Cours IFT6266,

Concepts importants du chapitre 1 du livre de Bishop

- Fonction de classification.
- Apprendre une fonction de classification.
- Surface de décision.
- Erreur quadratique.
- Fonction de régression.
- Ensemble d'apprentissage.
- Pré-traitement des données.
- Le fléau de la dimensionalité.
- Classes de fonctions polynomiales.
- Apprentissage supervisé, non-supervisé, par renforcement, *semi-supervisé*.
- Généralisation.
- Sur-apprentissage (overfitting) et sous-apprentissage (underfitting).
- Complexité d'un modèle. Complexité effective.
- Régularisation. Fonction de coût régularisée.
- Optimisation facile avec utilisation inefficace des paramètres versus optimisation difficile avec utilisation efficace des paramètres (Barron 93).
- Théorème de Bayes et classificateur optimal de Bayes. Prior et posterior.
- Inférence (apprentissage du modèle) versus décision (utiliser le modèle).
- Statistiques fréquentistes vs Bayesiennes.
- Mélange de densités.
- Fonctions discriminantes.
- Minimisation du risque empirique.
- Seuil de rejet.

Quelques détails supplémentaires sur ces concepts

Notation et notions de base:

- Vecteur d'entrée $x = (x_1, \ldots, x_p)$
- Variable de sortie ou sortie désirée y
- Prédiction $\hat{y}(x)$ ou $f(x)$
- Exemple d'apprentissage (x, y)
- Ensemble d'apprentissage $D = \{z_1, \ldots, z_n\}$, e.g. $z_i = (x_i, y_i)$ (ici x_i veut dire i-eme exemple, qui peut être un vecteur d'éléments x_{ij}).
- Fonction discriminante: $g(x)$ fonction continue de x à partir de laquelle on peut former une fonction discrète pour la classification, e.g. $f(x) = signe(g(x))$. $g(x)$ indique l'éloignement à la **surface de décision** (et de quel côté on est).

Régression et prédicteur affine:

- Erreur de classification: $1_{f(x) \neq y}$
- Fonction de coût quadratique: $(f(x) - y)^2$.
- Régression: estimateur de $E[Y|X]$.
- On peut montrer que $E[Y|X]$ est le f qui minimise $E[(f(X) - Y)^2]$.
- Régression linéaire: on choisit notre estimateur dans la classe des prédicteurs affines, $f(x) = w.x + b$, avec $\theta = (w, b)$ libre.
- Régression linéaire ordinaire: on choisit θ qui minimise l'erreur quadratique moyenne.
- Régression linéaire sur une base non-linéaire: on remplace x par $\phi(x)$. Dans le cas le plus simple ϕ est fixe (e.g. base des polynômes d'ordre $\leq d$).
- Pour simplifier on va incorporer le b dans w en ajoutant une dimension (1) à x ou $\phi(x)$.

- Représentation matricielle du problème: l'ensemble d'apprentissage en matrices donne $\mathbf{X}$ $(n \times p)$, avec i-eme rangée $= x_i$, et $\mathbf{Y}$ $(n \times 1)$. L'erreur quadratique totale est alors $C = \sum_i (y_i - \hat{y}_i)^2 = (\mathbf{Y} - \mathbf{X}w)'(\mathbf{Y} - \mathbf{X}w)$ et son minimum satisfait $\frac{\partial C}{\partial w} = 0$, ce qui donne $\frac{\partial C}{\partial w} = 2\mathbf{X}'(\mathbf{Y} - \mathbf{X}w)$
  On résoud:
  $$\hat{w} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$$

- Les prédicteurs affines sont en général trop restreints, mais on peut augmenter la richesse de cette classe en considérant $f(x) = w.\phi(x) + b$, avec $\phi(x)$ une fonction vectorielle suffisamment riche (e.g. prédicteurs polynomiaux).

- Si on choisit ϕ trop riche, on peut facilement obtenir de la sur-généralisation (overfitting). On peut éviter ce problème soit en contrôlant la richesse des $\phi_i(.)$ ou bien en pénalisant les $w =$ régularisation (e.g. on préfère un petit nombre de valeurs de w_i qui soit non-nulles, ou plus généralement $||w||_p$ petit). La richesse de la classe de fonctions (ou le degré de régularisation) doit être choisie en fonction du nombre de données. Si on choisit une classe trop restreinte on a de la sous-généralisation (underfitting).

- Avec $\phi(.)$ polynomial on a besoin de beaucoup de paramètres pour pouvoir représenter une fonction suffisamment intéressante, mais l'optimisation des paramètres est facile (convexe). Par contre si on restreint la dimensionalité de ϕ mais on permet d'apprendre certains paramètres de ϕ (réseau de neurones) on a une utilisation efficace des paramètres (Barron 93) au prix d'une optimisation plus difficile (non-convexe ou bien NP-dur).

Régression et classification non-paramétrique: K-plus-proches-voisins

- $$\hat{y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

  où $N_k(x)$ est l'ensemble des k $x_i \in D$ qui sont les plus proches de x selon une distance $d(x, x_i)$ (habituellement la distance Euclidienne).

- KPPV = Modèle constant local: moyenne des y_i des k voisins de x_i.

- Tesselation de Voronoï avec $k = 1$.

- Zero erreur de classification sur D avec $k = 1$. Augmenter k ne diminue pas l'erreur sur D mais peut améliorer la généralisation (espérance de la perte). Existence d'un k intermédiaire optimal. Ce k optimal dépend de la loi (inconnue) des données et de la quantité de données.

- On ne peut utiliser l'erreur d'apprentissage pour choisir k: k a une nature différente de w pour la régresssion linéaire. C'est un **hyper-paramètre**: il contrôle la richesse de la solution.

- **Fléau de la dimensionalité**:

  - Avec les méthodes non-paramétriques on caractérise une "région" avec les exemples d'apprentissages qui tombent dedans. Le nombre de "régions" avec un nombre minimal d'exemples par région augmente exponentiellement avec p, la dimension de chaque vecteur d'exemple.
  - Notre intuition ne fonctionne pas bien en haute dimension, p grand.
  - Soit $x_i \sim U[-1, 1]^p$ et point test x. Un hypercube autour de x avec fraction r des n exemples aura un côté de taille $r^{1/p}$. Approche 1 rapidement pour p grand $\rightarrow$ voisinage immense.
  - Presque tous les exemples sont sur les bords.

- **Biais et variance dans l'espace des fonctions**: soit $f(x)$ la solution optimale $E[Y|x]$ et $\hat{y}(x)$ notre estimateur, qui dépend des données D.
  bruit $= E[(f(x) - y)^2]$
  biais en $x = f(x) - E_D[\hat{y}(x)]$
  biais carré total $= E[f(x) - E_D[\hat{y}(x)]]^2$
  variance en $x = E[(\hat{y}(x) - E_D[\hat{y}(x)])^2]$
  variance $= E[(\hat{y}(x) - E[\hat{y}(x)])^2]$
  où les espérances incluent les données D. On peut parfois les calculer sous certaines hypothèses distributionnelles. On peut s'intéresser au biais et à la variance en un point x particulier. Pour simplifier les calculs on va aussi parfois considérer seulement le biais et la variance en gardant les x_i d'apprentissage fixe (donc conditionnels aux intrants d'apprentissage).

Classificateur de Bayes

- Théorème de Bayes. Prior et posterior. Voir notes sur les concepts préliminaires.

- Classificateur optimal de Bayes: choisir la classe y qui maximise $P(Y = y|X = x)$ minimise le nombre d'erreur de classification espérées. Mais si certaines erreurs coûtent plus cher que d'autre on fait une autre décision,

qui peut être obtenue à partir de $P(Y|X)$ et de la matrice des coûts de classification (C_{ij} = coût si on choisit la classe i quand la classe j est correcte).

- Classificateur de Bayes (classificateur via estimation de densité): on estime $P(Y = y|X = x)$ via la densité conditionnelle $P_X(x|Y = y)$, avec $\frac{P_X(x|Y=y)P(Y=y)}{\sum_y P_X(x|Y=y)P(Y=y)}$.
- Exemple du classificateur de Bayes Gaussien: on a une densité Gaussienne pour chaque classe $Y = y$, i.e. $P_X(x|Y = y)$ est normale en x). Surface de décision quadratique ou linéaire.
- Inférence (apprentissage du modèle) versus décision (utiliser le modèle). Peut se faire en deux étapes séparées (modèles probabilistes) ou en une étapes (modèles où l'on apprend directement une fonction de décision).
- Statistiques fréquentistes vs Bayesiennes.

Intro à la régression logistique

- Un classifieur linéaire probabiliste. On apprend $P(Y|X)$ avec Y discret (une classe). Dans le cas binaire

$$P(Y = 1|x) = \frac{1}{1 + e^{-w \cdot x - b}} = sigmoide(w \cdot x + b)$$

(avec $P(Y = 0|x) = 1 - P(Y = 1|x)$), avec $x \in \mathbb{R}^d$, $w \in \mathbb{R}^d$, $b \in \mathbb{R}$.

- Dans le cas à plus que 2 classes on utilise la fonction *softmax*, ($\mathbb{R}^q \mapsto \mathbb{R}^q$) ce qui nous donne le vecteur de probabilités

$$P(Y = .|x) = softmax(Wx + b)$$

où si $u = softmax(v)$ alors

$$u_i = \frac{e^{v_i}}{\sum_i e^{v_i}}.$$

On estime les paramètres du modèle en utilisant la log-vraisemblance conditionnelle ($\sum_t \log P(y_t|x_t)$) comme critère, possiblement régularisé (voir ci-bas).

Régularisation

- Si notre classe de fonctions est trop riche on risque la sur-généralisation.

- La situation est particulièrement grave dans le contexte non-paramétrique où on n'impose pas nécessairement une forme particulière à notre solution. Si on ne met pas de contrainte sur les fonctions solutions on obtient facilement l'erreur quadratique totale $= 0$ avec $f(x_i) = y_i$. Infinité de solutions. Laquelle choisir? certaines sont très mauvaises (e.g. $f(x)$=constante arbitraire sauf sur D).

- On peut se restreindre à un ensemble de fonctions plus petit.

- Et/ou ordonner les solutions selon une préférence à priori (solutions plus "simples"). Il faut donc définir une notion de préférence entre les solutions possibles (un terme de régularisation).

- Estimateur non-paramétrique: la classe de solutions est très vaste mais on impose une préférence (e.g. k grand pour KPPV = fonctions qui varient moins souvent).

- Estimateur paramétrique: on impose une classe restreinte de fonctions (e.g. fonctions linéaires). $\mathcal{F} = \{f : f = f_\theta(.)\}$.

- On formalise la notion de préférence à priori:

$$\min_f \sum_i L(f, z_i) \quad + \quad \lambda J(f)$$

  où λ un scalaire qui contrôle le niveau de pénalité et J une fonctionnelle qui ne dépend pas des données et qui représente la fonction de préférence.

- Exemple: les splines, $J(f) = \int |f''(x)|dx$ donne lieu à $f(x)$ cubique par parties.

- Autre exemple: régression logistique ou régression ordinaire, pour lesquelles on pénalisera souvent la solution par $\lambda||\theta||^2$ (avec θ le vecteur de paramètres).