Candidate Sampling

Dong Bingfeng

http://ml.dongbingfeng.cn

Motivation

- A great disparity in count between positive and negative samples with limited computational resources.
  - CTR/CVR prediction models.
  - Disease discrimination models.

- Softmax model with a large number of classifications.
  - Words prediction models in NLP.
  - DNN information retrieval models.

Odds

- Definition

$$odds(p\ vs\ q) = \frac{p}{q}, \quad odds(p) = \frac{p}{1-p}$$

$$logit(p) = log(\frac{p}{1-p}) = logodds(p)$$

- Odds and LR

$$p = \frac{1}{1 + \exp(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x})},$$

$$logodds(p) = logit(p) = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}$$

Sampled Logistic

- We have sampled negative samples with distribution Q(y|x), and wish to obtain P(y|x)

$$\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} = \ logodds(y\ came\ from\ Positive\ vs\ Negative\ \ |\ \boldsymbol{x})$$

$$= log\frac{P(y|\boldsymbol{x})}{Q(y|\boldsymbol{x})\big(1 - P(y|\boldsymbol{x})\big)} = log\frac{P(y|\boldsymbol{x})}{1 - P(y|\boldsymbol{x})} - logQ(y|\boldsymbol{x})$$

With sampling ratio 1/r, Q(y|x) = 1/r

$$F(\boldsymbol{x}, y) = F'(\boldsymbol{x}, y) + logQ(y|\boldsymbol{x}) = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}\ - \log(r)$$

Sampled Softmax

- Softmax **Training** Process

$$p(y|\mathbf{x}) = softmax(\mathbf{x}) = \frac{1}{Z}\exp\left(\mathbf{w}_y^T\mathbf{x}\right), \qquad where\ \ Z = \sum_y \exp(\mathbf{w}_y^T\mathbf{x})$$

$$\nabla \log p(y|\mathbf{x}) = \nabla\mathcal{E}(y) - \sum_{y_k} p(y_k|\mathbf{x})\nabla\mathcal{E}(y_k), \qquad where\ \ \mathcal{E}(y_k) = \mathbf{w}_y^T\mathbf{x}$$

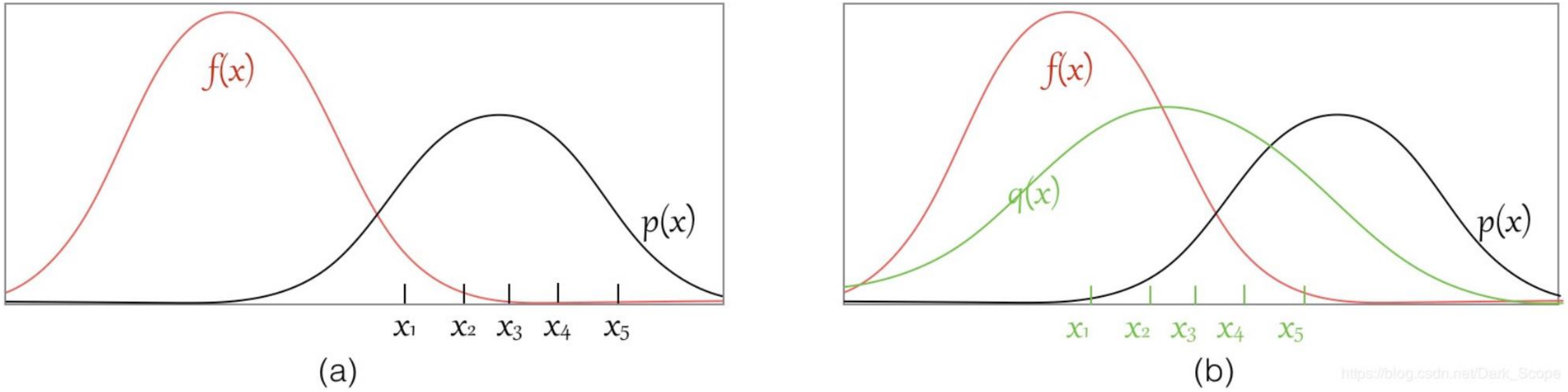
$$\sum_{y_i} p(y_i|\mathbf{x})\nabla\mathcal{E}(y_i) = E_{y_i\sim P}[\nabla\mathcal{E}(y_i)] \simeq \sum_{y_k} \widehat{w}\,(y_k)\nabla\mathcal{E}(y_k)$$

$$where\ \ \widehat{w}(y_k) = \frac{\widetilde{w}\,(y_k)}{\sum_{j=1}^{m}\widetilde{w}\,(y_j)}, \qquad \widetilde{w}\,(y_k) = \tilde{p}(y_k)/\tilde{q}(y_k) = \mathbf{w}_{y_k}^T\mathbf{x}/\tilde{q}(y_k)$$

Importance Sampling

- We need to obtain $E[f(x)], x \sim p$,

i.e. $E[f(x)] = \int_x f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx = \int f(x)w(x)q(x)dx$



$Ef(x) = \sum_{i=1}^{m} \hat{w}(x_i)f(x_i)$ where $\hat{w}(x_i) = \frac{\tilde{w}(x_i)}{\sum_{j=1}^{m}\tilde{w}(x_j)}$ $x_i \sim q$

Noise-Contrastive Estimation (NCE)

- For a statistical model which is specified through an unnormalized pdf $p_m^0(.\,;\alpha)$, we include the normalization constant as another parameter *c* of the model.

i.e. $$\ln p_m(.\,;\theta) = \ln p_m^0(.\,;\alpha) + \mathrm{c}$$

$\mathrm{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ is observed data set; $\mathrm{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_T)$ is an artificially generated data set of noise y with distribution $p_n(.)$

$$\hat{\theta}_T = \underset{\theta}{\operatorname{argmax}} \; J_T(\theta) \quad \text{where}$$

$$J_T(\theta) = \frac{1}{2T} \sum_t \ln\left[h(\mathbf{x}_t;\theta)\right] + \ln\left[1 - h(\mathbf{y}_t;\theta)\right]$$

$$h(\mathbf{u};\theta) = \frac{1}{1+\exp\left[-G(\mathbf{u};\theta)\right]},$$

$$G(\mathbf{u};\theta) = \ln p_m(\mathbf{u};\theta) - \ln p_n(\mathbf{u}).$$

Theory of NCE

- Define pdf of positive samples: $p_d(x_i) = pd_i$; pdf of negative samples: $p_n(x_i) = pn_i$; pdf with parameter θ of samples: $p_m(x_i) = pd_i$; N is the total samples number.
- Probability of observation

$$e^{l(\theta)} = \prod_i \left(\frac{pm_i}{pm_i + pn_i}\right)^{N \cdot pd_i} \left(\frac{pn_i}{pm_i + pn_i}\right)^{N \cdot pn_i}$$

$$\operatorname*{argmax}_\theta l(\theta) = \operatorname*{argmax}_\theta \sum pd_i \ln\left(\frac{pm_i}{pm_i + pn_i}\right) + pn_i \ln\left(\frac{pn_i}{pm_i + pn_i}\right)$$

- Derivative $l(\theta)$ by pm_i, we get $pd_i = pm_i$ when maximize $l(\theta)$.
- Note that there is NO constriction of $\sum pm_i = 1$, so pm_i is normalized by default.

Theory of NCE

- **Theorem**: If conditions (a) to (c) are fulfilled then $\hat{\theta}_T$ converges in probability to θ^*, **i.e.** $\hat{\theta}_T \xrightarrow{P} \theta^*$**.**
  (a) $p_n(.)$ is nonzero whenever $p_d(.)$ is nonzero
  (b) $sup_\theta | J_T(\theta) - J(\theta) | \xrightarrow{P} 0$
  (c) $\mathcal{I} = \int \mathbf{g}(\mathbf{x})\mathbf{g}(\mathbf{x})^T P(\mathbf{x}) p_d(\mathbf{x}) \mathrm{d}\mathbf{x}$ has full rank, where

$$P(\mathbf{x}) = \frac{p_n(\mathbf{x})}{p_d(\mathbf{x}) + p_n(\mathbf{x})}, \quad \mathbf{g}(\mathbf{x}) = \nabla_\theta \ln p_m(\mathbf{x}; \theta)|_{\theta^\star}$$

- Choice of the contrastive noise distribution: Some examples are a Gaussian or uniform distribution, a Gaussian mixture distribution, or an ICA distribution.

Formalized definition

- We have a multiclass problem where each training example $(\mathbf{x}_i, T_i)$ consists of a context $\mathbf{x}_i$, a small set of target classes T_i out of a large universe L of possible classes.

- We wish to learn a compatibility function $F(\mathbf{x}, y)$ which says something about the compatibility of a class y with a context $\mathbf{x}$.

- ***Candidate Sampling*** training methods involve constructing a training task in which for each training example $(\mathbf{x}_i, T_i)$, we only need to evaluate $F(\mathbf{x}, y)$ for a small set of candidate classes $C_i \subset L$, $L = \{T_i\}$.
  Typically, the set of candidates C_i is the union of the target classes with a randomly chosen sample of (other) classes $S_i \subset L$:

$$C_i = T_i \cup S_i$$

The random choice of S_i may or may not depend on $\mathbf{x}_i$ and/or T_i.

Candidate Sampling Algorithms

	Positive training classes associated with training example (x_i, T_i): $POS_i =$	Negative training classes associated with training example (x_i, T_i): $NEG_i =$	Input to Training Loss $G(x,y) =$	Training Loss	$F(x,y)$ gets trained to approximate:
Noise Contrastive Estimation (NCE)	T_i	S_i	$F(x,y) - log(Q(y\|x))$	Logistic	$log(P(y\|x))$
Negative Sampling	T_i	S_i	$F(x,y)$	Logistic	$log\left(\frac{P(y\|x)}{Q(y\|x)}\right)$
Sampled Logistic	T_i	$(S_i - T_i)$	$F(x,y) - log(Q(y\|x))$	Logistic	$logodds(y\|x) = log\left(\frac{P(y\|x)}{1-P(y\|x)}\right)$
Full Logistic	T_i	$(L - T_i)$	$F(x,y)$	Logistic	$log(odds(y\|x)) = log\left(\frac{P(y\|x)}{1-P(y\|x)}\right)$

Candidate Sampling Algorithms

	Positive training classes associated with training example (x_i, T_i): $POS_i =$	Negative training classes associated with training example (x_i, T_i): $NEG_i =$	Input to Training Loss $G(x,y) =$	Training Loss	$F(x,y)$ gets trained to approximate:
Full Softmax	$T_i = \{t_i\}$	$(L - T_i)$	$F(x,y)$	Softmax	$log(P(y\|x)) + K(x)$
Sampled Softmax	$T_i = \{t_i\}$	$(S_i - T_i)$	$F(x,y) - log(Q(y\|x))$	Softmax	$log(P(y\|x)) + K(x)$