

1

<ol style="list-style-type: none"><li>1. Propagation-Retro-propagation</li><li>2. Fonction d'activation</li><li>3. Fonction cout</li><li>4. Variantes du SGD</li></ol>	<p><i>Rappel</i></p>
--	----------------------

2

# Ce qu'on va voir

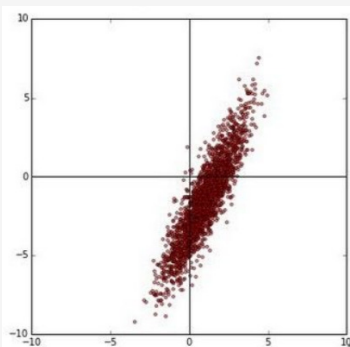
## Contenu

1. K-fold validation
2. Mesures d'évaluation de classification
3. Grille de recherche de paramètres
4. Eviter le surajustement grâce à la régulation

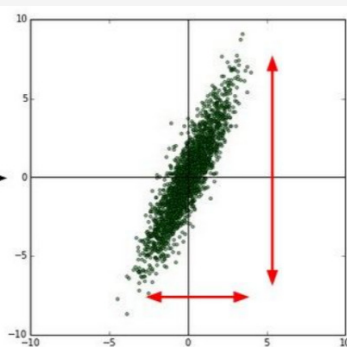
3

## Prétraitements des données

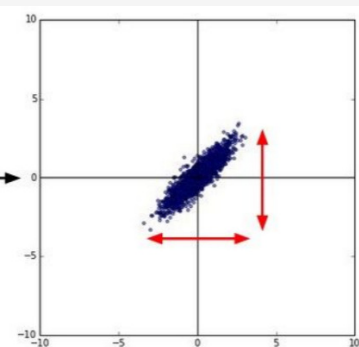
Données originales



Données centrées



Données normalisées



```
X -= np.mean(X, axis = 0)
```

```
X /= np.std(X, axis = 0)
```



4

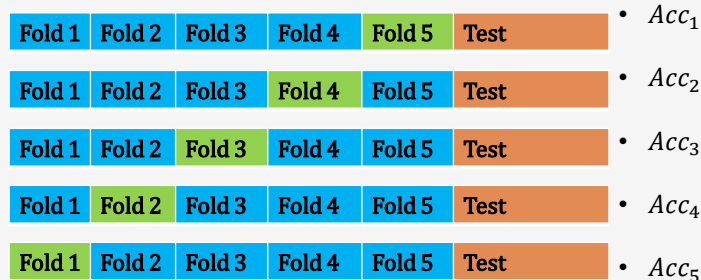
## Prétraitements des données

- Autres détails à propos des données centrées
- E.g on a une base d'image de  $225 \times 225 \times 3$  ( des image RGB)
- **Soustraire l'image moyenne**
  - La moyenne est une image de  $225 \times 225 \times 3$
- **Soustraire la moyenne par canal**
  - La moyenne par canal, 3 scalaires: [moyenne R, moyenne G, moyenne B]

5

## Partage de la base: validation K-fold

- **Train-Test:**  • Accuracy
- 80-20%; 70-30%
- **Train-Val-Test:**  • Val-accuracy  
• Test-accuracy
- 60-20-20%
- **Train-k-fold-Test:**
  - For  $i=1:k$ :
    - Train set:  $k-1$  folds
    - Val set: fold  $i$
    - Test: Test



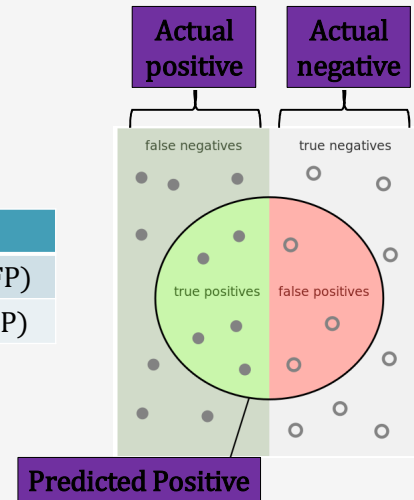
$$Acc = \sum_i Acc_i$$

6

## Mesure de d'évaluation de classification

- Confusion Matrix

		Predicted	
		Negative	Positive
Actual	Negative	True Negative (TN)	False Positive (FP)
	Positive	False Negative (FN)	True Positive (TP)



7

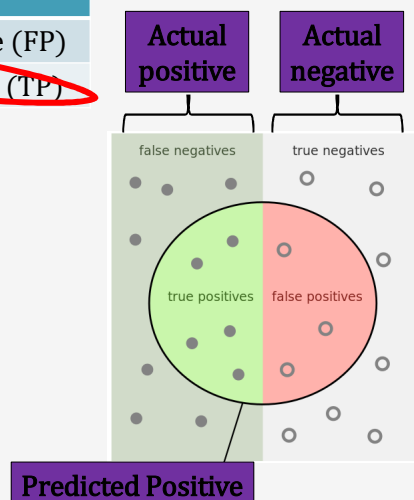
## Mesure de d'évaluation de classification

- Accuracy:

		Predicted	
		Negative	Positive
Actual	Negative	True Negative (TN)	False Positive (FP)
	Positive	False Negative (FN)	True Positive (TP)

- Performance générale du modèle

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



8

## Mesure de d'évaluation de classification

### • Precision:

		Predicted	
		Negative	Positive
Actual	Negative	True Negative (TN)	False Positive (FP)
	Positive	False Negative (FN)	True Positive (TP)

- Précision du modèle par

rapport aux « **prédiction positive** »

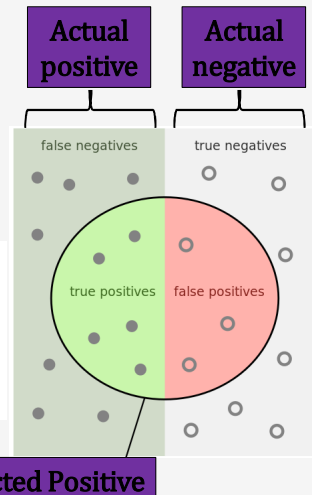
- Idéalement retourner une faible valeur pour les FP.

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

- E.g. Classification email (spam/non-spam)

Un email non spam (actual negative) est identifié comme spam (predicted positive)

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$



9

## Mesure de d'évaluation de classification

### • Recall:

		Predicted	
		Negative	Positive
Actual	Negative	True Negative (TN)	False Positive (FP)
	Positive	False Negative (FN)	True Positive (TP)

- Combien de « **actual positive** »

le modèle identifie comme positif (true positif)

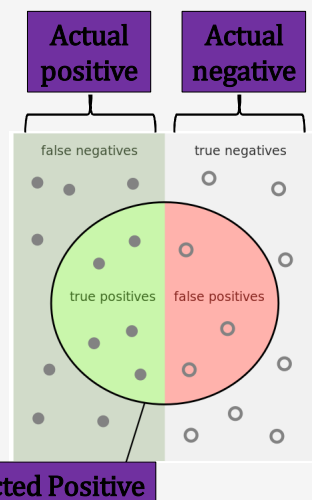
- Idéalement retourner une faible valeur pour les FN.

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

- E.g. Classification de patient contaminés (sain/malade)

Une personne malade (Actual positive) est prédite par le modèle comme saine (Prédicated negative)

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

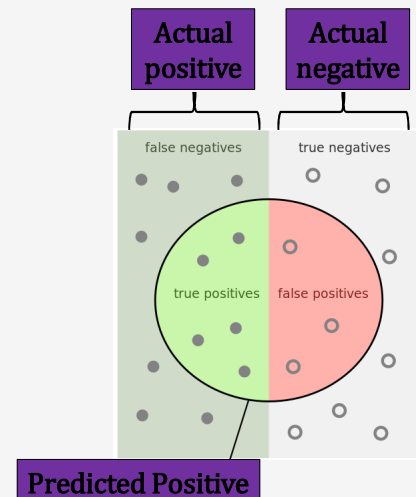


10

## Mesure de d'évaluation de classification

- **F-score:**
- Balance entre la Précision et le Rappel
- Différence avec Accuracy?
- Accuracy est sensible au True Negative.
- Dans de nombreuses applications les False Negative et les False Positive ont plus d'impact

$$F - score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$



11

## Optimisation des hyperparamètres

- **Paramètre:** les poids  $W^l$  et les biais  $b^l$  (initialisés aléatoirement avec valeur  $[0,1]$ )
- **Hyper-paramètres:**
  - Taux d'apprentissage  $\eta$  (*eta*) ou bien  $\alpha$  (*alpha*)
  - Epoch (n° d'itérations pour l'algorithme d'optimisation)
  - n° de couches cachées
  - n° de neurones dans chaque couche cachées
  - Fonction d'activation dans les couches cachées
  - Taille du mini-batch

12

## Optimisation des hyperparamètres

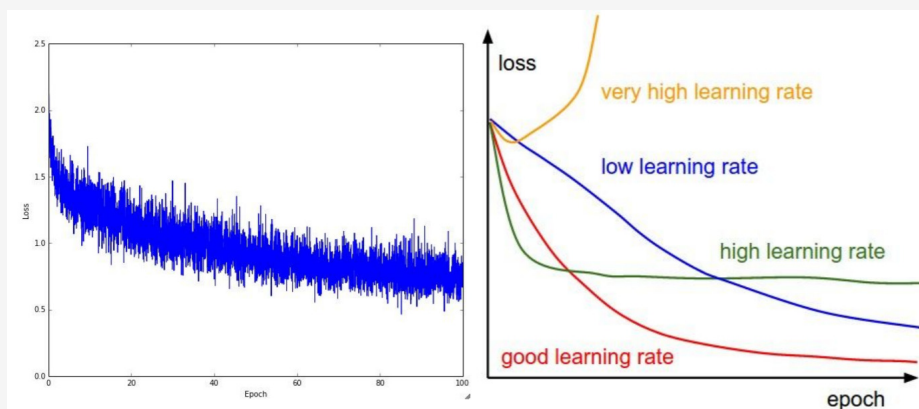
- **Paramètre:** les poids  $W^l$  et les biais  $b^l$
- **Hyper-paramètres:**
  - Taux d'apprentissage  $\eta$  (*eta*) ou bien  $\alpha$  (*alpha*)
  - Epoch (n° d'itérations pour l'algorithme d'optimisation)
  - n° de couches cachées
  - n° de neurones dans chaque couche cachées
  - Fonction d'activation dans les couches cachées
  - Taille du mini-batch

**Processus Empirique**  
**Dépend du problème à traiter**

13

## Optimisation des hyperparamètres

- **Hyper-paramètres:**
  - Taux d'apprentissage  $\eta$  (*eta*) ou bien  $\alpha$  (*alpha*)

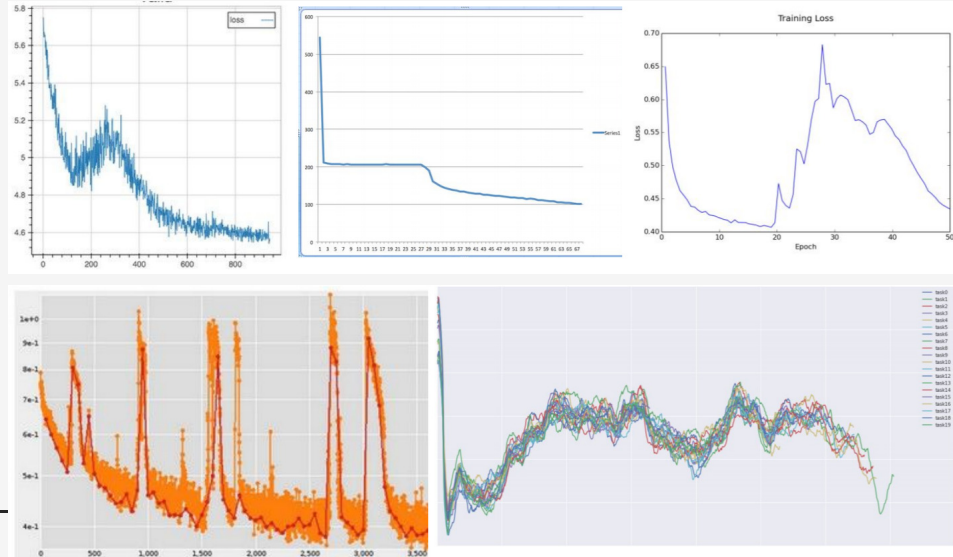


14

# Optimisation des hyperparamètres

$\eta$  (eta)

C'est pas toujours évident à expliquer!



15

# Optimisation des hyperparamètres

## • Hyper-paramètres:

– Dégradation de  $\eta$

○ *Inverse (Timed-based)*

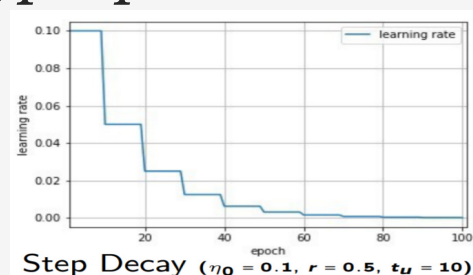
$$\eta_t = \frac{\eta_0}{1 + r \times t} \quad | \quad r: \text{decay rate}$$

○ *Exponential decay*

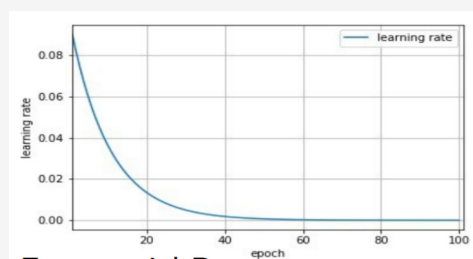
$$\eta_t = \eta_0 \times e^{-\lambda t}$$

○ *Step decay*

$$\eta_t = \eta_0 \times r^{t/t_u}$$



Step Decay ( $\eta_0 = 0.1$ ,  $r = 0.5$ ,  $t_u = 10$ )

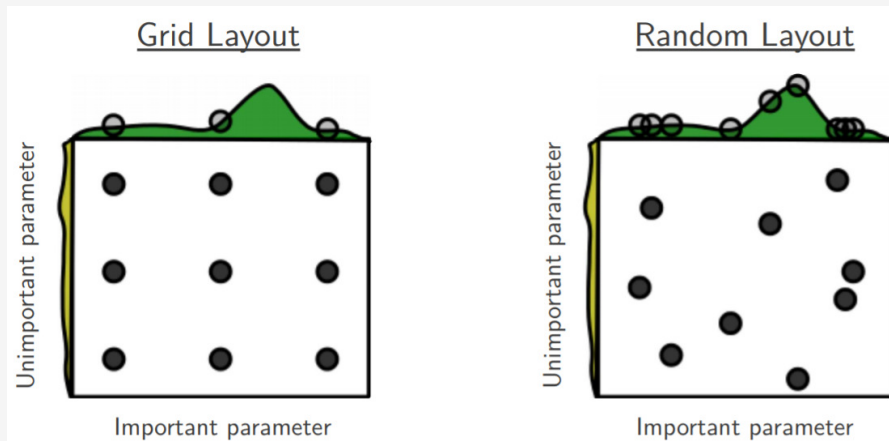


Exponential Decay ( $\eta_0 = 0.1$ ,  $\lambda = 0.15$ )

16



## *Optimisation des hyperparamètres: grille de recherche vs. recherche aléatoire*

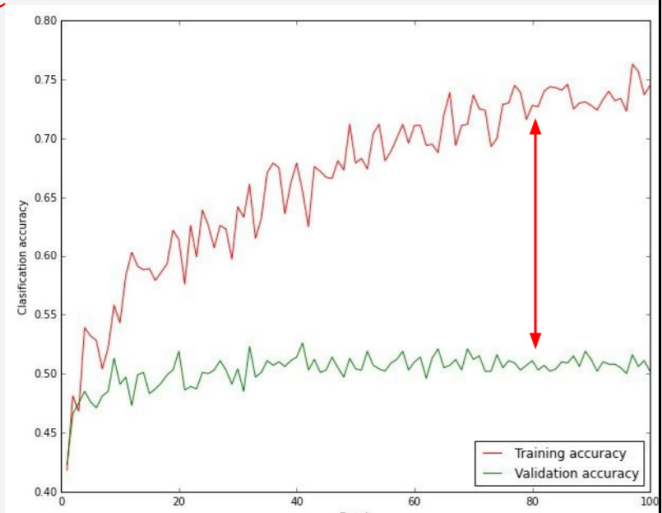


- J. Bergstr, Y. Bengio. « aRandom Search for Hyper-Parameter Optimization », Journal of Machine Learning Research, 2012

17

## *Sur-apprentissage*

- **Grand écart => sur-apprentissage**
- Le modèle ne peut pas généraliser sur l'ensemble Test (qu'il n'a pas vu auparavant).



18

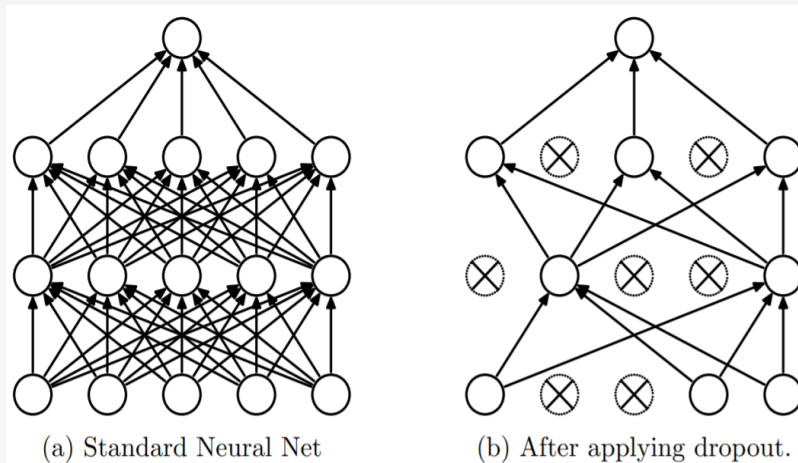
## Régularisation

- Solution:
  - Ajouter plus de données (pas évident et couteux)
  - Entraîner plusieurs modèles NN sur l'ensemble de données et prendre l'accuracy moyenne (très couteux)
  - Utiliser une régularisation

Améliore la généralisation du modèle

19

## Régularisation: Drop-out

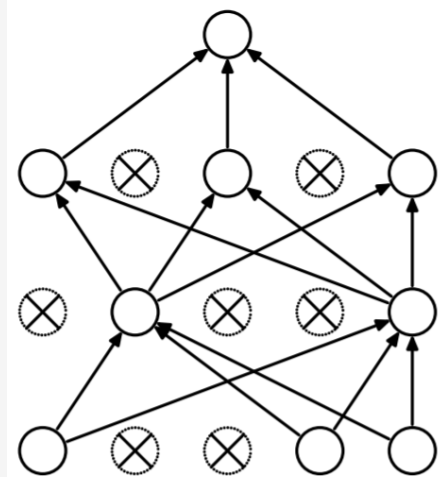


- N. Srivastava, et al 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting' , Journal of Machine Learning Research, 2014

20

## Régularisation: Drop-out

- **Processus aléatoire:**
- Durant l'entraînement, quelques neurones sont ignorés aléatoirement.
- Equivaut à entrainer les données avec des modèles différents des réseaux de neurones en parallèles.
- Dropout essaye de perturber la sur-adaptation des neurones aux données, et ainsi rendre le réseaux robuste.
- Dropout réduit les capacité du réseau => penser à l'appliquer sur un grand réseau (assez profond)

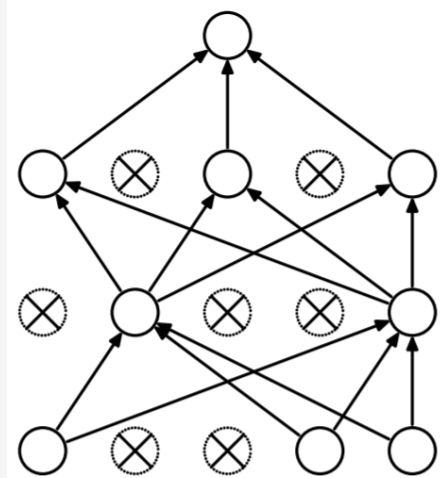


- N. Srivastava, et al 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting', Journal of Machine Learning Research, 2014

21

## Régularisation: Drop-out

- **Comment faire un Dropout?:**
- Dropout est implémenté au niveau des couches cachées ou ben couche d'entrée, mais pas la couche de sortie!.
- Implémenté à l'aide d'un hyperparamètres; une probabilité des neurones à retenir (ou bien à ignorer) au niveau de la couche courante.
- Généralement, 0.5 pour retenir les neurones dans une couche cachées, et un large valeur, 0.8 pour retenir les neurones de la couche d'entrée.



- N. Srivastava, et al 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting', Journal of Machine Learning Research, 2014

22