Linear Models and ML Fundamentals II

ML4SE

Denis Litvinov

September 21, 2021

Table of Contents

Recap on Choice of Quality Metric

Problem Statement

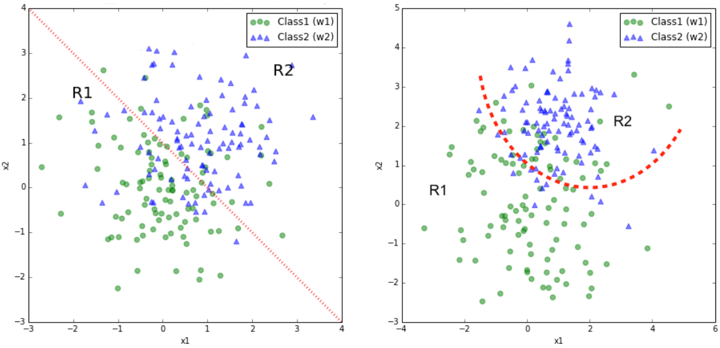
Given dataset $\{(x_i, y_i)\}_{i=1}^{N}$ of i.i.d. objects

Or, equvalently given:

$X \in R^{NxD}$ - feature matrix, where D is dimension of feature space and N - number of objects.

$Y \in \{0, 1\}^N$ - target vector

Sometimes we will use notation $Y \in \{-1, 1\}^N$



Bayesian Classifier

Bayesian classifier is the best possible classifier given we know all joint distribution $P(x, y)$ of features and labels. (Which is an unrealistic assumption.)
Bayesian risk:

$$R = \sum_{x,y} I[h(x) \neq y] P(x, y) c_y$$

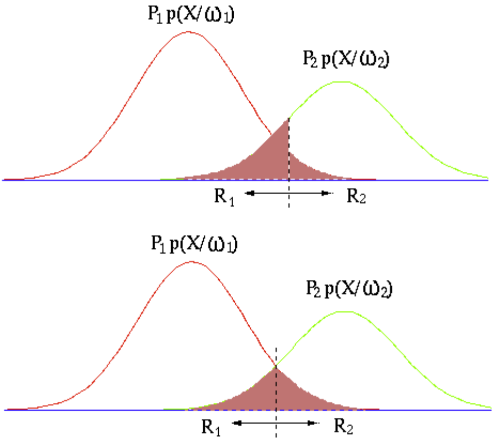
, where c_y is cost function for misclassification, e.g. $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ or $\begin{pmatrix} 0 & 0.7 \\ 0.3 & 0 \end{pmatrix}$
function $h(x)$ - decision function.
Which is minimized by

$$h(x) = \arg\max_y P(y|X)$$

$$h(x) = \arg\max_y P(X|y)P(y)c_y$$

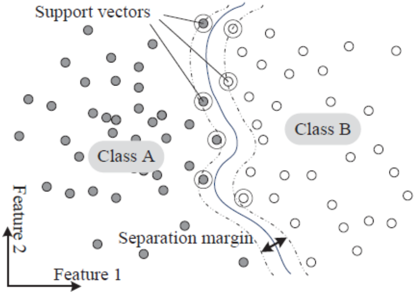
Bayesian Classifier

$\Longrightarrow$ Cost function and prior class probabilities are interchangeable!



Margin

For binary classification with $y \in \{-1, 1\}$ a variable $z = yh(x)$ is called **margin**.
Positive margin corresponds to successful classification, negative margin corresponds to error.
$|yh(x)|$ is a distance to decision boundary, which can be interpreted as confidence in classification of the object.



Logistic Regression

Suppose $\hat{y} = h(x)$ and $y \in \{0, 1\}$.
Show, that $h(x_i)$ should be $p(y = 1|x_i)$.
Probability to generate such samples from the point view of $h(x)$ is a likelihood L:

$$L = \prod_{i=1}^{N} h(x_i)^{[y_i=1]}(1 - h(x_i))^{[y_i=0]} \to \max_h$$

$$\log L = \sum_{i=1}^{N} [y_i = 1] \log h(x_i) + [y_i = 0] \log(1 - h(x_i)) \to \max_h$$

$$-\log L = -\sum_{i=1}^{N} [y_i = 1] \log h(x_i) + [y_i = 0] \log(1 - h(x_i)) \to \min_h$$

$$E[-\log L|x] = -p(y = 1|x) \log h(x) - p(y = 0|x) \log(1 - h(x)) \to \min_h$$

Logistic Regression

$$\frac{\partial E[-\log L|x]}{\partial h} = -\frac{p(y=1|x)}{h(x)} + \frac{1-p(y=1|x)}{1-h(x)} = 0$$

$$\implies h(x) = p(y=1|x) = \sigma(w^T x)$$

What are requirements for $\sigma(x)$?

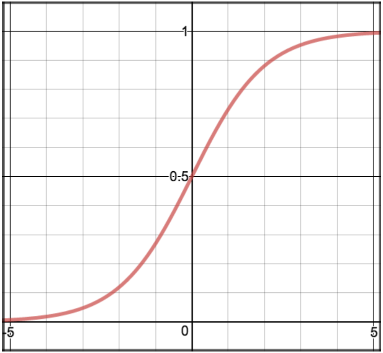
Logistic Regression

Requirements for $\sigma(x)$:

1. $\forall x \in R^D \ \sigma(x) \in [0, 1]$
2. $\sigma(0) = 0.5$
3. $\sigma(-x) = 1 - \sigma(x)$
4. σ - non-decreasing, continuous and differentiable

Sigmoid

$\sigma(z) = \frac{1}{1+e^{-z}}$



Logistic Regression

You can find 2 different formulae for logistic loss: via cross-entropy as shown above

$$Loss(y_i, p_i) = -y_i \log p(y_i = 1|x_i) - (1 - y_i) \log(1 - p(y_i = 1|x_i))$$

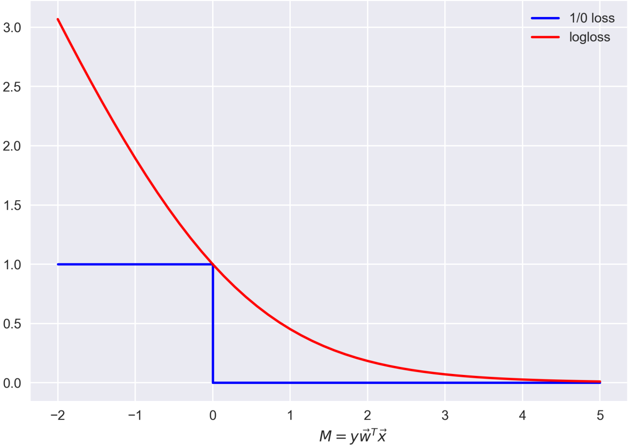
where probability of $y = 1$ class given sample x is

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

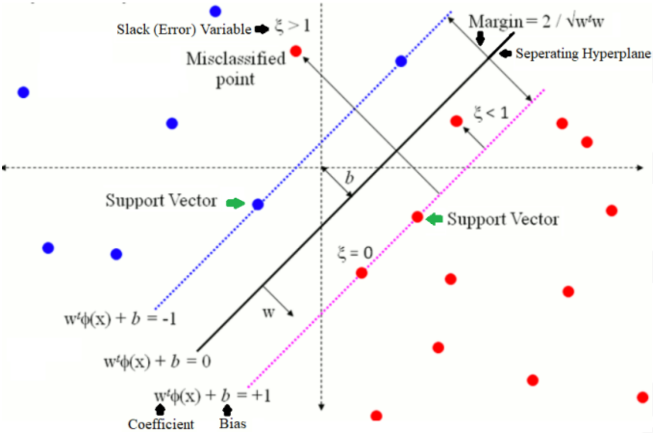
and with margins

$$Loss(y_i, x_i) = \log(1 + e^{-y_i w^T x_i})$$

Logistic Regression



Support Vector Machines



Support Vector Machines

Suppose we have some linear decision surface $h(x) = sign(w^T x)$
Then distance from point $x_0 \in R^D$ to decision surface is

$$\rho(x_0, h) = \frac{|w^T x|}{||w||_2}$$

Choose scale of w such that

$$\min_x |w^T x| = 1$$

Then distance from decision surface to the nearest object is

$$\min_x \frac{|w^T x|}{||w||_2} = \frac{1}{||w||_2} \min_x |w^T x| = \frac{1}{||w||_2}$$

For linear separable case we have optimization problem:

$$\begin{cases} \frac{1}{2}||w||_2^2 \rightarrow \min_w \\ y_i w^T x_i \ \geq 1 \end{cases}$$

Support Vector Machines

For linear inseparable case we introduce corrections for each object ξ_i:

$$\begin{cases} \frac{1}{2}||w||_2^2 + C\sum_{i=1}^{N}\xi_i \to \min_{w,\xi_i} \\ y_i w^T x_i \ \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases}$$

OR:

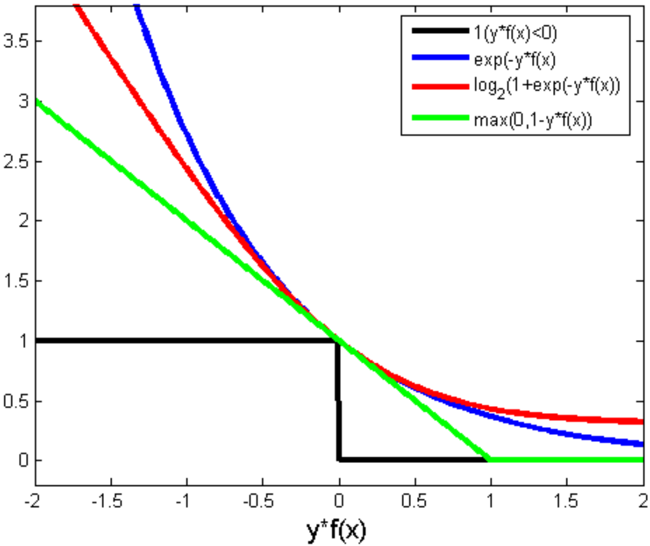
$$\xi_i = max(0, 1 - y_i w^T x_i)$$

Then,

$$Loss = \frac{1}{2C}||w||_2^2 + \sum_{i=1}^{N} max(0, 1 - y_i w^T x_i) \to \min_{w}$$

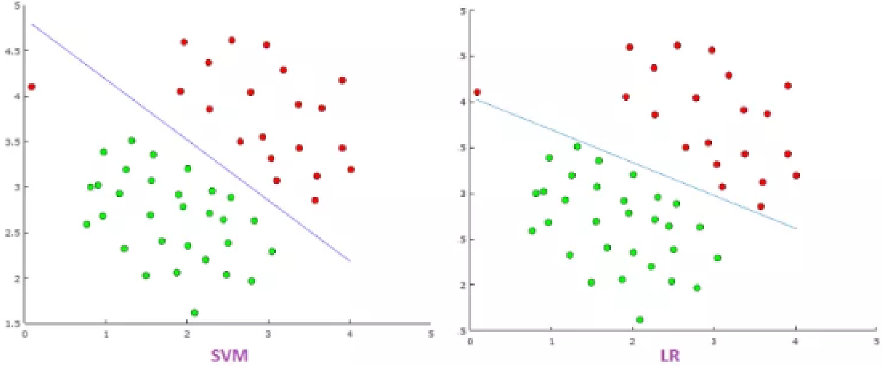
Unlike logistic regression, weight norm penalty already build in the model.

Losses


1(y*f(x)<0)
exp(-y*f(x)
$\log_2$(1+exp(-y*f(x))
max(0,1-y*f(x))
3.5
3
2.5
2
1.5
1
0.5
0
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
y*f(x)

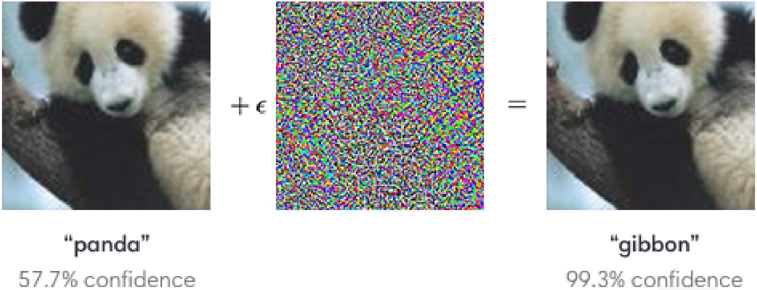
Robustness

Model is called robust if its performance do not change significantly for new samples drawn from the same distribution $P(x, y)$.
In other way, model robustness depends on how the model handle the outliers.



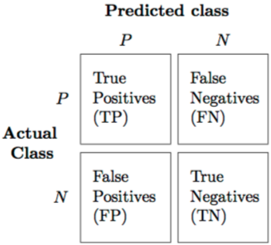
Adversarial Examples

Adversarial example is an object from original $P(x, y)$ with small perturbation in x such that it makes the model to give false predictions.



F1 score

Accuracy $acc(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}[\hat{y}_i = y_i]$



		Predicted class	
		P	*N*
Actual Class	*P*	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

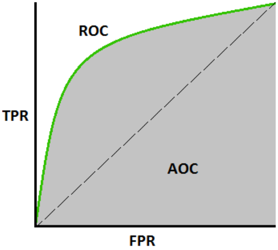
$$precision = \frac{TP}{TP + FP}$$

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

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

AUC

$FPR = \frac{FP}{FP+TN}$ false positive rate
$TPR = \frac{TP}{TP+FN}$ true positive rate
AUC = area under the curve ROC
$ROC(t) = (TPR(t), FPR(t))$ is parametrized by threshold t on the probability $p(y=1|x)$



How does class imbalance affect quality metrics given above?

Empirical Loss Minimization

In general, we want to optimize Expected Risk:

$$R = E[Loss(x,y)] = \int_{-\infty}^{\infty} Loss(x,y)dP(x,y) = Pr_{(x_i,y_i)\sim D}[Loss(x_i,y_i)]$$

But since we don't now the joint distribution $P(x,y)$, we can only deal with Empirical Risk (Loss functional):

$$\hat{R} = \sum_{i=1}^{N} Loss(x_i,y_i)$$

Generalization error

How well does $\hat{R}$ approximates R?
Using Hoeffding's inequality it can be shown, that given N random examples, and $\forall\delta > 0$, with probability $Pr >= 1 - \delta$, the following upper bound holds on the generalization error of h:

$$R \leq \hat{R} + \sqrt{\frac{\ln(1/\delta)}{2N}}$$

For finite hypothesis space H under the same conditions:

$$R \leq \hat{R} + \sqrt{\frac{\ln|H| + \ln(1/\delta)}{2N}}$$

where $\ln|H|$ serves as a measure of model complexity.

Rademacher Complexity

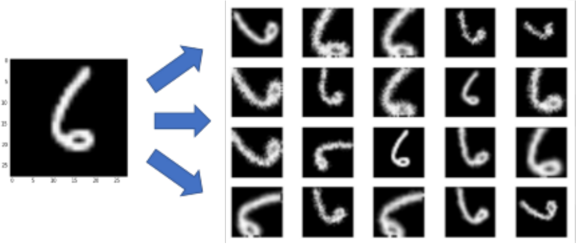
How to approximately estimate model complexity?
Given a dataset $\{(x_i)\}_{i=1}^N$,
Let $\sigma_i \sim random(\{-1, 1\})$ be random labels for each x_i.
Consider binary classification task. We expect a model to have high complexity if it can fit well any random label assignment on the dataset.
Rademacher Complexity can be estimated by expected average margin

$$E_\sigma[\max_{h \in H} \frac{1}{N} \sum_{i=1}^{N} \sigma_i h(x_i)]$$

We also say, that model with high complexity tends to memorize training samples.

Data Augmentation

Generalization error depends on number of samples *N* in the dataset.
How to increase *N*?
Apply some transformation to original data which is invariant to some desirable properties.



One vs Rest

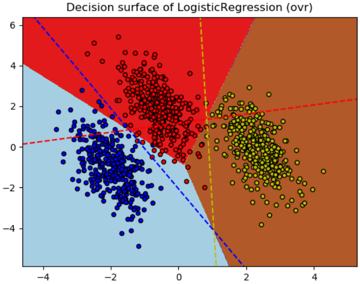
Idea: build multiclass classifier from several binary classifiers

Train K binary classifiers.

$$\hat{y} = \arg\max_k h_k(x)$$

Note:

1. h_k is unbalanced even if initial problem was balanced
2. scale of the confidence values may differ between the binary classifiers b_k

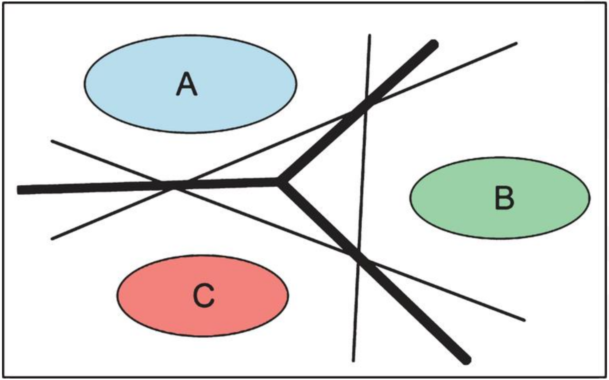


One vs One

Idea: build multiclass classifier from several binary classifiers
Train $K(K-1)/2$ binary classifiers.

$$\hat{y} = \arg\max_k \sum_{i \neq k} h_{ik}(x)$$

Note: One vs one is less prone to imbalance in dataset



Multinomial

Cross-entropy loss:

$$Loss(y_i, p_i) = -\sum_{k=1}^{K} y_{ik} \log p(y_{ik} = 1 | x_{ik})$$

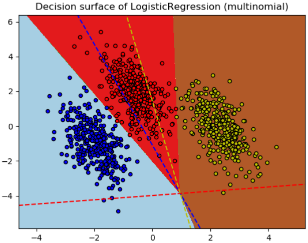
where k is a number of classes

$y_{ik} = I[y_i = k]$

Probability if x_i has class k (in vector form):

$$p(y_i = k | x_i) = softmax(x_i^T W)_k$$

where $W \in R^{DxK}$



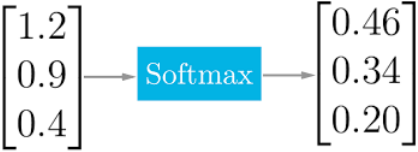
Softmax

Requirements:

1. non-negative
2. sums to 1 (is a probability)
3. monotonic increasing

for $z \in R^K$

$$softmax(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$


$\begin{bmatrix}1.2\\0.9\\0.4\end{bmatrix}$
Softmax
$\begin{bmatrix}0.46\\0.34\\0.20\end{bmatrix}$

Quality Metrics

There is no direct quality metric, it is assembled from metrics for binary classification problems.

$$micro\_precision = \frac{\sum_k TP_k}{\sum_k TP_k + \sum_k FP_k}$$

$$micro\_recall = \frac{\sum_k TP_k}{\sum_k TP_k + \sum_k FN_k}$$

$$micro\_f1 = \frac{2 * micro\_precision * micro\_recall}{micro\_precision + micro\_recall}$$

weight in proportion of class size:

$$weighted\_precision = \frac{1}{K} \sum_k \frac{|K|}{N} \frac{TP_k}{TP_k + FP_k}$$

$$weighted\_recall = \frac{1}{K} \sum_k \frac{|K|}{N} \frac{TP_k}{TP_k + FN_k}$$

$$weighted\_f1 = \frac{2 * weighted\_precision * weighted\_recall}{weighted\_precision + weighted\_recall}$$

Quality Metrics

Macro averaging is insensitive to class imbalance

$$macro\_precision = \frac{1}{K}\sum_k \frac{TP_k}{TP_k + FP_k}$$

$$macro\_recall = \frac{1}{K}\sum_k \frac{TP_k}{TP_k + FN_k}$$

$$macro\_f1 = \frac{2 * macro\_precision * macro\_recall}{macro\_precision + macro\_recall}$$

Validation

Till now:

1. $\{x_i, y_i\}_{i=1}^N$ is sampled from $P(x, y)$
2. Split into to non-overlapping subsets (train and test)
3. Model $h(x_i; w, \theta)$ is described by its trainable weights w and non-trainable hyperparams θ
4. choose some hyperparam value $\theta = \theta_0$ and train model
$$\sum_{i \in train} Loss(h(x_i; w, \theta_0), y_i) \rightarrow \min_w$$
5. test model performance on test dataset
$$\hat{R}(\theta_0) = \sum_{i \in test} Loss(h(x_i; w, \theta_0), y_i)$$
6. we expect that it is a good approximation of true generalization error
$$R(\theta_0) = E_{(x,y) \sim P(x,y)}[Loss(h(x_i; w, \theta_0), y_i)]$$
7. How to choose hyperparam θ?

Validation

Usually want to optimize hyperparams by testing several values of θ on the test set and choosing the best one.
But the performance on the test set $\hat{R}(\theta)$(empirical risk) is a random variable, which can depend on the particular train
test split! Here validation comes into play.
We can say that $\hat{R}(\theta)$ is a point estimate of expected risk $R(\theta)$, which has its bias and variance.
Different validation schemes try to minimize bias or variance or both.

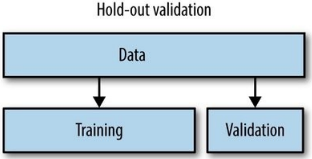
Hold Out

Given dataset of m objects, create m experiments:

1. create split train:val, usually in proportion 70:30, 80:20 or 90:10
2. fit model weights on train subset
3. evaluate performance on the val subset

Properties:

- High bias and low variance of estimate
- O(1) complexity
- Usually done when we have large dataset and or very heavy model



Cross Validation

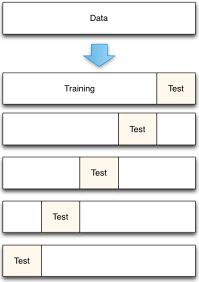
k = number of folds folds = non-intersecting subsets of the dataset

Make k experiments:

1. create split for $k-1$:1
2. train on $(k-1)$ folds and evaluate performance on the k-th fold
3. change split
4. Average scores over all experiments

Properties:

- Moderate bias and variance of estimate
- O(k) complexity

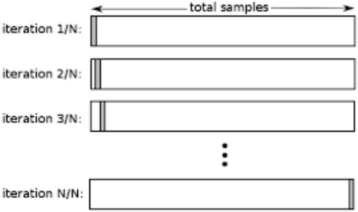


Leave One Out

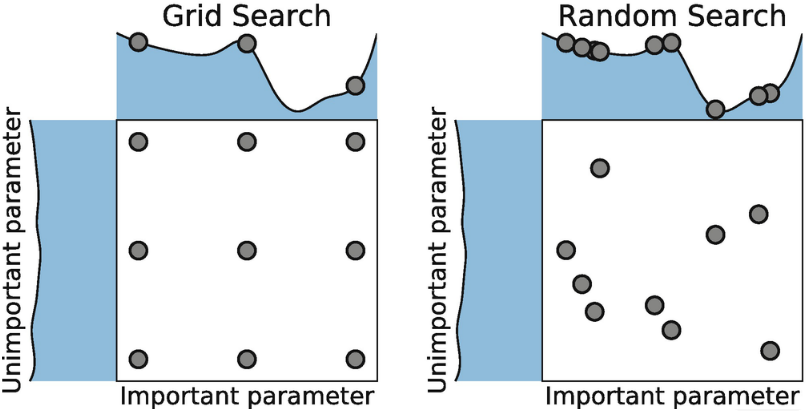
Cross validation with $k = N$

Properties:

- Low bias and High Variance of estimate
- O(N) complexity
- Usually done when we have very small dataset
- There are performance metrics (e.g. AUC) that cannot be computed just on one sample.



Hyperparam Search

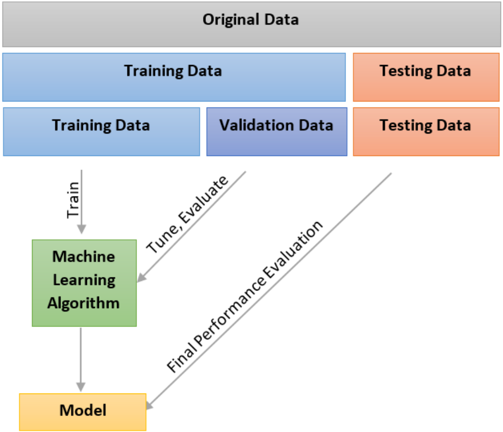


Common Pipeline

1. Split dataset for train, test parts
2. Choose validation scheme on training data
3. Train model on the train dataset without regularization, try to achieve zero training loss
4. Add regularization, tune hyperparams on validation
5. Evaluate final model performance on test dataset. Choose between different model families.

In practice we usually use chosen quality metric instead of loss function for choosing hyperparams and final testing.

Common Pipeline


Original Data
Training Data
Testing Data
Training Data
Validation Data
Testing Data
Train
Tune, Evaluate
Final Performance Evaluation
Machine Learning Algorithm
Model