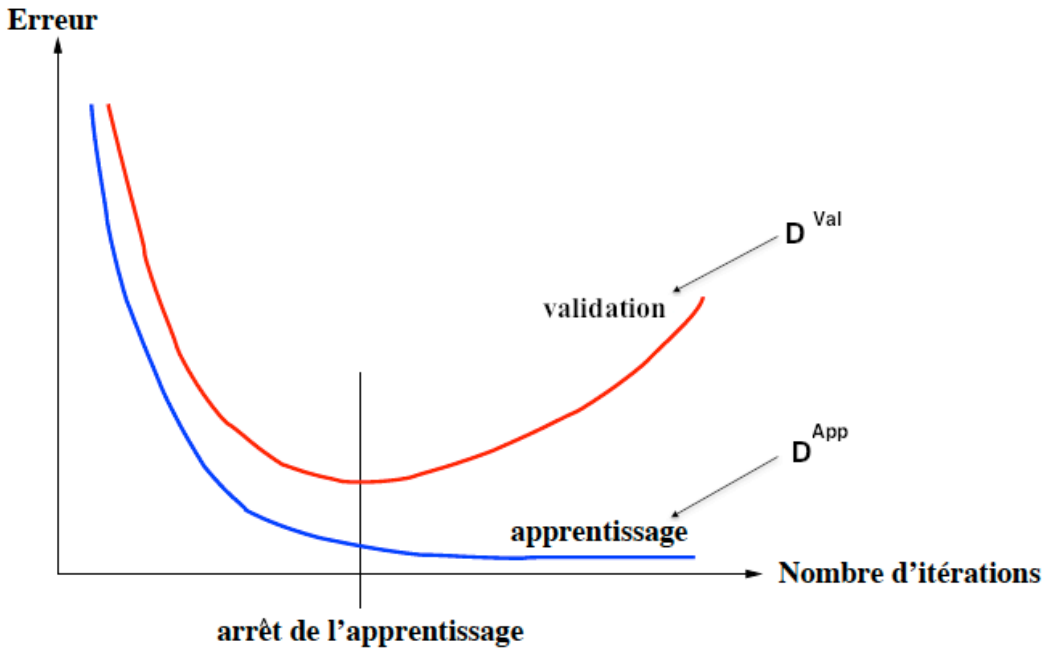
Perceptron multi-couche : régression (suite) et classification supervisée

"Early stopping"

→ Trop apprendre à partir d'un ensemble de taille limitée nuit aux performances en généralisation.
→ Pour éviter le phénomène de sur-apprentissage, on peut **utiliser la méhode "early stopping"** pour **déterminer quand arrêter l'apprentissage.**



- On utilise $\mathbf{D}^{App}$ pour calculer les poids du réseau (faire l'apprentissage)
  ☞ **L'erreur sur $\mathbf{D}^{App}$ décroît puis se stabilise** ⟶ voir **la courbe en bleu**
- On utilise $\mathbf{D}^{Val}$ pour déterminer quand arrêter l'apprentissage
  ☞ **L'erreur sur $\mathbf{D}^{Val}$ passe par un minimum (c'est là qu'il faut arrêter l'apprentissage : voir la droite verticale) avant de croître.** ⟶ voir **la courbe en rouge**
- A la fin de l'apprentissage, on utilise $\mathbf{D}^{Test}$ **pour évaluer les performances du réseau obtenu.**

Utiliser un réseau avec une couche cachée pour faire des apprentissages et utiliser le "early stopping" pour arrêter l'apprentissage.

1 Un problème réel de classification supervisée : les Iris de Fisher

les Iris de Fisher (**voir fiche 4**) correspondent à 150 fleurs décrites par 4 variables quantitatives : **longueur du sépale**, **largeur du sépale**, **longueur du pétal** et **largeur du pétal**. Les 150 fleurs sont réparties en 3 différentes espèces : **iris setosa**, **iris versicolor** et **iris virginica**. Chaque classe est composée de 50 fleurs. La classe setosa est linéairement séparable des deux autres, alors que versicolor et virginica ne le sont pas. Le fichier **iris_don.mat** contient les vecteurs d'entrée en dimension 4 décrivant les 150 iris. Le fichier **iris_cls.mat** contient les classes des 150 iris.
Les données sont disponibles en ligne à l'adresse suivante : `http://deptinfo.cnam.fr/new/spip.php?article851`
Télécharger puis décompresser les deux fichiers **iris_don.mat.zip** et **iris_cls.mat.zip**
Pour lire ces fichiers sous Matlab faire : **load -ascii** iris_don.mat
load -ascii iris_cls.mat

Codage des sorties

Le codage classique des sorties désirées pour la classification utilise un neurone de sortie par classe, avec une valeur désirée haute pour le neurone de la classe correcte, et une valeur désirée faible pour les autres classes.

Nous allons utiliser le codage suivant : **classe 1** $\longrightarrow$ **(1 0 0)**
classe 2 $\longrightarrow$ **(0 1 0)**
classe 3 $\longrightarrow$ **(0 0 1)**

Vous pouvez générer la matrice de sortie en utilisant les instructions Matlab suivantes :

```
Iris_output=zeros(150,3);
Iris_output(find(Iris_cls==1),1)=1;
Iris_output(find(Iris_cls==2),2)=1 ;
Iris_output(find(Iris_cls==3),3)=1;
```

Fonction d'activation des neurones de sortie

Pour un problème de classification, il vaut mieux utiliser des fonctions d'activation non linéaires à la couche de sortie. Utiliser la fonction **'softmax'** pour le problème des Iris.

Travail à faire

Utiliser 120 iris pour l'apprentissage et les autres pour éviter le sur-apprentissage.
→ Ensemble d'apprentissage : 120 iris
→ Ensemble de validation : 30 iris
Utiliser l'algorithme 'scg'.
Faire varier le nombre de neurones cachés.
Pour chacune de ces experiences,
a) évaluer l'erreur de classification au moyen de la matrice de confusion.
b) Identifier les instances mal classées.

Par exemple, pour la matrice de confusion, vous pouvez utiliser les instructions Matlab suivantes :

```
Ycalculee=mlpfwd(Net,Iris_input_centree_reduit);
classe_t=Iris_cls;
[max_y classe_y]=max(Ycalculee');
classe_y=classe_y';
exemples_mal_classes=find(classe_t~=classe_y)
NetClass=zeros(150,3);
NetClass(find(classe_y==1),1)=1;
NetClass(find(classe_y==2),2)=1;
NetClass(find(classe_y==3),3)=1;
conffig(NetClass,Iris_output);
```

(→ Compte-rendu 2)

☞ **À rendre au plus tard pour le lundi 11 février 2019**

Dans ce compte rendu, on vous demande de chercher le nombre <<optimal>> de cycles d'apprentissage pour le problème **"sunspot"**.

1. Utiliser un réseau de neurones avec **une couche cachée**.
2. Utiliser une fonction d'activation linéaire (**'linear'**) à la couche de sortie.
3. Utiliser l'algorithme d'apprentissage **'scg'**.
4. Utiliser **10** puis **3** neurones cachés.
5. Faire varier le **nombre de cycles d'apprentissage** $\longrightarrow$ **options(14)**

Il faut envoyer par email à **meziane.yacoub@cnam.fr** un petit rapport (**format pdf**) contenant :

1. Les différents résultats obtenus sous forme d'un tableau (comme le tableau ci-dessous).
2. Vos commentaires sur ces résultats.
3. Un dossier compressé contenant vos programmes

☞ Il faut choisir 12 valeurs pour **options(14)** :
→ les 7 valeurs indiquées sur le tableau + 5 valeurs qui vous semblent significatives.

Remarque
→ Pour que les résulats ne dépendent pas des poids initiaux (de l'initialisation des poids)
On peut sauvegarder les poids initiaux
Net=mlp(···
Net_Initial=Net;
Ensuite on peut commencer l'apprentissage avec Net_Initial

Nombre de cycles d'apprentissage	**Nombre de neurones cachés**	**ARV sur** D^{App}	**ARV sur** D^{Val}	**ARV sur** D^{Test}
5	**10**			
10	**10**			
15	**10**			
20	**10**			
25	**10**			
200	**10**			
500	**10**			
5	**3**			
10	**3**			
15	**3**			
20	**3**			
25	**3**			
200	**3**			
500	**3**			