Solution à l'intra de IFT6266 A99

1. (3 points) *Quelle est la différence fondamentale entre ce qu'on fait dans le domaine des algorithmes d'apprentissage et ce qu'on fait en optimisation?*

   En optimisation on utilise des algorithmes permettant de trouver le minimum (ou maximum) d'une fonction C donnée:
   $$\theta^* = \mathrm{argmin}_\theta C(\theta).$$
   On considère que ce qu'on optimise est bien ce qu'on veut vraiment optimiser.

   Par contre dans le domaine des algorithmes d'apprentissage, on a aussi un problème d'optimisation, mais la fonction C est inconnue. Plus précisément, elle a généralement la forme
   $$\theta^* = \mathrm{argmin}_\theta \int Q(\theta, Z)P(Z)dZ$$
   avec Q connue, P **inconnue**, mais un ensemble D de données tirées de P connu.

2. (2 points) *Nous avons défini* **l' induction** *comme le mécanisme qui nous amène d'un ensemble de données à une* fonction *(dont on voudrait minimiser l'erreur espérée sur tout le domaine des données). Vapnik a défini* **la transduction** *comme un mécanisme alternatif où on cherche seulement à trouver une* réponse *à un point test x donné, en minimisant l'erreur de généralisation espérée à ce point (i.e., on ne cherche pas la fonction $f(x)$ pour tout x, on cherche la valeur $\hat{y}$ associée à un x particulier). Parmi les algorithmes d'apprentissage que nous avons vu en classe, nommez-en qui sont plus apparentés à de la transduction qu'à de l'induction.*

   Les algorithmes non-paramétriques que nous avons vus (K-plus proches voisins, modèles à noyaux ou Parzen windows, histogrammes) peuvent tous être vu comme faisant de la transduction. En effet, dans tous les cas on obtient localement une fonction spécialement pour l'entrée x donnée (une fonction constante en général, mais pas nécessairement) en minimisant un critère de coût empirique **local** (autour de x).

3. (3 points) *Expliquez dans vos mots (pas nécessairement formellement) le concept de* **capacité** *et comment cette notion est reliée à la différence entre* **l'erreur d'apprentissage** *et* **l'erreur de généralisation espérée**.

   La capacité est une mesure de la diversité des fonctions présentes dans un ensemble de fonctions (qui peut être fini ou infini). Pour simplifier on peut dire que la capacité est une mesure de la taille d'un ensemble de fonctions. Par exemple dans le cas d'un classifieur linéaire ou d'une régression linéaire la capacité est égale au nombre de paramètres libres. Dans le cas d'un ensemble fini de fonctions la capacité est bornée par le logarithme de nombre de fonctions. Plus la capacité est grande est plus petite sera l'erreur d'entrainement. L'erreur de généralisation espérée commence par décroitre quand on augmente la capacité à partir de 0, atteind un minimum (à la capacité dite optimale), et ensuite remonte (phénomène "d'overfitting").

4. (3 points) *Qu'est-ce que le* **"dilemme biais-variance"***? vous pouvez l'expliquer en mots ou bien en formules (et un dessin pourrait aussi aider).*

   L'erreur quadratique de généralisation peut être décomposée en trois termes: bruit + biais + variance. Soit $\mathcal{F}$ notre ensemble de fonctions, $f(D)$ la fonction obtenu par apprentissage sur un échantillon D, et $F(x) = E[Y|x]$ la fonction qui minimise l'erreur quadratique. Le bruit, $E[(F(X) - Y)^2|X]$ est la variance conditionnelle de la variable de sortie Y et ne dépend pas de l'apprentissage. Le biais, $E[(F(X) - E[f(D)](X))^2]$, dépend de la "distance" entre l'ensemble $\mathcal{F}$ et la fonction idéale F. Plus la capacité de $\mathcal{F}$ est grande est plus le biais devrait être petit. Le biais est du au fait que $\mathcal{F}$ en général ne contient pas F. La variance, $E[E[f(D)](X) - f(D)(X)]$, est due à la taille finie de l'échantillon D: dépendant de D on trouvera un $f(D)$ différent, et d'autant plus différent (en probabilité) que $\mathcal{F}$ est grand. Donc la variance augmente avec la capacité de $\mathcal{F}$. Le dilemme tient au fait qu'une augmentation de la capacité a un effet contraire sur le biais et la variance. Il existe un compromis idéal justement quand la capacité est optimale.

5. (2 points) *Supposons que nous connaissions le modèle $P(Y|X)$ de la* **probabilité conditionnelle de classes** *discrètes Y associées à un vecteur d'entrée X pour faire de la reconnaissance des formes. Prouvez que si on doit choisir une classe, celle qui minimise l'erreur de classification espérée est celle qui a la plus grande probabilité conditionnelle.*

   Si P est juste, alors la probabilité d'erreur si on choisit la classe Y est 1 moins la probabilité de donner la réponse correcte, qui est $P(Y|X)$.

   Si on veut choisir Y qui minimise la probabilité d'erreur, on choisit donc

   $$C = \text{argmin}_Y(1 - P(Y|X)) = \text{argmax}_Y P(Y|X)$$

6. (4 points) *Supposons qu'on vous donne un ensemble de données de taille fixe l, et que vous* **devez** *construire un* **classifieur linéaire**. *Vous avez au départ des entrées dans $\mathcal{R}^n$, mais vous trouvez que vous n'arrivez pas à classifier suffisamment bien les exemples,* aussi bien à l'entrainement qu'au test hors-échantillon (les deux erreurs sont très élevées et comparables). *Un de vos collaborateurs vous présente alors deux options: (a) éliminer certaines des variables d'entrée, (b) rajouter des nouvelles variables d'entrée qui seraient des fonctions fixes des variables existantes (e.g. des produits). Parmi ces deux, quelle approche suivriez-vous, et pourquoi?*

   Puisque l'erreur d'apprentissage et l'erreur de test sont toutes deux élévées, cela veut dire que nous somme dans le régime ou la capacité est bien au-dessous de la capacité optimale (voir dessin fait en classe et dans les notes). [voir discussion page précedente sur la relation entre capacité, erreur d'apprentissage et erreur de généralisation].

   Il faut donc augmenter la capacité, donc choisir (b) (qui rajoute des paramètres libres), puisque (a) la ferait plutot baisser (enlever des paramètres libres).

7. (3 points) *Nous avons vu qu'on peut améliorer la généralisation obtenue d'une régression linéaire en rajoutant un terme de pénalité proportionnel à la norme des paramètres. On peut faire de même pour des réseaux de neurones. Supposons qu'on vous suggère d'utiliser un mécanisme semblable pour spéficiquement réduire le nombre d'entrées qui ont effectivement un impact significatif sur la sortie (sans faire une recherche combinatoire dans l'espace des sous-ensembles d'entrées). Comment pourriez vous utiliser un mécanisme de pénalisation pour y arriver?*

   Soit w_{ij} les poids de la **première couche** du réseau de neurones (qui multiplient donc les entrées). Si beaucoup d'entre eux étaient 0 ou près de 0, on utiliserait moins les entrées (en particulier si tous les poids associés à une entrée sont 0, c'est comme si cette entrée était inexistante). On peut donc simplement rajouter à notre critère d'apprentissage une fonction de pénalité telle que

   $$\lambda \sum_{ij} w_{ij}^2$$

ou bien

$$\lambda \sum_{ij} |w_{ij}|.$$

8. (3 points) *Considérons un problème de* **classification binaire**, *en utilisant un réseau de neurones, entrainé par le critère de moindre carrés, avec une sortie désirée $Y = 1$ quand la classe correcte est 1, et une sortie désirée $Y = 0$ quand la classe correcte est 0, pour une entrée X. Montrez que la fonction qui minimise l'erreur quadratique en espérance (sur les données) est effectivement la distribution conditionnelle $f(x) = P(Y = 1|X = x)$.*

   Nous avons déjà montré (en TPs) que la fonction $f(x) = E[Y|X = x]$ est celle qui minimise l'erreur quadratique. Or dans notre cas Y peut seulement prendre les valeurs 0 et 1, donc l'espérance est la somme pondérée

$$f(x) = E[Y|X = x] = 1 \times P(Y = 1|X = x) + 0 \times P(Y = 1|X = x) = P(Y = 1|X = x)$$

9. (3 points) *Suggérez une méthode pour choisir le nombre de prototypes ("clusters") qu'un algorithme de clustering (apprentissage non-supervisé) comme K-means devrait chercher. N.B. l'erreur de reconstruction (qui est minimisée par K-means) ne serait pas appropriée car plus on a de prototypes et plus basse sera l'erreur de reconstruction aussi bien sur un ensemble d'apprentissage qu'un ensemble de test.*

   Beaucoup ont répondu avec justesse que pour comparer des solutions obtenue avec des K différents, il faudrait rajouter un terme proportionnel à K à l'erreur de reconstruction. Cependant, comment choisir la forme de ce terme et la force de la pénalité? Une solution est de considérer que l'algorithme essaie en fait de modéliser la distribution des données: on peut associer les clusters obtenus aux centres des Gaussiennes d'une mixture de Gaussiennes. On peut ensuite associer à la mixture de Gaussiennes la **log-vraisemblance** hors-échantillon obtenue sur des données non-utilisées pour choisir les paramètres: il s'agit d'une mesure de performance sur laquelle la notion de généralisation s'applique sans problème. Le nombre de clusters s'apparente à une mesure de capacité. S'il est trop grand la log-vraisemblance hors-échantillon ne sera pas très bonne. On choisira donc K qui minimise la log-vraisemblance hors-échantillon (estimée sur un ensemble de test).

10. (4 points) *On vous donne un ensemble de données avec des entrées X et des sorties Y et on vous demande d'appliquer les techniques de réseaux de neurones pour pouvoir prédire le genre de valeur qu'on attend de Y pour un X donné. Les X représentent l'image d'une route vue par un conducteur et les Y représente une décision de tourner plus ou moins à droite ou à gauche pour rester sur la route (e.g. Y représente un angle entre -90° et 90°). Cependant, on vous avertit que l'on s'attend à ce que, pour un X donné, il existe plusieurs valeurs de Y compatibles avec ce X (à cause des intersections et autres embranchements de la route), mais que la valeur intermédiaire de ces Y n'est pas du tout désirable. Le but est* **d'entrainer un robot à conduire une voiture** *(en apprenant à imiter les choix d'un conducteur humain). Vous ne devriez pas utiliser le critère de moindres carrés car le robot pourrait choisir d'aller tout droit (et donc de foncer dans un mur) quand il devrait en fait aller plutôt à droite ou plutôt à gauche (e.g. une route avec un embranchement en "Y"). Pendant l'entrainement, on pourrait avoir pour la même image parfois la réponse "a gauche" (e.g. $Y = -45^o$) et parfois la réponse "à droite" (e.g. $Y = 45^o$), mais il ne faut pas répondre entre les deux (e.g., tout droit $Y = 0^o$). Or, le critère de moindre carrés nous amène à chercher $E[Y|X]$ (qui est donc la moyenne entre les différentes réponses possibles pour un X donné). Comment sortir de ce pétrin? Suggestion: considérez comme critère la maximisation de la vraisemblance conditionnelle de $P(Y|X)$ selon un certain modèle, et utilisez un critère autre que celui qui associe à X une simple normale pour Y (i.e., celui qui serait équivalent au critère de moindres carrés). Expliquez dans les grandes lignes comment calculer le gradient pour entrainer le modèle, et comment interpréter les sorties pour éviter de foncer dans un mur.*

    Il y a au moins deux types de bonnes réponses:

    (a) Voir le problème comme un problème de classifiation, i.e., discrétriser Y (e.g. en 10 valeurs), et apprendre le modèle $P(Y|X)$ qui maximise la vraisemblance conditionnelle (par exemple en utilisant la fonction softmax à la sortie d'un réseau de neurones, tel que vu dans le TP2, avec une sortie par classe, e.g. 10 sorties). Pour l'utilisation, il suffit de choisir la classe la plus probable. Pour l'entrainement on calcule le gradient de la log-vraisemblance conditionnelle

$$\sum_t \log P_\theta(y_t|x_t)$$

par rapport aux paramètres θ.

(b) Voir le problème comme un problème d'estimation de densité conditionnelle, en sachant que la distribution conditionnelle de Y étant donné X peut être **multi-modale**. Par exemple on peut donc utiliser pour $P(Y|X)$ une mixture de au moins 2 Gaussiennes. Il faudra donc que le réseau de neurones produisent en sortie les paramètres de cette mixture (soit les centres des deux Gaussiennes, et possiblement leurs variances et leurs poids). On procède ensuite comme d'habitude pour l'estimation des paramètres en utilisant le gradient de la log-vraisemblance, comme ci-haut. Notez qu'on ne pourra pas utiliser l'algorithme EM car même quand on suppose connue la Gaussienne génératrice pour chaque exemple, on ne sait pas maximiser analytiquement pour obtenir les nouveaux paramètres.