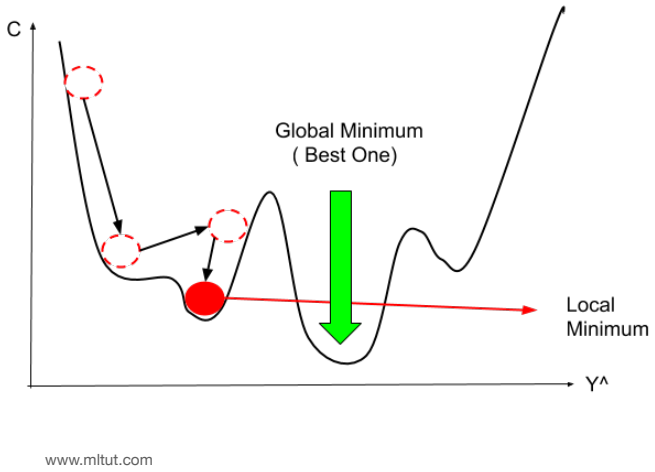


In real life loss functions are high dimensional



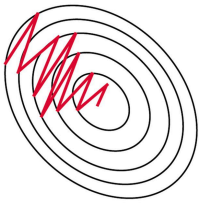
"Stochastic Gradient/Mirror Descent: Minimax Optimality and Implicit Regularization" 2019 International Conference on Learning Representations

Global vs. local minimum

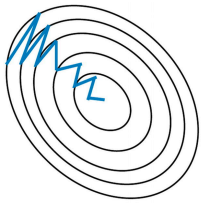


Numerous flavors of gradient descent algorithms

- Mini-batch gradient descent
- SGD with/without momentum
- …
- In practice it’s been shown that rescaling your feature space to the same scale is beneficial to gradient descent



Stochastic Gradient Descent **withhout** Momentum



Stochastic Gradient Descent **with** Momentum

eloquentarduino.github.io
ELOQUENT ARDUINO BLOG

Useful resources

- Scikit-learn provides a number of ML code examples
  - https://scikit-learn.org/stable/
  - Code examples and tutorials
- Towards data science
  - https://towardsdatascience.com/
  - Not every article is free but there are a lot of free entries there
- Kaggle
  - https://www.kaggle.com/
  - Datasets, code examples, competitions

Scikitlearn toy datasets

- https://scikit-learn.org/stable/datasets/index.html

`load_boston`(*[, return_X_y])	Load and return the boston house-prices dataset (regression).
`load_iris`(* [, return_X_y, as_frame])	Load and return the iris dataset (classification).
`load_diabetes`(* [, return_X_y, as_frame])	Load and return the diabetes dataset (regression).
`load_digits`(* [, n_class, return_X_y, as_frame])	Load and return the digits dataset (classification).
`load_linnerud`(* [, return_X_y, as_frame])	Load and return the physical excercise linnerud dataset.
`load_wine`(* [, return_X_y, as_frame])	Load and return the wine dataset (classification).
`load_breast_cancer`(* [, return_X_y, as_frame])	Load and return the breast cancer wisconsin dataset (classification).

7.2.1. Boston house prices dataset

Data Set Characteristics:

Number of Instances:	506
Number of Attributes:	13 numeric/categorical predictive. Median Value (attribute 14) is usually the target.
Attribute Information (in order):	• CRIM per capita crime rate by town • ZN proportion of residential land zoned for lots over 25,000 sq.ft. • INDUS proportion of non-retail business acres per town • CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise) • NOX nitric oxides concentration (parts per 10 million) • RM average number of rooms per dwelling • AGE proportion of owner-occupied units built prior to 1940 • DIS weighted distances to five Boston employment centres • RAD index of accessibility to radial highways • TAX full-value property-tax rate per $10,000 • PTRATIO pupil-teacher ratio by town • B 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town • LSTAT % lower status of the population • MEDV Median value of owner-occupied homes in $1000's
Missing Attribute Values:	None
Creator:	Harrison, D. and Rubinfeld, D.L.

Scikitlearn real world datasets

- https://scikit-learn.org/stable/datasets/index.html

`fetch_olivetti_faces`(* [, data_home, ...])	Load the Olivetti faces data-set from AT&T (classification).
`fetch_20newsgroups`(* [, data_home, subset, ...])	Load the filenames and data from the 20 newsgroups dataset (classification).
`fetch_20newsgroups_vectorized`(* [, subset, ...])	Load the 20 newsgroups dataset and vectorize it into token counts (classification).
`fetch_lfw_people`(* [, data_home, funneled, ...])	Load the Labeled Faces in the Wild (LFW) people dataset (classification).
`fetch_lfw_pairs`(* [, subset, data_home, ...])	Load the Labeled Faces in the Wild (LFW) pairs dataset (classification).
`fetch_covtype`(*[, data_home, ...])	Load the covertype dataset (classification).
`fetch_rcv1`(* [, data_home, subset, ...])	Load the RCV1 multilabel dataset (classification).
`fetch_kddcup99`(* [, subset, data_home, ...])	Load the kddcup99 dataset (classification).
`fetch_california_housing`(* [, data_home, ...])	Load the California housing dataset (regression).

7.3.1. The Olivetti faces dataset

This dataset contains a set of face images taken between April 1992 and April 1994 at AT&T Laboratories Cambridge. The `sklearn.datasets.fetch_olivetti_faces` function is the data fetching / caching function that downloads the data archive from AT&T.

As described on the original website:

There are ten different images of each of 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement).

Data Set Characteristics:

Classes	40
Samples total	400
Dimensionality	4096
Features	real, between 0 and 1

The image is quantized to 256 grey levels and stored as unsigned 8-bit integers; the loader will convert these to floating point values on the interval [0, 1], which are easier to work with for many algorithms.

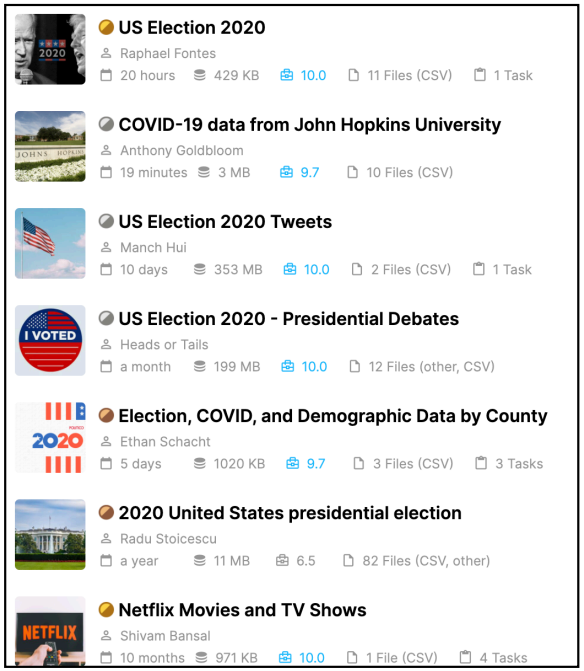
The “target” for this database is an integer from 0 to 39 indicating the identity of the person pictured; however, with only 10 examples per class, this relatively small dataset is more interesting from an unsupervised or semi-supervised perspective.

The original dataset consisted of 92 x 112, while the version available here consists of 64x64 images.

When using these images, please give credit to AT&T Laboratories Cambridge.

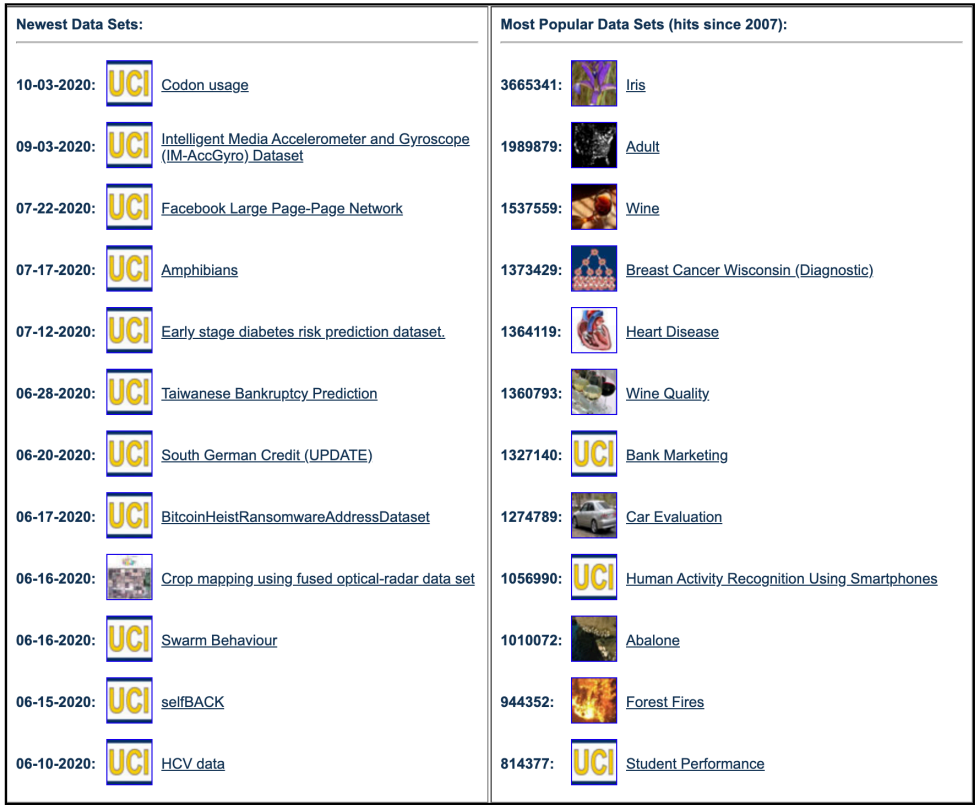
Kaggle public datasets

- https://www.kaggle.com/datasets



UC Irvine ML repository

- https://archive.ics.uci.edu/ml/index.php



Newest Data Sets:

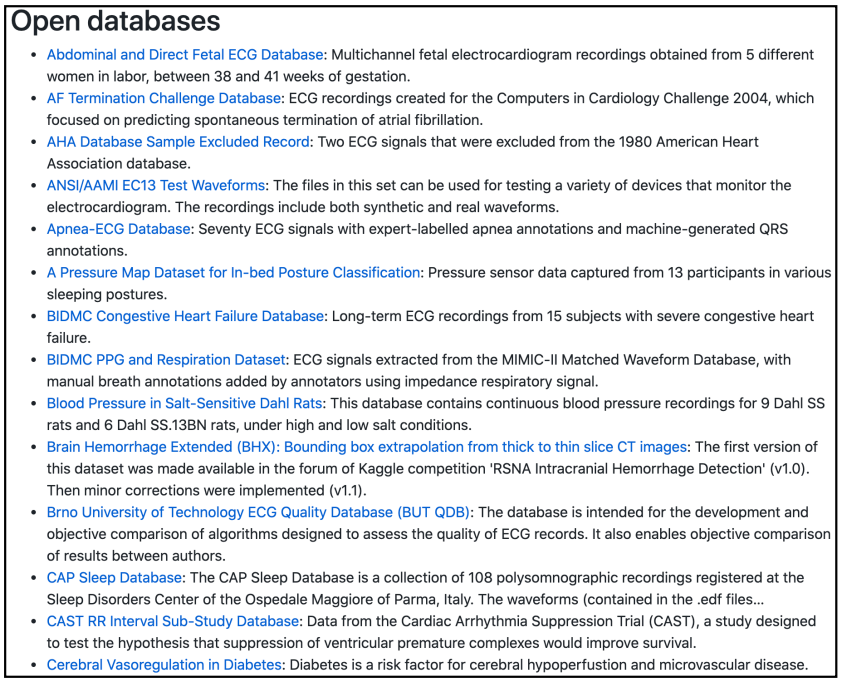
10-03-2020:	Codon usage
09-03-2020:	Intelligent Media Accelerometer and Gyroscope (IM-AccGyro) Dataset
07-22-2020:	Facebook Large Page-Page Network
07-17-2020:	Amphibians
07-12-2020:	Early stage diabetes risk prediction dataset.
06-28-2020:	Taiwanese Bankruptcy Prediction
06-20-2020:	South German Credit (UPDATE)
06-17-2020:	BitcoinHeistRansomwareAddressDataset
06-16-2020:	Crop mapping using fused optical-radar data set
06-16-2020:	Swarm Behaviour
06-15-2020:	selfBACK
06-10-2020:	HCV data

Most Popular Data Sets (hits since 2007):

3665341:	Iris
1989879:	Adult
1537559:	Wine
1373429:	Breast Cancer Wisconsin (Diagnostic)
1364119:	Heart Disease
1360793:	Wine Quality
1327140:	Bank Marketing
1274789:	Car Evaluation
1056990:	Human Activity Recognition Using Smartphones
1010072:	Abalone
944352:	Forest Fires
814377:	Student Performance

Physionet

- Biomedical research data
- https://www.physionet.org/about/database/



Open databases

- Abdominal and Direct Fetal ECG Database: Multichannel fetal electrocardiogram recordings obtained from 5 different women in labor, between 38 and 41 weeks of gestation.
- AF Termination Challenge Database: ECG recordings created for the Computers in Cardiology Challenge 2004, which focused on predicting spontaneous termination of atrial fibrillation.
- AHA Database Sample Excluded Record: Two ECG signals that were excluded from the 1980 American Heart Association database.
- ANSI/AAMI EC13 Test Waveforms: The files in this set can be used for testing a variety of devices that monitor the electrocardiogram. The recordings include both synthetic and real waveforms.
- Apnea-ECG Database: Seventy ECG signals with expert-labelled apnea annotations and machine-generated QRS annotations.
- A Pressure Map Dataset for In-bed Posture Classification: Pressure sensor data captured from 13 participants in various sleeping postures.
- BIDMC Congestive Heart Failure Database: Long-term ECG recordings from 15 subjects with severe congestive heart failure.
- BIDMC PPG and Respiration Dataset: ECG signals extracted from the MIMIC-II Matched Waveform Database, with manual breath annotations added by annotators using impedance respiratory signal.
- Blood Pressure in Salt-Sensitive Dahl Rats: This database contains continuous blood pressure recordings for 9 Dahl SS rats and 6 Dahl SS.13BN rats, under high and low salt conditions.
- Brain Hemorrhage Extended (BHX): Bounding box extrapolation from thick to thin slice CT images: The first version of this dataset was made available in the forum of Kaggle competition 'RSNA Intracranial Hemorrhage Detection' (v1.0). Then minor corrections were implemented (v1.1).
- Brno University of Technology ECG Quality Database (BUT QDB): The database is intended for the development and objective comparison of algorithms designed to assess the quality of ECG records. It also enables objective comparison of results between authors.
- CAP Sleep Database: The CAP Sleep Database is a collection of 108 polysomnographic recordings registered at the Sleep Disorders Center of the Ospedale Maggiore of Parma, Italy. The waveforms (contained in the .edf files...
- CAST RR Interval Sub-Study Database: Data from the Cardiac Arrhythmia Suppression Trial (CAST), a study designed to test the hypothesis that suppression of ventricular premature complexes would improve survival.
- Cerebral Vasoregulation in Diabetes: Diabetes is a risk factor for cerebral hypoperfustion and microvascular disease.

Many more public dataset repositories available

- Google is your friend!

Let’s look at some python examples

- Jupyter notebooks
  - Gradient descent examples
    - *Gradient_descent.ipynb*
  - Cross-validation examples
    - *Train_validation_test_Iris.ipynb*

memegenerator.net