La régression quantile

Et comment l'améliorer avec des réseaux de neurones

Adam Salvail

Adam.Salvail@USherbrooke.ca
BISOUS / SMART
Département d'informatique
Université de Sherbrooke

27 février 2015

Sommaire de la présentation

La régression en apprentissage automatique

Deux grands domaines de l'apprentissage automatique :

- la classification ;
- la régression.

Que recherche-t-on en régression ?

- La moyenne (distance L_2) ;
- La médiane (distance L_1) ;
- Les paramètres d'une distribution.

Citation

Henri Poincaré aurait dit :

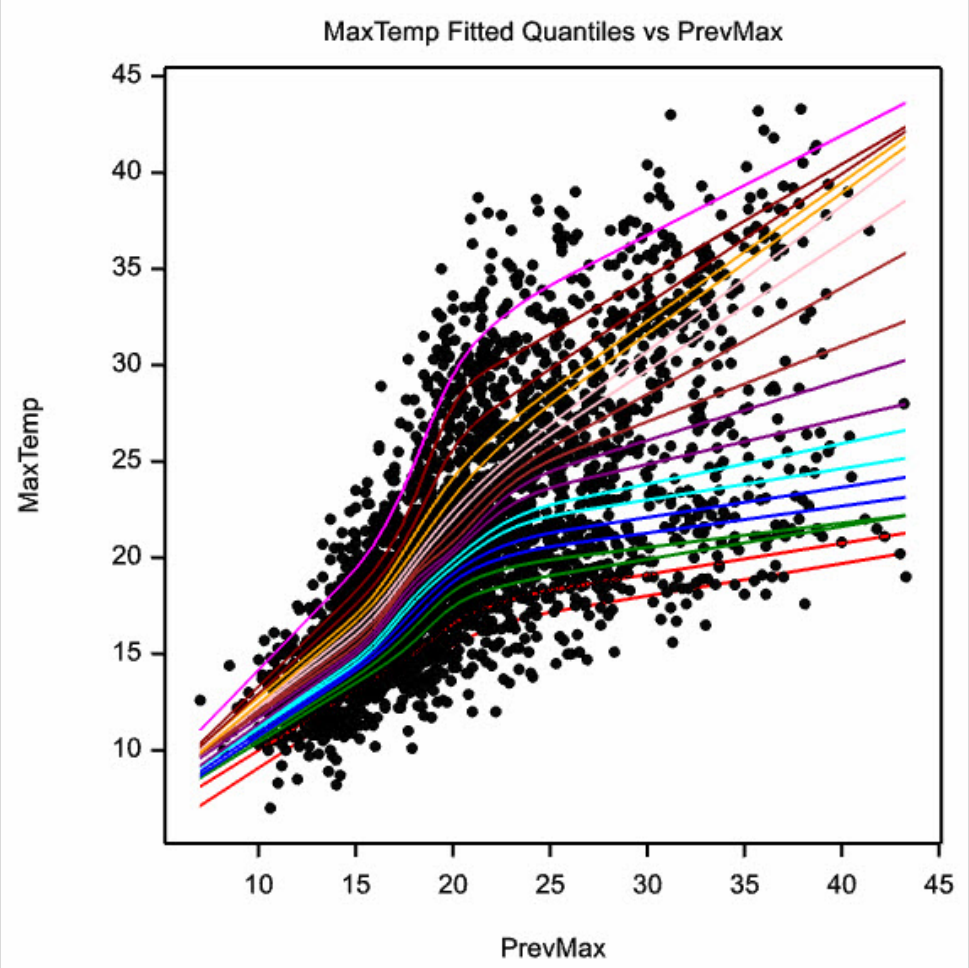
≪ Everyone believes in the [Gaussian] law of errors, the experimenters because they think it is a mathematical theorem, the mathematicians because they think it is an experimental fact. ≫

La régression quantile

Roger Koenker et Gilbert Bassett Jr. (1978) apporte une alternative : la régression quantile.

- C'est une méthode non-paramétrique ;
- Peut facilement donner une prédiction de la médiane ;
- Peut également donner de une idée générale de la forme de la distribution.

Exemple de régression quantile



Motivation : Récentes avancées en réseaux de neurones

Depuis 2006, les réseaux de neurones ont fait un retour en force !

- Initialization judicieuse des paramètres ;
- Meilleure régularisation (Dropout, Maxout) ;
- Innovation au niveau des fonctions d'activation ;
- Meilleurs algorithmes d'optimisation stochastique.

La régression usuelle

Nous étudions un phénomène modélisé par une distribution D. Soit $(\mathbf{X}, \mathbf{Y}) \sim D$ un échantillon de D où $\mathbf{x} \in \mathbf{X}$ est un vecteur de variables explicatives et $y \in \mathbf{Y}$ la valeur à prédire.

$$\mathbb{E}\left[\mathbb{P}(y|\mathbf{x})\right] \approx f_{\boldsymbol{\theta}^*}(\mathbf{x}) \Leftrightarrow \boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \frac{1}{|\mathbf{X}|} \sum_{\mathbf{x} \in \mathbf{X}} L_2(f_{\boldsymbol{\theta}}(\mathbf{x}), y)$$

$$\text{Médiane}\left[\mathbb{P}(y|\mathbf{x})\right] \approx f_{\boldsymbol{\theta}^*}(\mathbf{x}) \Leftrightarrow \boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \frac{1}{|\mathbf{X}|} \sum_{\mathbf{x} \in \mathbf{X}} L_1(f_{\boldsymbol{\theta}}(\mathbf{x}), y)$$

Quantiles

Un quantile Q_τ se définit comme suit :

$$Q_\tau(D) = \inf\{d : F_D(d) \geq \tau\}$$

où $F_D(d) = \mathbb{P}(D \leq d)$ est la fonction de répartition de D.

Notez que $Q_{0.50}(D)$ est la médiane de D.

Exemple d'utilisation des quantiles

En finance, une utilisation intéressante des quantiles (et de la régression quantile) est donnée par la valeur à risque.

$$\mathrm{VaR}_\tau(L) = Q_{1-\tau}(L)$$

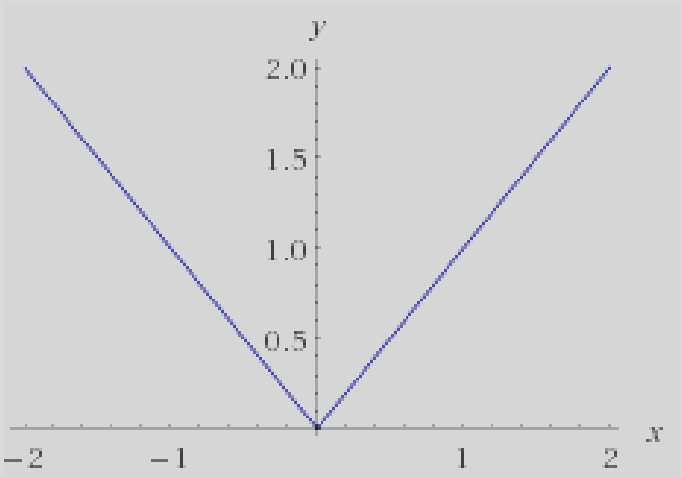
Cette valeur représente le seuil que la perte monétaire encourrue dépassera avec probabilité τ.

La fonction de perte machine à boule

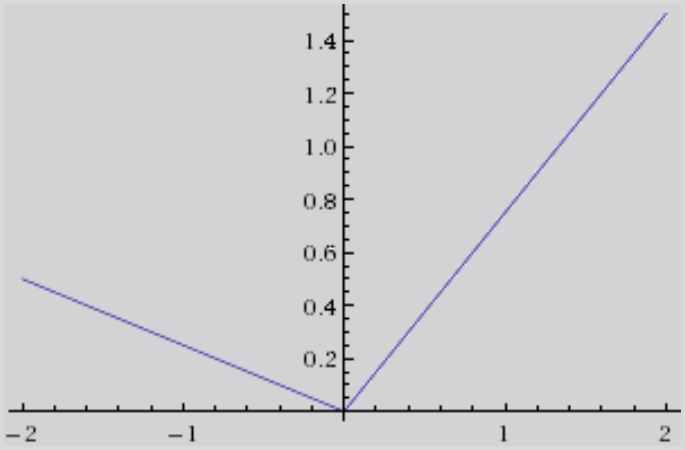
Dans leur travail séminal de 1978, Koenker et Bassett Jr. ont introduit la *pinball loss function*, librement traduite par la fonction de perte machine à boule.
On la dénote L_τ pour $\tau \in (0, 1)$.

$$L_\tau = \begin{cases} \tau x & \text{si } x \geq 0 \\ (\tau - 1)x & \text{sinon.} \end{cases}$$

Exemple de la fonction de perte machine à boule



La perte L_1.



La perte $L_{0.25}$.

Preuve de la convergence vers les quantiles

Soit $d \sim D$. On cherche $Q_\tau(D)$. On montre que :

$$Q_\tau(D) = \arg\min_x \mathbb{E}_D[L_\tau(d - x)].$$

Le membre de droite se réécrit

$$\arg\min_x (\tau - 1) \int_{-\infty}^{x} d - x \, \mathrm{d}F_D(d) + \tau \int_{x}^{\infty} d - x \, \mathrm{d}F_D(d).$$

En prenant la dérivé par rapport à x, on retrouve

$$-(\tau - 1) \int_{-\infty}^{x} \mathrm{d}F_D(d) - \tau \int_{x}^{\infty} \mathrm{d}F_D(d) = F_D(x) - \tau.$$

Preuve de la convergence vers les quantiles (suite)

Supposons que $x^* = Q_\tau$. Alors,

$$F_D(Q_\tau) = \tau.$$

Donc, Q_τ est un point stationnaire de $L_\tau(d - x)$.

Comme la dérivée de $F_D(x)$ est la fonction de densité de D, elle est toujours positive. Par conséquent, Q_τ est vraiment la valeur qui minimise la fonction de perte machine à boule.

Version linéaire de la régression quantile

Le modèle original introduit par Koenker et Bassett tentait de trouver

$$\beta^* = \arg\min_{\beta \in \mathbb{R}^k} \mathbb{E}[L_\tau(Y - X\beta)]$$

Pour y arriver, ils reformulaient le problème comme un programme linéaire qui pouvait ensuite être résolu par simplexe.

Versions non-linéaire de la régression quantile

Voici quelques travaux tentant d'utiliser un modèle non-linéaire :

- Taylor (2001) : Utilisation d'un réseau de neurones simple ;
- Koenker et al. (2005) : Modèle non-linéaire utilisant des splines ;
- Takeuchi et al. (2005) : Utilisation de SVM comme modèle ;
- Meinshausen (2006) : Modèle utilisant des forêts d'arbre aléatoire (*Random Forests*) ;
- Boukouvalas et al. (2012) : Utilisation de processus gaussiens.

Réseau de neurones

Soit $(\mathbf{X}, \mathbf{Y})$ un échantillon de notre distribution D.
On cherche le meilleur vecteur de paramètres θ pour le modèle $f_\theta(\mathbf{X})$ qui pourra le plus précisément prédire $Q_\tau(D)$.

Un réseau de neurones est une successions de couches qui prennent une entrée, lui applique une transformation linéaire, puis une fonction d'activation non-linéaire. La k-ième couche peut se décrire comme :

$$\mathbf{z}_k = \sigma(\mathbf{W}_k \mathbf{z}_{k-1} + \mathbf{b}_k)$$

où $\mathbf{W}_k, \mathbf{b}_k \in \theta$ sont les paramètres à optimiser et σ_k est une fonction d'activation.
En posant $\mathbf{z}_0 = \mathbf{x}$, on définit le modèle $f_\theta(\mathbf{x}) = \mathbf{z}_n$.

Fonctions d'activations

Le choix de la fonction d'activation s'avère être crucial pour l'entrainement d'un réseau de neurones. Voici quelques choix :

- sigmoïde : $\frac{1}{1+e^{-x}}$;
- tangente hyperbolique : $\tanh(x)$;
- *Rectified Linear Unit* : $\max(0, x)$;
- *Leaky Rectified Linear Unit* : $\max(0, x) + 0.01 \min(0, x)$;
- *Parametrized Rectified Linear Unit* : $\max(0, x) + \alpha \min(0, x)$.

Récents développements

Un article de Microsoft, paru la semaine passée, a utilisé la PReLU sur *ImageNet* pour obtenir des résultats supérieurs à ceux d'un humain !

Dropout

Afin d'éviter le surentrainement, il faut régulariser les paramètres du modèle que l'on entraine. Pour y arriver, plusieurs méthodes s'offrent à nous :

- Ajouter une pénalité $\lambda\|\boldsymbol{\theta}\|$ à la fonction de perte ;
- Ajouter un bruit aléatoire aux entrées : $\mathbf{z}_0 = \mathbf{x} + N(0, \sigma)$;
- Utiliser Dropout (ou Maxout, ou DropConnect).

Récents développements

Un article de Google, paru la semaine passée, a utilisé une technique appelée *batch normalization* afin d'obtenir des résultats semblables à ceux de Microsoft et pouvant remplacer (et de façon plus efficiente) Dropout.

Différentiation automatique et gradients

Pour arriver à optimiser les paramètres du réseau de neurones, on utilise le gradient de la fonction de perte

$$\nabla_{\boldsymbol{\theta}} \left[\frac{1}{|\mathbf{X}|} \sum_{(\mathbf{x},y)\in(\mathbf{X},\mathbf{Y})} L_\tau(f_{\boldsymbol{\theta}}(\mathbf{x}), y) \right].$$

Pour ce faire, la différentiation automatique est très utile !
Dans le domaine des réseaux de neurones, la différentiation automatique porte parfois le nom de *backpropagation*.

Problèmes de non-différentiabilité

La fonction de perte machine à boule a un grand défaut : elle est non différentiable en $x = 0$.
Pour contrer le problème, plusieurs solutions :

- utiliser un algorithme robuste aux points de non-différentiabilité ;
- lisser la fonction de perte ;
- faire semblant que le problème n'existe pas, celui-ci n'étant *presque jamais* un problème.

La dernière solution fonctionne étonnament bien !

Entrainement du modèle

Pour entrainer le modèle, on utilise un algorithmes d'optimisation.

SGD La descente de gradient (stochastique). Très simple et couramment utilisé en apprentissage automatique. Elle suit la direction de plus forte descente.

AdaGrad Plus récent, ajuste individuellement chaque dimension du gradient en fonction des gradients antérieurs.

Adam Basé sur `AdaGrad`, `Adam` généralise le concept mis de l'avant par `AdaGrad` de modifier chaque dimension du gradient en prenant également en considération le second moment de la fonction.

L-BFGS Algorithme d'approximation de la dérivée seconde de la fonction objectif. Utilise l'information ainsi approximer pour corriger la direction dans laquelle faire un pas.

Évaluation d'un modèle

Il existe quelques problèmes liés à la régression quantile.

- Contrairement à la régression usuelle, la perte optimale n'est pas zéro.
- Pire, on ne connait pas la vraie valeur des quantiles de la distribution conditionnelle.

Pour évaluer la performance de notre prédicteur de quantiles, on utilise donc deux critères :

- La fonction de perte machine à boule ;
- La *ramp loss* qui calcul la distance L_1 moyenne entre les quantiles marginaux de $\mathbf{Y}$ et de $f_{\theta}(\mathbf{X})$.

Expériences et résultats

Plusieurs tests de comparaison :

- **Modèle** `QRANN`, `quantregForest`, `kqr`, `qrnn` ;
- **Données** Boston housing, Communities and crimes, Power plant output, Million songs dataset, South Australia Electricity Demand et Wine quality.

Pour `QRANN`, nous avons également testé différentes combinaisons de topologie, fonction d'activation, lissage, régularisation et algorithme d'optimisation.

Résultats

Malheureusement, toutes ses expériences sont en cours de calcul sur Mammouth. Sur les tests préliminaires, `QRANN` est compétitif ou surpasse ses concurants.

Conclusions et futurs directions de recherche

En conclusion :

- La régression quantile s'avère être un puissant outil de prédiction.
- QRANN est, a priori, une méthode compétitive et moderne de régression quantile.
- Surtout, QRANN est assez efficace pour être entrainé sur de grands jeux de données (Million songs dataset), alors que les performances de ses plus proches compétiteurs se dégradent rapidement avec la taille de l'échantillon.

La prochaine étape : passer à un modèle temporel de régression quantile. Les prédictions du domaine des séries chronologiques profitent vraiment de l'information additionnelle apportée par la régression quantile.