1. Évaluation et sélection des modèles décisionnels (**2 points)**

   Considérons un jeu de données de $N = 1000$ transactions bancaires dont $n = 50$ sont frauduleuses. Chaque transaction est représentée par $p = 10$ variables (heure, montant, type de dépense…).

   - On choisit d'entraîner une SVM pour le problème de classification binaire fraude/non-fraude. La recherche d'hyperparamètres s'effectue sur la grille suivante: noyau linéaire ou gaussien (RBF), régularisation $C = 0.1, 1.0, 10, 100$. En supposant que la méthode de validation croisée utilisée est *leave one-out*, combien de SVM différentes sont entraînées? (**1 point**)
   - Les spécifications métiers sont de détecter automatiquement les transactions possiblement frauduleuses sans jamais en rater. Les transactions identifiées sont ensuite vérifiées par des humains. Doit-on maximiser la précision ou le rappel? Justifier brièvement. (**1 point**)

   Correction :

   - La grille comporte huit combinaisons d'hyperparamètres (4 valeurs de régularisation $\times$ 2 noyaux). Chaque modèle subit une validation *leave one-out*, donc N modèles entraînés. Au total, il y a $8 \times 1000 = 8000$ modèles entraînés (éventuellement $+1$ si l'on considère le modèle final réentraîné sur toutes les données).
   - Présenté de cette façon, on cherche à maximiser le rappel: on veut minimiser les faux négatifs. Les faux positifs (possiblement nombreux) seront filtrés par les humains.

2. Arbres de décision / Forêts aléatoires (**4 points** : 1p+1p+1p+1p)

   (a) Supposons que toutes les feuilles d'un arbre de décision contiennent seulement des éléments avec une seule valeur pour la variable cible. Quel est le principal défaut de ce type de construction ?

   (b) Quelles sont les méthodes qui permettent de gérer le sur-apprentissage pour les arbres de décision ?

   (c) Que se passe-t-il si le nombre d'arbres dans un modèle de type *bagging* ou *forêt aléatoire* est trop grand ? Trop petit ?

   (d) L'importance (le poids) α_m d'un classifieur dans un mélange $G(x) = \text{sign} \sum_{i=1}^{M} \alpha_m G_m(x)$ obtenu par AdaBoost est donnée par la formule :

   $$\alpha_m = \log\left(\frac{1 - e_m}{e_m}\right)$$

   Que représente e_m dans cette formule ? Expliquez ce choix de α_m (commentez par rapport à la valeur de e_m).

Correction : (**2 points** : 1p+1p+1p+1p)

(a) Ce type de construction produit souvent des arbres trop profonds, qui généralisent mal (sur-apprentissage).

(b) Méthodes qui permettent de gérer le sur-apprentissage : élagage, limiter la profondeur de l'arbre, limiter la taille min des feuilles. Dans les trois cas l'effet est de générer des arbres plus petits, mais le résultat est généralement différent suivant la méthode choisie.

(c) Trop grand : si le nombre d'arbres est plus grand qu'un seuil (à déterminer par validation croisée), le temps de traitement sera inutilement long sans aucun bénéfice. On peut aussi avoir du sur-apprentissage si les arbres sont trop profonds, mais cela est causé par la profondeur des arbres.
Trop petit : sous-apprentissage, surtout si les arbres sont peu profonds.

(d) α_m est le taux d'erreurs du classifier G_m pondérées par les poids w_i. Si le taux d'erreurs e_m est petit on est devant un bon classifieur, son poids α_m dans le mélange final sera donc grand. Au contraire, un classifieur qui fait beaucoup d'erreurs, aura moins d'impact (α_m petit). Dans tous les cas, la valeur de e_m doit être plus grande de 0.5 (le classifieur faible doit être meilleur qu'un choix aléatoire) sinon le poids α_m devient négatif.

3. SVM / Méthodes à noyaux (**4 points** : 1p+1p+1p+1p)

(a) Définissez la notion de ***marge*** pour un classifieur linéaire. Pourquoi maximiser la marge est un bon principe pour la classification ?

(b) SVM : comment on gére le cas où les données ne sont pas linéairement séparables ?

(c) Décrivez brièvement les deux méthodes présentées en cours pour réaliser l'estimation du support d'une densité via des méthodes à noyaux.

(d) Montrez que $K(x, y) : \mathbf{R} \times \mathbf{R} \to \mathbf{R}$, $K(x, y) = e^{x+y}$ est un noyau défini positif.

Correction :

(a) La marge est la distance entre la l'hyperplan séparateur et l'exemple d'apprentissage le plus proche de celui ci. Maximiser la marge c'est s'assurer que le séparateur passe au milieux entre les deux classes, dans les but d'avoir une meilleure généralisation.

(b) Pour les données non séparable linéairement on minimise le cout des mauvaises classifications via l'introduction des variables de relâchement (variables ressort). La fonction objectif contient dans ce cas deux termes : un correspondant à la marge et un deuxième lié à l'erreur de classification.

(c) SVDD : estimation de la sphère englobante minimale ; OCVSM : estimation de l'hyperplan le plus éloigné de l'origine.

(d) Voir cours : C'est le noyau conforme correspondant à $\phi(x) = e^x$

4. Entraînement des réseaux de neurones **(4 points)**

   - Préciser l'enjeu principal dans l'algorithme de rétro-propagation du gradient de l'erreur, ainsi que la place de l'étape "forward" et "backward".
   - À quoi un entraînement par "batch" correspond-il ? Quel est son intérêt ?
   - Quel est l'intérêt de la méthode de "momentum" ?

   Correction : (2+1+1)

   - L'enjeu est de calculer le gradient de l'erreur par rapport aux différents paramètres du réseau. Pour cela, on procède récursivement de la fin du réseau vers le début, c'est l'étape backward. Le calcul du gradient va nécessiter le calcul des activations à chaque niveau du réseau, c'est l'étape "forward".
   - On calcule le gradient sur un sous-ensemble d'exemples vs tout le jeu de données. Permet une convergence plus rapide.
   - Permet de contrer les oscillations "parasites" dans la descente de gradient lorsque la matrice Hessienne est mal conditionnée. La mémoire du passé permet de soutenir les gradients faibles dans des sens consistants et de diminuer les gradients forts dans des sens oppposés.

5. Réseaux convolutifs **(3 points)**

   - Donner 2 avantages des réseaux convolutifs par rapports aux réseaux complètement connectés.
   - Un réseau localement connecté contient-il plus ou moins de paramètres par rapport à un réseau convolutif ? Donnez une application où l'utilisation de réseaux localement connectés est pertinente.

   Correction : (1+2)

   - Passage à l'échelle, robustesse/invariance par rapport à des déformations locales, inclusion de l'information spatiale
   - Localement connecté à plus de paramètres. Pertinent lorsqu'on veut un encodage absolu de la position spatiale, par exemple reconnaissance de visages.

6. Réseaux récurrents **(3 points)**

   - Décrire l'entraînement des réseaux récurrents avec la méthode "backpropagation through time " (BPTT). À quoi correspond la profondeur du réseau et quelle difficulté cela pose-il ?
   - Dans quel cas utiliser des réseaux récurrents bi-directionnels ?

Correction : (2+1)

- Cela revient à appliquer l'algorithme de rétro-propagation classique sur le réseau "déplié". La profondeur du réseau correspond à l'intervalle temporel considéré, celle-ci peut donc rapidement devenir très grande - d'où des problèmes de gradients évanescents.
- Modéliser la dépendance temporelle dans les deux sens, utile par exemple en NLP (mais moins pertinent dans les tâche prédictives où on veut prédire le futur à partir du passé).