Introduction

NN (perceptron) depuis (60-70?) aujourd'hui, bc d'applications, CNN Notre but est alors de comprendre et savoir comment mettre en oeuvre, NN qui s'execute dans un temps raisonnable

Table of contents

1 What is a Neural Network?

Peut être vu comme une fonction qui prend des données (une image) en entrée est des classes (par exemple animal) en sortie Bien évidemment, c'est une fonction très compliqué avec par exemple pour un image RGB, X inputs D'où l'idée de la considerer comme une composition des fonctions facilement (linéaire) calculé par les ordis La structure est la suivante: (input, neurons, weights, bias, output) <- ce sont les paramtres, des milliers! La question est donc comment trouver les parametres, on parle alors de l'apprentissage du NN.
Apprentissage: Le principe est simple: On part d'une (dataset) déja classifié, un ("training set") et on modifie le NN pour Pour cela il faut introduire une fonction qui quantifie la précision du NN, "fonction perte" ou "cost function" notée d. C'est une fonction qui prend comme argument le resultat du NN noté y_tilde et le compare ave le resultat attendu y Ici, on connait le y correspondant à chaque input et on cherche à minimiser cette fonction perte en ajustant les param du res (càd w,b) L'apprentisage peut donc être vu comme un problème d'optimisation ... (Pour representer), on montre le principe avec un algorithme d'optimisation classique, le gradient descent. En gros, le gradient de la f perte nous dit quelles parametres faut-il augmenter et diminuer (et avec quelle poids) pour augmenter la f perte le plus plus possible Or, on veut minimiser donc on prend le moins de ca. Permettant (pour le moment), que l'on sache calculer le gradient à chaque itéretion. Notre "stratégie" et donc d'initialiser le NN aleatoirement puis, on "suit" le gradient à chaque itéreation jusqu'à ce que on trouve un min il est important de noter que la fonction perte est en fait la somme de d de toutes les données Finalement, on peut tester avec "testing set" ...
Realisation: Fonction activation Introduit de la non-linéarité, raisons d'optimisation: exemple: arctan Softmax les sorties sont positives et la somme fait 1, peut être vu comme proba Fonction Perte distance -> fonction perte
Forward Backward Algorithme d'optimisation

2 What is a *Conolutional* Neural Network

CNN, en effet c'est très semblable à un NN classique: on ajoute des couches convolutionnels. Il se trouve que ca donne des resultats impressionants. Ici les filtres sont optimisés Convolution Downsampling Backward

3 Application

Le probleme MNIST (classiqe) greyscale (entre 0 et 1), une seule channel "Par batch" Choix de fonctions maxpool reLU plus simple que arctan et sigmoid, très utilisé en pratique Adam adapté à optimisation par batch, plus sophistiqué que Stochastic Gradient Desc Choix de structure 2 couches conv, 8 filtres, stride = 1, batchsize = 2 couches denses (dims) Resultat / temps d'execution Modifications

4 Conclusion

...