Explication de l'algorithme pour la méthode EC

Rivaldi Tristan

2024-03-01

Explication de l'algorithme pour la méthode EC

Le but de cet algorithme est de calculer T_{EC} pour que la confiance moyenne corresponde à la précision moyenne sur l'ensemble de validation. Pour ce faire, on dispose d'un ensemble de validation (x_i, y_i) pour $i = 1, \dots, n_{\text{val}}$, et d'un classifieur $\widehat{f} : X \to \mathbb{R}^K$ où les $x_i \in \mathbb{R}^p$ sont les caractéristiques (les variables) et les $y_i \in [\![1, K]\!]$ sont les étiquettes de classes associées à ces caractéristiques. L'algorithme calcule les logits $z^{(i)} = f(x_i) \in \mathbb{R}^K$ et produit la sortie $\widehat{y}_i = \arg\max_k z_k^{(i)}$.

Le classifieur prend en entrée des données (ici les caractéristiques x_i extraites d'une observation) et y attribue des logits $z^{(i)} = (z_1^{(i)}, ..., z_K^{(i)})$ où pour tous k appartenant à $[\![1, K]\!]$, chaque $z_k^{(i)} \in \mathbb{R}$. Pour un k fixé $z_k^{(i)}$ correspond au logit (score) associé à la classe k. Le fait de produire la sortie $\widehat{y}_i = \arg\max_k z_k^{(i)}$ signifie que la classe correspondant au logit le plus élevé est choisie comme la prédiction. Pour la classe d'appartenance de x_i, l'algorithme choisit de prédire que la classe associée à x_i est $\widehat{y}_i$.

Ensuite, l'algorithme calcule la précision moyenne sur l'ensemble de validation :

$$A_{val} = \frac{1}{n_{\text{val}}} \sum_i \mathbb{1}(y_i = \widehat{y}_i) \ .$$

Les logits sont des valeurs brutes, résultant de la dernière couche d'un réseau de neurones avant l'application d'une fonction d'activation. Ces valeurs brutes ne sont pas normalisées et peuvent être n'importe quel nombre réel.

Cependant, avant d'obtenir les probabilités associées à chaque classe, les logits passent généralement par une fonction d'activation softmax. La fonction softmax transforme les logits en probabilités, produisant une distribution de probabilité sur les classes. Les valeurs résultantes après

la fonction softmax seront dans l'intervalle [0,1], et leur somme sera égale à 1. La fonction softmax va transformer le vecteur $z^{(i)}$ en un vecteur $z^{(i)'} = \sigma(z^{(i)})$ où :

$$\begin{aligned} \sigma : \mathbb{R}^K &\to \mathbb{R}^K \\ (z_1, \dots, z_K) &\mapsto \left(\frac{e^{z_1}}{\sum_{j=1}^K e^{z_j}}, \dots, \frac{e^{z_K}}{\sum_{j=1}^K e^{z_j}} \right) \quad . \end{aligned}$$

On a que $\sigma(z^{(i)})_k$ est la probabilité telle qu'estimée par le réseau, que x_i appartienne à la classe k. Pour un $x_i \in \mathbb{R}^p$ la prédiction finale du modèle pour prédire quelle est la classe associée à x_i, est alors donner par $\hat{y}_i = \arg\max_k \sigma(z^{(i)})_k$ pour k appartenant à $[\![1, K]\!]$ et la confiance de prédiction associé est définie comme étant: $\max_k \sigma(z^{(i)})_k$. De plus, on a bien :

$$\sum_{k=1}^K \sigma(z^{(i)})_k = \frac{\sum_{k=1}^K e^{z_k^{(i)}}}{\sum_{j=1}^K e^{z_j^{(i)}}} = 1 \quad .$$

Pour trouver T_{EC}, on va en fait prendre T_{EC} tel que :

$$\frac{1}{n_{\text{val}}} \sum_{i=1}^{n_{\text{val}}} \max_{k=1}^K \sigma \left(\frac{z^{(i)}}{T_{\text{EC}}} \right)_k = A_{\text{val}} \quad .$$

De cette manière, T_{EC} permet à ce que la probabilité maximale d'appartenir à une classe après l'application de la fonction softmax soit en accord avec la précision moyenne sur l'ensemble de validation. De cette manière on a bien que la confiance moyenne correspond à la précision moyenne sur l'ensemble de validation.

Dans la pratique

Voici le code pour calculer la température T_{EC}:

(CF. ALSO: https://pytorch.org/torcheval/main/generated/torcheval.metrics.MulticlassAccuracy.html#torchev

```
# Fonction pour trouver la température optimale pour la méthode EC
def opt_temp(logits, labels, T_min=0.01, T_max=10.0):
    validation_error = get_classification_error(logits, labels)

    def objective(T):
        probas = torch.max(torch.softmax(logits / T, dim=1), dim=1)[0]
        return torch.mean(probas) - (1.0 - validation_error)
```

```
    res = optimize.root_scalar(objective, bracket=[T_min, T_max])
    return float(res.root)
```

Dans la pratique, le code qui calcule la température T_{EC} est une fonction qu'on applique a un modèle de réseau de neurones et qui prend en entrée :

- les logits: qui sont les sorties brutes du réseau de neurones avant l'application de la fonction softmax.
- les labels: qui sont les étiquettes réelles associées aux données
- T_{min} qui est la température minimale à considérer (0,01)
- T_{max} qui est la température maximale à considérer (10)

La première étape consiste à calculer l'erreur de classification du modèle à l'aide de la fonction `get_classification_error(logits, labels)`. Cela donne la mesure de la performance actuelle du modèle. Par exemple si elle vaut 0.30 cela veux dire que dans 70% des cas le modèle associe correctement les étiquettes et les données du modèle.

Ensuite, une fonction objectif est définie, notée `objective(T)`, qui prend la température T comme paramètre. À l'intérieur de cette fonction, les logits sont divisés par la température T, puis passés à travers la fonction softmax. On extrait ensuite les probabilités maximales pour chaque exemple avec l'aide de la fonction `torch.max`. La valeur de retour de la fonction objectif est la moyenne de ces probabilités maximales moins la complémentaire de l'erreur de validation c'est-à-dire la précision moyenne sur l'ensemble de validation. On utilise l'optimiseur `optimize.root_scalar` de la bibliothèque `scipy` pour trouver la racine de cette fonction dans l'intervalle spécifié par $[T_{min}, T_{max}]$.

Résultats Obtenus

Nous avons utilisé des modèles pré-entraînés provenant du dépôt Github: https://github.com/chenyaofo/pytorch-cifar-models.

Modèle	Score de Brier	ECE
Modèle non calibré	0.455	0.104
Calibré avec TS	0.436	0.030
Calibré avec TEC	0.436	0.028

Table 1: Résultats obtenus pour CIFAR-100 Resnet20

On peux voir les résultats obtenus : Tableau 1

L'Expected Calibration Error (ECE)

L'Expected Calibration Error (ECE) est une mesure d'évaluation de la calibration d'un modèle de réseau de neurones, particulièrement dans le contexte de la classification probabiliste (cf. Guo et al. (2017)). La calibration se réfère à la justesse des prédictions de probabilité du modèle, c'est-à-dire à quel point les probabilités prédites correspondent aux fréquences réelles des événements.

On dit qu'un algorithme de classification est calibré si la probabilité prédite $\hat{p}$, correspond à la probabilité réelle que la prédiction soit bonne. Ce qui revient mathématiquement à:

$$P(\hat{y} = y|\hat{p} = p) = p, \quad \forall p \in [0; 1]$$

Où $\hat{y}$ est la classe prédite et y est la vraie classe. Dans tous les contextes pratiques, atteindre une calibration parfaite est impossible. L'erreur de calibration est une valeur qui représente la calibration du modèle sur l'ensemble des prédictions. Il s'agit de l'espérance mathématique de la différence entre la réalité et la confiance du modèle. On a donc:

$$ECE = \mathbb{E}[P(\hat{y} = y|\hat{p} = p) - p]_{\hat{p}}$$

On a donc que une valeur faible de l'ECE indique une bonne calibration, tandis qu'une valeur élevée suggère une mauvaise calibration. En effet, une ECE faible indique que le modèle a une tendance à produire des probabilités proches des véritables probabilités d'appartenance à une classe.

La calibration d'un modèle peut être visualisée par un diagramme de fiabilité (reliability diagram). Pour estimer la précision attendue à partir d'échantillons finis de taille N, il faut regrouper les prédictions en M intervalles (chacun de taille $\frac{1}{M}$). On considère l'intervalle $I_m = (\frac{m-1}{M}, \frac{m}{M}]$ et B_m l'ensemble des indices des échantillons dont la confiance de prédiction se situe dans l'intervalle I_m. Pour chaque groupe B_m on calcule l'accuracy qui correspond à la proportion d'échantillons correctement classés et la confiance moyenne:

$$acc(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} 1(y_i = \hat{y}_i) \ ,$$

$$conf(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i \ .$$

Puisqu'il y a un nombre fini M de groupes, on calcule l'erreur de calibration comme suit :

$$ECE = \frac{1}{N} \sum_{m=1}^{M} |B_m||acc(B_m) - conf(B_m)| \ .$$

Score de Brier

Le score de Brier est une fonction de Score qui mesure l'exactitude des prédictions probabilistes. Pour les prédictions unidimensionnelles, elle est strictement équivalente à l'erreur quadratique moyenne aux probabilités prédites.

Définition dans le cas d'une variable binaire :

Considérons une variable à prévoir qui ne peut prendre que deux valeurs (réalisation ou non d'un évènement). Si on dispose d'un ensemble de n prévisions de la probabilité de réalisation de cet évènement et des observations correspondantes, le score de Brier est défini par :

$$B_s = \frac{1}{n}\sum_{i=1}^{n}(p_i - o_i)^2$$

où les p_i sont les probabilités prévues correspondantes à la réalisation de l'évènement, les o_i (déterministes) sont la i-ème observation valant 1 si l'évènement est réalisé et 0 sinon et n le nombre de prévisions.

Définition générale :

Dans le cas où une variable peut prendre plus de 2 valeurs. Le score de Brier est alors défini par :

$$B_s = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}(p_{i,j} - o_{i,j})^2$$

où R est le nombre de classes possibles dans lesquelles l'évènement peut tomber, et N le nombre total d'instances de toutes les classes. $p_{i,j}$ représente la probabilité prédite pour la classe i, $o_{i,j}$ vaut 1 si si la ième observation est de la catégorie j et 0 sinon.

Le score de Brier peut être décomposé en 3 composantes additives : incertitude, fiabilité et résolution

$$B_s = F - R + I$$

I : terme d'incertitude qui prend en compte la dispersion des observations F : terme de fiabilité qui mesure dans quelle circonstances les probabilités prévues sont proches des probabilités réelles compte tenu d'une prévision. R : terme de résolution qui mesure la distance entre les probabilités d'occurence

Interprétation et décomposition :

Plus la valeur du score de Brier sera faible plus la prédiction sera bonne et une prévision parfaite obtiendra un score de 0. A l'inverse le plus mauvais score sera de 1.

On peut aussi décomposer le score de Brier de la façon suivante :

La Regression logistique

La régression logistique est une technique de modélisation statistique utilisée pour prédire la probabilité qu'une variable descriptive binaire prenne l'une des deux valeurs possibles (0 ou 1), que l'on peut noter Y cette variable (elle appartient à $\{0,1\}^n$) en fonction d'un ensemble de variables explicatives que l'on note $X = [\mathbf{x_1}, \dots, \mathbf{x_n}]^\top \in \mathbb{R}^{n \times p}$, avec n observations et p variables. C'est une méthode couramment utilisée en apprentissage automatique et en statistiques.

La régression logistique a une interprétation probabiliste, elle permet de modéliser $P(Y = 1|\mathbf{x})$, où $\mathbf{x} \in \mathbb{R}^p$.

En utilisant la loi de Bayes et le fait que :

$$P(\mathbf{x}) = P(\mathbf{x}|Y = 1)P(Y = 1) + P(\mathbf{x}|Y = 0)P(Y = 0)$$

nous avons :

$$\begin{aligned} P(Y = 1|\mathbf{x}) &= \frac{P(\mathbf{x}|Y = 1)P(Y = 1)}{P(\mathbf{x}|Y = 1)P(Y = 1) + P(\mathbf{x}|Y = 0)P(Y = 0)} \\ &= \frac{1}{1 + \frac{P(\mathbf{x}|Y=0)P(Y=0)}{P(\mathbf{x}|Y=1)P(Y=1)}} \\ &= \frac{1}{1 + \frac{P(Y=0|\mathbf{x})}{P(Y=1|\mathbf{x})}} \end{aligned}$$

On note $f(\mathbf{x}) := \log\left(\frac{P(Y=1|\mathbf{x})}{P(Y=0|\mathbf{x})}\right)$.

On a ainsi $P(Y = 1|\mathbf{x}) =: \sigma(f(\mathbf{x}))$ avec $\sigma(z) = \frac{1}{1+e^{-z}}$.

La fonction σ, appelée fonction logistique, satisfait les propriétés suivantes :

$$\sigma(-z) = 1 - \sigma(z)$$

$$\frac{d\sigma(z)}{dz} = \sigma(z)\sigma(-z)$$

L'intérêt de la fonction logistique réside dans sa capacité à transformer une fonction f à valeurs dans $\mathbb{R}$ en une probabilité comprise entre 0 et 1.

La régression logistique revient en fait à supposer que f est linéaire de la forme $f : \mathbf{x} \mapsto \theta^\top \mathbf{x}$ avec $\theta \in \mathbb{R}^p$.

Sous cette hypothèse, la règle de classification est simplement :

$$\begin{cases} \text{si } \theta^\top \mathbf{x} \leq 0, \text{ on étiquette 0 au point } \mathbf{x} \\ \text{si } \theta^\top \mathbf{x} > 0, \text{ on étiquette 1 au point } \mathbf{x} \end{cases}$$

On obtient donc :

$$P(Y = 1|\mathbf{x}) = \sigma(\theta^\top \mathbf{x})$$

$$P(Y = 0|\mathbf{x}) = 1 - \sigma(\theta^\top \mathbf{x}) = \sigma(-\theta^\top \mathbf{x})$$

Le but maintenant est d'estimer θ. Nous avons $(x_i, y_i)_{1 \leq i \leq n}$ où $x_i \in \mathbb{R}^p$ et $y_i \in \{0, 1\}$ qui constitue un échantillon de taille n. . On a alors :

$$P(Y = y_i|\mathbf{x} = x_i) = \sigma(\theta^\top x_i)^{y_i} \sigma(-\theta^\top x_i)^{1-y_i}$$

La log-vraisemblance s'exprime alors de cette manière :

$$L(\theta) = \sum_{i=1}^{n} \log \left(\sigma(\theta^\top x_i)^{y_i} \sigma(-\theta^\top x_i)^{1-y_i} \right)$$

où l est définie comme étant la fonction de perte logistique. Il faut ensuite avoir recours à des algorithmes itératifs (descente de gradient, méthode de Newton,...) pour trouver $\widehat{\theta}$.

On peut passer du cadre binaire au cadre multi-classes avec K classes, par exemple, c'est-à-dire en ayant Y appartenant à $\{1, 2, ..., K\}$. De nouveau, on modélise les probabilités conditionnelles des classes, ou plutôt leur log-ratio, par des quantités linéaires :

$$\log \left(\frac{P(Y = k|\mathbf{x})}{P(Y = K|\mathbf{x})} \right) = \theta_k^\top \mathbf{x_i}$$

pour $k \in \{1, 2, ..., K - 1\}$ et $\theta_k \in \mathbb{R}^p$.

On a alors pour paramètre global: $\theta \in [\theta_1, ..., \theta_K] \in \mathbb{R}^{p \times K}$ et pour $k \in \{1, 2, ..., K\}$:

$$P(Y = k|\mathbf{x}) = \frac{\exp(\langle \theta_k, \mathbf{x} \rangle)}{\sum_{l=1}^{K} \exp(\langle \theta_l, \mathbf{x} \rangle)} = \frac{\exp(\theta_k^T \mathbf{x})}{\sum_{l=1}^{K} \exp(\theta_l^T \mathbf{x})}$$

On peut écrire cette égalité sous forme vectorielle en utilisant la notation softmax, on a alors :

$$(P(Y = k|\mathbf{x}))_{k=1,...,K} = softmax(\theta_1^T \mathbf{x}, .., \theta_K^T \mathbf{x})$$

Pour la régression softmax, la log-vraisemblance peut être exprimée comme suit :

$$L(\theta) = \sum_{i=1}^{n} \sum_{k=1}^{K} 1(y_i = k) \log \left(\frac{e^{\theta_k^\top x_i}}{\sum_{l=1}^{K} e^{\theta_l^\top x_i}} \right)$$

On utilise ensuite des méthodes algorithmiques pour trouver $\hat{\theta}$.

Bibliographie

Guo, Chuan, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. "On Calibration of Modern Neural Networks." In *International Conference on Machine Learning*, 1321–30. PMLR.