1 Pré-traitement

Observer si **l'ordre des valeurs** diffèrent beaucoup entre des variables. Typiquement, si une VAR1 est borné en [-2, 4] et une VAR2 en [0.7, 1.3], une normalisation est nécessaire. Cela permet d'éviter qu'une variable (dans cet ex. VAR1) influence trop sur le résultat final.

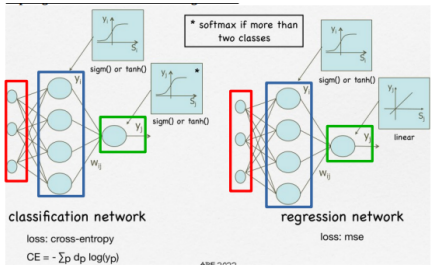
1.1 Normalisation dans un intervalle [0, 1]

$$X' = \frac{X_{in} - X_{min}}{X_{max} - X_{min}}$$

1.2 Normalisation dans un intervalle [-1, 1]

$$X' = 2 \cdot \frac{X_{in} - X_{min}}{X_{max} - X_{min}} - 1$$

2 Topologies entre classification et régression



Dans le sens de lecture

Couche d'entrée : nb. variables d'entrées

Couche cachée : nb. neurones cachés

Couche de sortie : nb. de variabels de sorties

2.1 Remarques

On a des poids synaptiques sur **chaque connexion**.

Pour calculer le **total**, on somme chaque poids possible :

— Poids **d'entrée** : nb. entrées · nb. neurones

— Poids **de sortie** : nb sorties · nb. neurones

— Ne pas oublier les **biais**

Pour encoder le résultat, le plus simple est de partir sur du One-Hot, **1 parmis N** (un bit actif par sortie possible).

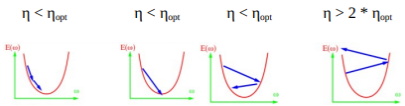
2.2 Considérations pratiques

Topologie : Nombre de couches, Nombre de neurones cachés par couche

Initialisation des poids : Trop faible → tout a le même effet, Trop grand → les fonctions d'activations saturent

Fonctions d'activations : Sigmoïde, Tangente Hyperbolique et ReLu

Learning rate



3 Méthodes pour éviter l'overfitting

Arrêt prématuré, Régularisation, Data augmentation, Dropout, Réduire complexité

4 Sélection du modèle

Choisir des hyper-paramètres/optimisation → Cross-Validation par configuration → Évaluer performance → Tester sur données réelles → Évaluer performance

Configuration selon : Nombre de couches, Nombre de neurones par couche, learning rate, momentum, nombre d'epochs, ...

5 Fonction du MLP

$$Y_k = f\left(\sum W_{ik} \cdot X_i + b_k\right)$$

6 Cross-validation

On a 1'000 observations et il est réservé 20% pour créer un ensemble de test. Le reste est alors utilisé pour faire une validation croisée 10-fold.

Informations

1000 observations, 200 sont des tests. 800 pour train / validation.

Calculs

La validation croisée nous indique que l'on peut séparer notre base de train./valid. 10 fois.

La taille des batchs est identique, donc dans ce cas, on aura des batchs de 80 observations.

On prend ensuite un de ces batchs pour validation et le reste d'entraînement, soit une séparation 80/720.

Ces valeurs permettent de définir le nombre de fois où les poids sont mis à jour : 720 fois

Et si l'on utilise une taille de batch de 10, les poids seraient mis à jour : $^{720}/_{10} = 72$ fois.

7 Matrice de confusion

		Prédit	
Réel		TP	FN
		FP	TN

accuracy : $\frac{TP+TN}{TP+FN+FP+TN}$

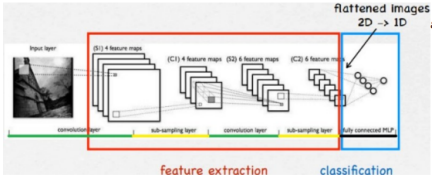
precision : $\frac{TP}{TP+FP}$

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

f-score : $\frac{2TP}{2TP+FP+FN}$

8 CNN

Permet de résoudre des problèmes de reco. d'objets, par la correspondance naïve des formes. Si un objet n'apparaît pas dans les données avec la même taille, au même endroit, le chevauchement entre le modèle et l'objet reconnu peut être faible.



Couches de convolution : Tailles de noyau variantes (3x3, 5x5, 7x7) permettant l'identification sur différentes échelles.

Couches de sous-échantillonnage : Max-pool élimine les valeurs non-maximales. But : Réduire le calcul des couches supérieures et fournit un « résumé » des stats des caractéristiques des couches inférieures → image résultante plus petite

Couches entièrement connectées : Perceptron multicouche/Neurone Artificiel ⇔ réseau peu profond

8.1 Architectures

AlexNet (2012) : Utilise ReLus (au lieu de tanh), augmentation des data et décrochages (dropout) nécessaires pour éviter de

ZF Net (2013) : Utiliser des filtres plus petits (7x7) et augmente le nombre de filtres

VGG Net (2014) : Utilise uniquement des filtres 3x3 avec un stride et pad de 1 ainsi que des couches maxpooling 2x2 avec un stride 2 : - Rééchantillonnage de l'entrée avec stride (déplacement de X pixels entre les applications du filtre) - Les noyaux convolutifs de taille 11x11 (121param), 5x5 (25param) et 3x3 (9param) peuvent être reproduits en utilisant plusieurs noyaux 3x3 comme blocs de construction, réduisant le nombre de paramètres à apprendre. (stride = 1) 5x5 2 layer 3x3 // 11x11 5 layer 3x3.

GoogLeNet (2015) : Nouvelle architecture, sans couche entièrement connectées. - Inception Networks : Au lieu de choisir une taille de kernel pour une couche, on les utilise tous en parallèle et on concatène la sortie. On utilise des convolution 1x1 avant des convolution 3x3 ou 5x5 pour mettre en commun les caractéristiques et réduire le nombre d'opérations. - Batch-normalization : L'apprentissage est amélioré si les inputs sont normalisés. normalization sur des sous réseau ou couche. Normalisation entrainable qui s'applique sur chaque mini-batch. Permet d'accélérer la converge d'entrainement donc un learning rate plus haut (et plus rapide directement).

DenseNet vs ResNet : Dense allège le probl. du vanishing gradient, renforce la propag. des features et encourage la réutili. des features il réduit beaucoup le nombre de paramètre.

ResNet (2015) : Dégradation des performances plus le réseau est profond. Sauf en utilisant des « raccourcis de connexion ». (152 couches) - Les blocks résiduels assument qu'il est plus facile d'apprendre x F(x) + x que x H(x) (H est new)

DenseNet (2017) : Pour chaque couche, les sorties de convolution de toutes les précédentes couches sont utilisées comme entrées. Cela permet d'atténuer le problème de vanishing gradiant, renforcer la propagation des features, favoriser la réutilisation des features, réduire le nombre de paramètres

EfficientNets (2020) : Equilibrer soigneusement la profondeur, la largeur et la résolution peut mener à de meilleures performances.

9 Transfer learning

Utiliser les première couches d'un modèle CNN qui a été entrainé sur des données (beaucoup) et faire en sorte d'uniquement fine-tune les couches suivantes pour l'utiliser sur une nouvelle tâche.

Exemple MobileNet. On charge le modèle et ses poids, on modifie les layers subséquent et on drop les sorties originales par les notres qui matchent le nombre de sorties que l'on veut.

On freeze les premiers layers et on set les derniers comme entrainable et ensuite on entraine ce modèle.

Vector embedding : On peut réduire nos objets à des vecteurs de nombre on peut les considérer comme des features de nos objets. Cela permet de faire des rapprochements sémantiques en calculant leur proximité dans un espace vectoriel. Pour faire du transfer learning on peut également avoir un modèle qui précalcule des vector embeggings à partir des données d'entrée et utiliser ces vecteurs en tant qu'entrer dans un nouveau modèle pour entrainer des nouvelles données, par exemple K-NN.

Few-shot learning : Modèle contenant peu d'exemple. Entrainer un modèle qui résoud plusieurs tâches et s'attendre qu'il résolve une nouvelle.

Meta-learning : entrainer un modèle un certains nombre d'epochs sur une variété de tâches t.q il peut résoudre des nouvelles tâches sur un petit nombre d'exemple.

Avantages CNN : extraction feature automatique (avant : HOG et SIFT). Objectifs : trouver des patterns et des features automatiquement, questions à se poser : est-ce que les patterns et correlation trouvés ont du sens.

Performances : une bonne perf sur un dataset de benchmark ne garanti pas de bonnes perf dans la vraie vie. On ne peut pas prédire à 100% le comportement d'une solution CNN. Faire attention aux biais !

Spatial translation invariance : reconnaitre des objets apparaissant à différentes échelles.

Bruteforce correlations : un visage ayant plus de yeux qu'il ne faut peut être associé à une classe face de manière plus sûre qu'un visage normale. (output max by image occlusion).

Adversarial attack : trouver des moyens de tromper l'IA (ascent gradient ou algorithme evolutif)

One-pixel attack : on peut perturber un modèle en modifiant un pixel (cas ImageNet).

Visualization tools : feature map, activation maximization, filter activation statistics, deconvolution, class activation maps, occlusion analysis.

Activation maximization : prendre le gradient de l'activation par rapport à l'entrée pour maximiser l'activation.

Class activation maps : Somme pondérée des activations du dernier layers. Upsampled pour match la taille en entrée. Heatmap des endroits où le CNN porte son attention.