Deep Learning Cheat Sheet

Evaluation Metrics

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

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

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

$$Specificity = \frac{TP}{TN + FP}$$

$$Fscore = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

$$macro\ average = \frac{1}{n} \sum_{i=1}^{n} avg_i$$

Activation Functions

Sigmoid : $f(z) = \frac{1}{1+e^{-z}}$ — Smooth and differentiable. Used in output layers for binary classification.

Hyperbolic Tangent (tanh) : $f(z) = \tanh(z)$ — Smooth, differentiable, output centered around 0. Used in LSTM.

Rectified Linear Unit (ReLU) : $f(z) = \max(0, z)$ — Non-linear, used as a standard, but has dying units problem for $z < 0$.

Leaky ReLU : $f(z) = \begin{cases} z & \text{if } z \geq 0 \\ \alpha z & \text{if } z < 0 \end{cases}$ — Addresses dying units problem with a small α (typical $\alpha = 0.01$).

Exponential Linear Unit (ELU) : $f(z) = \begin{cases} z & \text{if } z \geq 0 \\ \alpha(e^z - 1) & \text{if } z < 0 \end{cases}$ — Similar to Leaky ReLU but more computationally expensive.

Softmax : $f(z_i) = \frac{e^{z_i}}{\sum_{j=0}^{K-1} e^{z_j}}$ — Used in the last layer for multi-class classification, outputs a probability distribution.

Data Preparation

Min-max [0,1] : $x' = \frac{(x - x_{min})}{(x_{max} - x_{min})}$

Min-max [-1,1] : $x' = 2 \cdot min\_max(x) - 1$
min-max doesn't handle outliers.

Z-norm : $x' = \frac{(x - \mu)}{\sigma}$

Scaling & Centering
Scaling improves the numerical stability, the convergence speed and accuracy of the learning algorithms. Centering improves the robustness of the learning algorithms

Gradient Descent

Initialize parameter vector θ_0
repeat
  Compute the gradient of the cost function at current position $\theta_t : \nabla_\theta J(\theta_t)$
  Update the parameter vector by moving against the gradient : $\theta_{t+1} = \theta_t - \alpha \cdot \nabla_\theta J(\theta_t)$
  where α is the learning rate.
until change in θ is small

MSE

$$J_{MSE}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}(i) - y(i))^2$$

where :

- $\hat{y}(i) = h_\theta(x(i))$ is the prediction of the model,
- $y(i)$ is the true outcome,
- m is the number of training examples.

$$\nabla_w J_{MSE}(w, b) = \frac{1}{m} \sum_{i=1}^{m} \hat{y}(i) \cdot (1 - \hat{y}(i)) \cdot (\hat{y}(i) - y(i)) \cdot x(i)$$

$$\nabla_b J_{MSE}(w, b) = \frac{1}{m} \sum_{i=1}^{m} \hat{y}(i) \cdot (1 - \hat{y}(i)) \cdot (\hat{y}(i) - y(i))$$

Cross Entropy

$$J_{CE}(\theta) = -\sum_{i=1}^{m} y(i) \cdot \log h_\theta(x(i)) + (1 - y(i)) \cdot \log(1 - h_\theta(x(i)))$$

where :

- $p_\theta(y(i) \mid x(i))$ is the probability model parameterized by θ, predicting the probability of the true class $y(i)$ given the input $x(i)$,
- m is the number of observations or data points in the dataset.

$\nabla_w J_{CE}(w, b) = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}(i) - y(i)) \cdot x(i)$
$\nabla_b J_{CE}(w, b) = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}(i) - y(i))$

Performance Measures

Matrice de Confusion
Confusion Table
ROC
Precision
Recall

Bias & Variance

Model Selection
Bias
Variance

Theory

Compute Graph
Universal Approximation Theorem

Curse of Dimensionality

Backpropagation

MLP Layer
Matrix Notation
Full Batch
Batch Normalization

BGD	SGD	MBGD
Smooth	Wiggling, needs smoothing	Slightly wiggling
Not wiggling	Wiggles around minimum	Wiggles around minimum
Strictly decreasing cost	Not necessarily decreasing cost	Typically decreasing cost
Many epochs needed	Few epochs needed	Less epochs than BGD, more than SGD needed
Choose larger learning rate	Choose smaller learning rate	Choose medium learning rate (dependent on model)
No out-of-core support – all data in RAM ($\sim m$).	Out-of-core support - not all data to be kept in RAM of a single machine.	Out-of-core support - not all data to be kept in RAM of a single machine
Easy to parallelise	Not easy to parallelise	Easy to parallelise

Vanishing Exploding Gradient

Saturation
Variance Change
Xavier & Heu Initialization
Batch Normalization
Non Saturating Activation Function
Gradient Clipping

Optimizers

Momentum
AdaGrad
RMS Prop
Adam
Scheduler

Regularization

Weight Penalty
Dropout
Early Stopping

CNN

Convolutional Layer
Pooling Layer

Unbalanced Dataset

Bayesian Approach
Discrete
Continuous
Medical Test

DeepCNN

Conf2D Params
MaxPooling
LeNet5
AlexNet
VGGnet
GoogleNet
ResNet
Pattern

Feature Visualization

Data Preparation
Network
Compile
Evaluate
Activation Map

Data Augmentation

Principle
Types
Strategies
Keras

Functional API

Sequential vs Functionals
Architecture 1
Architecture 2
Architecture 3

Transfer Learning

Principle
Keras Code
MobileNet
Strategies

RNN

Use Case
Model Category
Recurrence Net
Single Layer
Many to Many
Un exemple par catégorie
Stacked RNN

LSTM

Long Term Memory Unit Cell
Gates
Backprop
Keras
GRE

Word Embedding

Word
Training

Sentiment Classification

Strategy

Autoencoder

Definition
Use Case

GenRNN

Many to Many
Many to One

Attention

Sequence to Sequence
Attention

Transformer

High-Level Architecture
Self-Attention
Full Architecture