Machine learning fundamentals

PytzMLS2018@IdabaX: CIVE UDOM Tanzania.

Anthony Faustine

PhD Fellow
(IDLab research group-Ghent University)

04*th* April 2018

GHENT UNIVERSITY

IDLab INTERNET & DATA LAB

Learning goal

- Understand the basics of Machine learning and its applications.
- Learn how to formulate learning problem.
- Explore different challenges of machine learning models.
- Learn best practise for designing and evaluationg machine learning models.

Outline

What is ML ?

Machine learning (ML): the science (and art) of programming computers so they can learn from data.

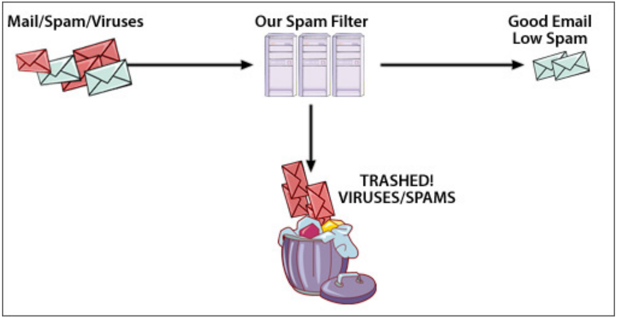
Learn from data

- Automatically detect patterns in data and
- Build models that explain the world
- Use the uncovered pattern to understand what is happening (inference) and to predict what will happen(prediction).

This gives computers the ability to learn without being explicitly programmed.

Why ML?

Consider how you would write a spam filter using traditional programming techniques.



Why ML?

- Hard problems in high dimensions, like many modern CV or NLP problems require complex models ⇒ difcult to program the correct behavior by hand.
- Machines can discover hidden, non-obvious patterns.
- A system might need to adapt to a changing environment.
- A learning algorithm might be able to perform better than its human programmers.

ML applications

As an exciting and fast-moving field ML has many applications.

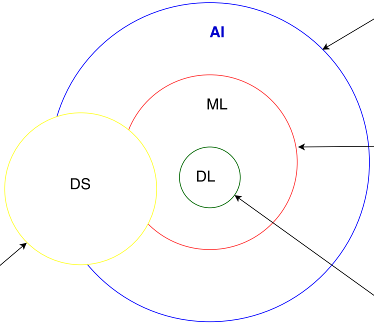
- Computer vision: Object Classification in Photograph, image captioning.
- Speech recognition, Automatic Machine Translation,
- Communication systems
- Robots learning complex behaviors:
- Recommendations services (predict interests (Facebook), predict other books you might like (Amazon), .
- Medical diagnosis.
- Bank(Fraud detection and prevention).
- Computaional biology (tumor detection, drug discovery and DNA sequencing).
- Search engines (Google).
- Anamloly and events detection (IoT, factory predictive maintance).

ML types

Machine learning is usually divide into three major types:

1. **Supervised Learning**
   - Learn a model from a given set of input-output pairs, in order to predict the output of new inputs.
   - Further grouped into Regression and classification problems.
2. **Unsupervised Learning**
   - Discover patterns and learn the structure of unlabelled data.
   - Example Distribution modeling and Clustering.
3. **Reinforcement Learning**
   - Learn what actions to take in a given situation, based on rewards and penalties. More details on RL
   - Example consider teaching a dog a new trick: you cannot tell it what to do, but you can reward/punish it.

AI vs ML vs Deep learning Vs Data science



Artificial Intelligence

- Enable computer to do think
- Old school AI (Simple rule systems) not used.

Machine learning

- Subset of AI that enable computer to learn from data.

Data Science

- Understating or making sense of data
- Visualisation, Data ingineering, Data products etc

Deep learning

- Subset of ML that use a cascade of multiple layers of nonlinear processing for pattern recognition and for feature learning.

Outline

Formulate a learning problem

To formulate ML problem mathematically, you need first to define Model (Hypothesis) and Loss function:

Model (Hypothesis)

Given a set of labeled training examples $\{\mathbf{x}_{1:N}, \mathbf{y}_{1:N}\}$:

- A model is a set of allowable functions $f_\theta(\mathbf{x};\theta)$ that compute predictions $\hat{\mathbf{y}}$ from the inputs $\mathbf{x}$ $\Rightarrow$ map inputs $\mathbf{x}$ to outputs $\mathbf{y}$ parameterized by θ.

Loss function

Given a set of labeled data $\{\mathbf{x}_{1:D}, \mathbf{y}_{1:M}\}$ and the hypothesis $f(\mathbf{x};\theta)$

- Loss function $\mathcal{L}_\theta(f(\mathbf{x};\theta), \mathbf{y})$ defines how well the model $f(\mathbf{x};\theta)$ fit the data $\Rightarrow$ howfar off the prediction $\hat{y}$ is from the output y.

Formulate a learning problem

To formulate ML problem mathematically, you need first to define Model (Hypothesis) and Loss function:

Model (Hypothesis)

Given a set of labeled training examples $\{\mathbf{x}_{1:N}, \mathbf{y}_{1:N}\}$:

- A model is a set of allowable functions $f_\theta(\mathbf{x}; \theta)$ that compute predictions $\hat{\mathbf{y}}$ from the inputs $\mathbf{x}$ $\Rightarrow$ map inputs $\mathbf{x}$ to outputs $\mathbf{y}$ parameterized by θ.

Loss function

Given a set of labeled data $\{\mathbf{x}_{1:D}, \mathbf{y}_{1:M}\}$ and the hypothesis $f(\mathbf{x}; \theta)$

- Loss function $\mathcal{L}_\theta(f(\mathbf{x}; \theta), \mathbf{y})$ defines how well the model $f(\mathbf{x}; \theta)$ fit the data $\Rightarrow$ howfar off the prediction $\hat{y}$ is from the output y.

Formulate a learning problem: Optimisation problem

The loss, averaged over all the training examples is called cost function given by:

$$J_\theta = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_\theta(\hat{y}^{(i)}, y^{(i)})$$

Optimisation Problem

After model and loss function definition we need to solve an optimisation problem.

- Find the model parameters θ that best fit the data $\Rightarrow$ Empirical Risk Minimization.

$$\arg\min_\theta \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_\theta(\hat{y}^{(i)}, y^{(i)}) \tag{1}$$

- Objective: minimize a cost function J_θ with respect to the model parameters θ

Formulate a learning problem: Optimisation problem

The loss, averaged over all the training examples is called cost function given by:

$$J_\theta = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_\theta(\hat{y}^{(i)}, y^{(i)})$$

Optimisation Problem

After model and loss function definition we need to solve an optimisation problem.

- Find the model parameters θ that best fit the data $\Rightarrow$ Empirical Risk Minimization.

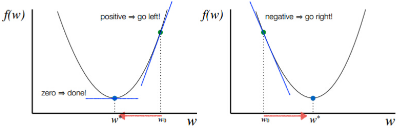
$$\arg\min_\theta \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_\theta(\hat{y}^{(i)}, y^{(i)}) \tag{1}$$

- Objective: minimize a cost function J_θ with respect to the model parameters θ

Formulate a learning problem: Gradient Descent

Gradient descent: procedure to minimize a loss function.

- compute gradient
- take step in opposite direction



1. Initilize parameter θ,
2. Loop until converge,
3. Compute gradient: $\frac{\partial J_\theta}{\partial \theta}$
4. Update parameters: $\theta^{t+1} = \theta^t - \alpha \frac{\partial J_\theta}{\partial \theta}$
5. Return parameter θ

Formulate a learning problem: Gradient Descent

α is the learning rate $\rightarrow$ determine size of step we take to reach local minimum.

Issues:

- What is the appropriate value of α?
- Avoid non-global minimum

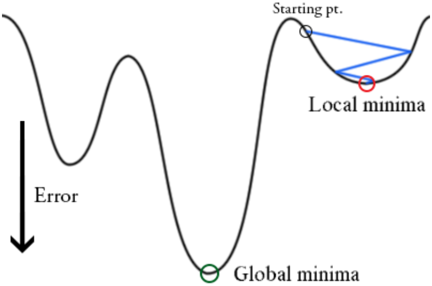


Figure 1: Gradient Descent.

Formulate a learning problem: Linear regression

Linear regression: predict a scalar-valued target, such as the price of stock.

1. weather forecasting.
2. house pricing prediction.
3. student performance prediction.
4. ....
5. ....

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

Figure 2: dataset.

Formulate a learning problem: Linear regression

In linear regression, the model consists of linear functions given by:

$$f(\mathbf{x};\theta) = \sum_j w_j x_j + b$$

where w is the weights, and b is the bias.

The loss function is given by:

$$\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$$

The cost function:

$$\begin{aligned} J_\theta &= \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) \\ &= \frac{1}{2N}\sum_{i=1}^{N}\left(\sum_j w_j x_j^{(i)} + b - y^{(i)}\right) \end{aligned}$$

In vectorized form:

$$J_\theta = \frac{1}{2N}\|\hat{\mathbf{y}} - \mathbf{y}\|^2 = \frac{1}{2N}(\hat{\mathbf{y}} - \mathbf{y})^T(\hat{\mathbf{y}} - \mathbf{y}) \quad \text{where} \quad \hat{\mathbf{y}} = \mathbf{w}^{\mathbf{T}}\mathbf{x}$$

Formulate a learning problem: Linear regression

Use gradient descent to solve the minimum cost function J_θ

$$\theta^{t+1} = \theta^t - \alpha \frac{\partial J_\theta}{\partial \theta}$$

For parameter $\mathbf{w}$ and $\mathbf{b}$:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \alpha \frac{\partial J_\theta}{\partial \mathbf{w}}$$

$$\mathbf{b}^{t+1} = \mathbf{b}^t - \alpha \frac{\partial J_\theta}{\partial \mathbf{b}}$$

where:

$$\frac{\partial J_\theta}{\partial \mathbf{w}} = \frac{1}{N}\mathbf{x}^{\mathrm{T}}(\hat{\mathbf{y}} - \mathbf{y})$$

$$\frac{\partial J_\theta}{\partial \mathbf{b}} = \frac{1}{N}(\hat{\mathbf{y}} - \mathbf{y})$$

```
import torch
import torch.nn as nn
import torch.optim.SGD as SGD

feature_dim = 1
target_dim  = 1
model=nn.Linear(feature_dim,target_dim)
loss_fn = nn.MSELoss()
optimizer=SGD(model.parameters(), lr=0.1)
for epoch in range(100):
    optimizer.zero_grad()
    y_pred=model(x)
    cost=loss_fn(y_pred,y)
    cost.backward()
    optimizer.step()
```

Formulate a learning problem: Linear regression

Use gradient descent to solve the minimum cost function J_θ

$$\theta^{t+1} = \theta^t - \alpha \frac{\partial J_\theta}{\partial \theta}$$

For parameter $\mathbf{w}$ and $\mathbf{b}$:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \alpha \frac{\partial J_\theta}{\partial \mathbf{w}}$$

$$\mathbf{b}^{t+1} = \mathbf{b}^t - \alpha \frac{\partial J_\theta}{\partial \mathbf{b}}$$

where:

$$\frac{\partial J_\theta}{\partial \mathbf{w}} = \frac{1}{N}\mathbf{x}^{\mathbf{T}}(\hat{\mathbf{y}} - \mathbf{y})$$

$$\frac{\partial J_\theta}{\partial \mathbf{b}} = \frac{1}{N}(\hat{\mathbf{y}} - \mathbf{y})$$

```
import torch
import torch.nn as nn
import torch.optim.SGD as SGD

feature_dim = 1
target_dim  = 1
model=nn.Linear(feature_dim,target_dim)
loss_fn = nn.MSELoss()
optimizer=SGD(model.parameters(), lr=0.1)
for epoch in range(100):
    optimizer.zero_grad()
    y_pred=model(x)
    cost=loss_fn(y_pred,y)
    cost.backward()
    optimizer.step()
```

Formulate a learning problem: Linear regression

Use gradient descent to solve the minimum cost function J_θ

$$\theta^{t+1} = \theta^t - \alpha \frac{\partial J_\theta}{\partial \theta}$$

For parameter $\mathbf{w}$ and $\mathbf{b}$:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \alpha \frac{\partial J_\theta}{\partial \mathbf{w}}$$

$$\mathbf{b}^{t+1} = \mathbf{b}^t - \alpha \frac{\partial J_\theta}{\partial \mathbf{b}}$$

where:

$$\frac{\partial J_\theta}{\partial \mathbf{w}} = \frac{1}{N} \mathbf{x}^{\mathbf{T}} (\hat{\mathbf{y}} - \mathbf{y})$$

$$\frac{\partial J_\theta}{\partial \mathbf{b}} = \frac{1}{N} (\hat{\mathbf{y}} - \mathbf{y})$$

```python
import torch
import torch.nn as nn
import torch.optim.SGD as SGD

feature_dim = 1
target_dim  = 1
model=nn.Linear(feature_dim,target_dim)
loss_fn = nn.MSELoss()
optimizer=SGD(model.parameters(), lr=0.1)
for epoch in range(100):
    optimizer.zero_grad()
    y_pred=model(x)
    cost=loss_fn(y_pred,y)
    cost.backward()
    optimizer.step()
```

Formulate a learning problem: Classification

Goal is to learn a mapping from inputs x to target y such that $y \in \{1 \dots k\}$ where k is the number of classes.

- If $k = 2$, this is called binary classification.
- If $k > 2$, this is called multiclass classification.
- If each instance of x is associated with more than one label to each instance, this is called multilabel classification

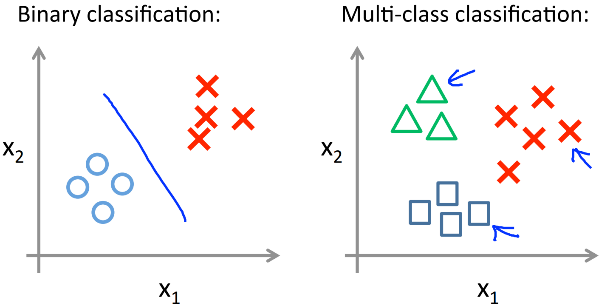


Figure 3: dataset.

Classification: Logistic regression

Goal is to predict the binary target class $y \in \{0, 1\}$.

Model is given by:

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

where

$$z = \mathbf{w}^{\mathbf{T}}\mathbf{x} + \mathbf{b}$$

```
import torch
from torch.nn as nn
import torch.nn.functional as F

z=nn.Linear(feature_dim, 1)
model = F.sigmoid(z)
```

This function squashes the predictions to be between 0 and 1 such that:

$$p(y = 1 \mid x, \theta) = \sigma(z)$$

and

$$p(y = 0 \mid \mathbf{x}, \theta) = 1 - \sigma(\mathbf{z})$$

Classification: Logistic regression

Loss function: it is called crossentropy and defined as:

$$\mathcal{L}_{CE}(\hat{y}, y) = \begin{cases} -\log \hat{y} & \text{if } y = 1 \\ -\log(1-\hat{y}) & \text{if } y = 0 \end{cases}$$

The crossentropy can be written in other form as:

$$\mathcal{L}_{CE}(\hat{y}, y) = -y \log \hat{y} - (1-y)\log(1-\hat{y})$$

The cost function J_θ with respect to the model parameters θ is thus:

$$J_\theta = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_{CE}(\hat{y}, y)$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left(-y^{(i)}\log \hat{y}^{(i)} - (1-y^{(i)})\log(1-\hat{y}^{(i)})\right)$$

```
loss_fn = nn.BCELoss()
y_pred=model(x)
cost=loss_fn(y_pred,y)
```

Multi-class Classification

What about classification tasks with more than two categories?

- Targets form a discrete set $\{1, ..., k\}$.
- It's often more convenient to represent them as indicator vectors, or a one-of-k encoding:

Model: softmax function

$$\hat{y}_k = softmax(z_1 \dots z_k) = \frac{e^{z_k}}{\sum_k e^{z_k}}$$

where

$$z_k = \sum_j w_{kj} x_j + b$$

Loss Function: cross-entropy for multiple-output case

$$\mathcal{L}_{CE}(\hat{y}, y) = -\sum_{k=1}^{K} y_k \log \hat{y}_k$$
$$= -\mathbf{y}^{\mathbf{T}} \log \hat{\mathbf{y}}$$

```
z = nn.linear(feature_dim, num_class)
y = F.softmax(z)
loss = nn.CrossEntropyLoss()
```

Multi-class Classification

Cost funcion

$$J_\theta = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_{CE}(\hat{y}, y)$$

$$= \frac{-1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_k \log \hat{y}_k$$

The gradient descent algorithm will be:

$$\mathbf{w}^{t+1} = \mathbf{w}^{t} - \alpha\frac{\partial J_\theta}{\partial \mathbf{w}} \text{ where } \frac{\partial J_\theta}{\partial \mathbf{w}} = \frac{1}{N}\mathbf{x}^{\mathbf{T}}(\hat{\mathbf{y}} - \mathbf{y})$$

$$\mathbf{b}^{t+1} = \mathbf{b}^{t} - \alpha\frac{\partial J_\theta}{\partial \mathbf{b}} \text{ where } \frac{\partial J_\theta}{\partial \mathbf{b}} = \frac{1}{N}(\hat{\mathbf{y}} - \mathbf{y})$$

DEMO

Outline

Data: Irrelevant Features

Consider relevant features (inputs) $\rightarrow$ features that are correlated with prediction.

- ML systems will only learn efficiently if the training data contain enough relevant features.

The process of identifying relevant feature from data is called Feature engineering.

Feature Engineering

Involves:

- Feature selection: selecting the most useful features to train on among existing features
- Feature extraction: combining existing features to produce a more useful one (e.g Dimension reduction)

Overfitting and Underfitting

Central ML challenge: ML algorithm must perform well on new unseen inputs $\Rightarrow$ Generalization.

- When training the ML model on training set we measure training error.
- When testing the ML model on test set we measure test error (generalization error) $\Rightarrow$ should be low as possible.

The performance of ML models depends on these two factors:

1. generation error $\Rightarrow$ small is better.
2. the gap between generalization error and train error.

Overfitting and Underfitting

Overfitting (variance) : Occur when the gap between training error and test error is too large.

- The model performs well on the training data, but it does not generalize well.

Underfitting (bias): Occur when the model is not able to obtain sufficiently low error value on training set.

- Excessively simple model

Both underfitting and overfitting lead to poor predictions on new data and they do not generalize well.

Overfitting and Underfitting

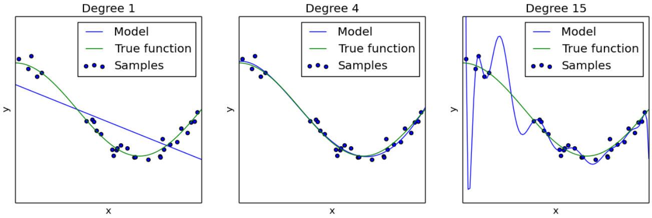


Figure 4: Overfitting vs Underfitting: `credit: scikit-learn.org`

Overfitting and Underfitting

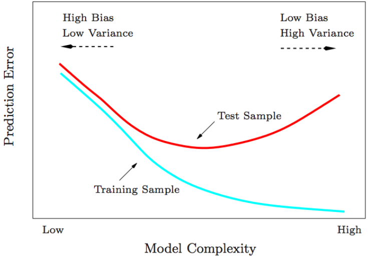


Figure 5: Model complexity: `credit: Gerardnico`

Variance-Bias Tradeoff

- For high bias, we have a very simple model.
- For the case of high variance, the model become complex.

To control bias and variance

1. Reduce number of features
2. Alter the capacity of the model (regularization).

Overfitting and Underfitting: Regularization

Regularization: Reduces overfitting by adding a complexity penalty to the loss function.

$$\mathcal{L}_{reg}(\hat{y}, y) = \mathcal{L}(\hat{y}, y) + \frac{\lambda}{2}\Omega(\mathbf{w})$$

where:

- $\lambda \geq 0$ is the regularization parameter.
- $\Omega(\mathbf{w})$ is the regularization functions which is defined as:

$$\Omega(\mathbf{w}) = \mathbf{w}^2 \text{ for L2 regulization}$$
$$\Omega(\mathbf{w}) = |\mathbf{w}| \text{ for L1 regulization}$$

Outline

Evaluation protocols

Learning algorithms require the tuning of many meta-parameters (Hyper-parameters).

Hyper-parameters

Hyper-parameter: a parameter of a model that is not trained (specified before training)

- Have a strong impact on the performance, resulting in over-fitting through experiments.
- We must be extra careful with performance estimation.
- The process of choosing the best hyper-parameters is called Model selection.

Evaluation protocols

The best practise is to spilit your data into three disjoint sets.

Train set

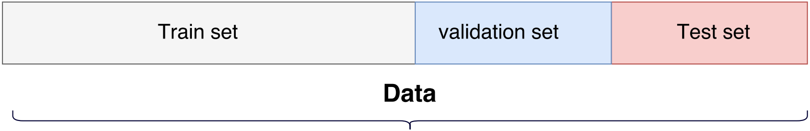
- Used for learning

Validation set

- Used for model selection

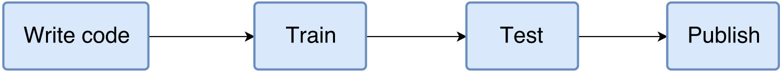
Test set

- Used for estimating generalization performance

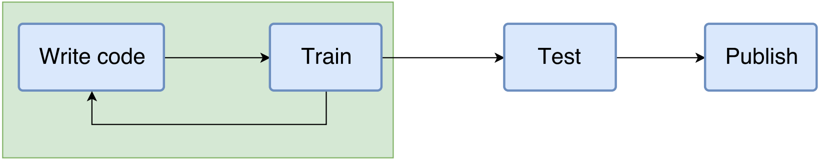


Evaluation protocols: Development cycle

The ideal development cycle is



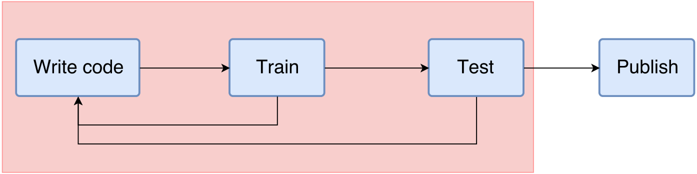
or in practice something like



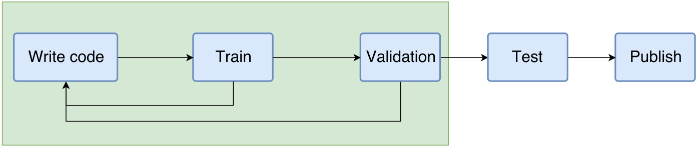
- There may be over-fitting, but it does not bias the final performance evaluation.

Evaluation protocols: Development cycle

Unfortunately, it often looks like



- This should be avoided at all costs.
- The standard strategy is to have a separate validation set for the tuning.



Evaluation protocols: Cross validation

Cross validation: Statistical method for evaluating how well a given algorithm will generalize when trained on a specific data set.

- Used to estimate the performance of learning algorithm with less variance than a single train-test set split.
- In cross validation we split the data repetedely and train a multiple models.

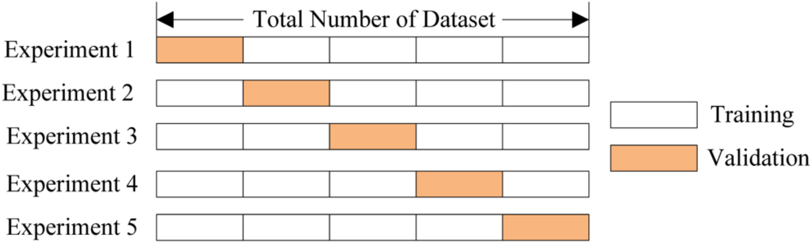


Figure 6: Cross validation: `credit: kaggle.com`

Evaluation protocols: Cross validation Types

K-fold cross validation

- It works by splitting the dataset into k-parts (e.g.k=3, k=5 or k=10).
- Each split of the data is called a fold.

Figure 7: K-Fold: `credit: Juan Buhagiar`

Startified K-fold cross validation

- The folds are selected so that the mean response value is approximately equal in all the folds.

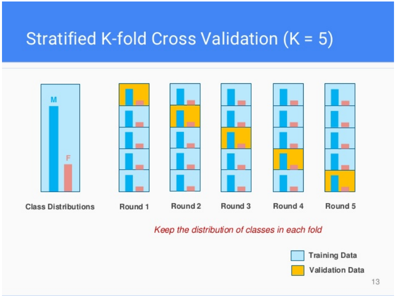


Figure 8: SK-Fold: `credit: Mark Peng`

Performance metrics

How to measure the performance of a trained model?

Many options availabe: depend on type of problem at hand.

- `Classification`: Accuracy, Precision, Recall, Confusion matrix etc.
- `Regression`: RMSE, Explained variance score, Mean absolute error etc.
- `Clustering`: Adjusted Rand index, inter-cluster density etc.

Example: scik-learn metrics

Outline

Data exploration and cleanup

- Spend time cleaning up your data ⇒ remove errors, outliers, nose and handle missing data.
- Explore your data: visualize and identify potential correlations between inputs and outputs or between input dimensions.
- Transform non-numerical data: on-hot encoding, embending etc.

ID	Gender
1	Male
2	Female
3	Not Specified
4	Not Specified
5	Female

ID	Male	Female	Not Specified
1	1	0	0
2	0	1	0
3	0	0	1
4	0	0	1
5	0	1	0

Figure 9: on-hot encoding `credit: Adwin Jahn`

Feature transformation

Normalization Make your features consistent $\Rightarrow$ easy for ML algorithm to learn.

1. Centering: Move your dataset so that it centered around the origin.
$$\mathbf{x} \leftarrow (\mathbf{x} - \mu)/\sigma$$
2. Scaling: rescale your data such that each feature has maximum absolute of 1.
$$\mathbf{x} \leftarrow \frac{\mathbf{x}}{\max |\mathbf{x}|}$$
3. scikit-learn example

Dimension Reduction

- PCA.

Lab 1: Machine Learning Fundamentals

Part 1.Regression Problem:
Objective: Implement machine learning algorithm to predict the best house price for a sample house using data related to housing prices at Boston from kaggle dataset.

Part 2.Classification Problem:
Objective:Predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the PIMA dataset.

References I

- Intro to Neural Networks and Machine Learning:csc321 2018, Toronto University.
- Deep learning in Pytorch, Francois Fleurent: EPFL 2018.
- Machine Learning course by Andrew Ng: Coursera.