Pseudo-Label

Pseudo-Label: The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks

ICML 2013 Workshop

259+ citations

Pseudo-Labels: picking up the class which has the maximum predicted probability, and is used as if they were the true labels.

- Using Pseudo-Labels has same effect as using **Entropy Regularization**.
- ***Entropy Regularization***: The conditional entropy of the class probabilities can be used for a measure of class overlap. By minimizing the entropy for unlabeled data, the overlap of class probability distribution can be reduced. It favors a low-density separation between classes.
- Low density separation between classes is one of the commonly used assumption in semi-supervised learning.
- One of the common way to train a large network is divided into 2 phases:
  - Phase 1: **Unsupervised pre-training**: Learn weights of a network using unsupervised approaches like Auto-Encoder.
  - Phase 2: **Fine-Tuning**: Train the weights of the model using the pre-trained weights received from Phase1 and the labeled dataset available for the given task.
- **Sigmoid Unit**:

$$s(x) = \frac{1}{1 + e^{-x}}$$

  - Activation value of this usually stands for binary probability.
  - We assume that the probability of each label is independent from each other.
  - We use sigmoid output unit in order to make the best use of saturation region of sigmoid.
- **Rectified Linear Unit**:

$$s(x) = \max(0, x)$$

  - Biological plausible more than sigmoid and hyperbolic tangent.
  - **Because rectifier network gives rise to real zeroes of hidden activations and thus truly sparse representations, this unit can boost up the network performance**

Denoising Auto-Encoder:

- Unsupervised learning algorithm
- Used to make the learned representation robust to partial corruption of the input data.

$$h_i = s\left(\sum_{j=1}^{d_v} W_{ij}\tilde{x_j} + b_i\right)$$

$$\hat{x} = s\left(\sum_{j=1}^{d_h} W_{ij}\tilde{h_i} + a_j\right)$$

- Here $\tilde{x}_j$ is the corrupted version of the j^{th} input value, $\hat{x}_j$ is the reconstruction of the j^{th} input value.
- **In Auto-Encoder we reduce the reconstruction error between the input x_j and $\hat{x}_j$**
- ***For Binary input value, the common choice of the reconstruction error is*** $Cross\ Entropy$:

$$L(x, x_j = \sum_{j=1}^{d_v} -x_j log\hat{x}_j - (1-x_j)log(1-\hat{x}_j)$$

Dropout:

- On the network activations, we randomly omit the hidden unit for each example(or minibatch).

$$h_i^k = drop\left(s^k\left(\sum_{j=1}^{d^k} W_{ij}^k h_j^{k-1} + b_i^k\right)\right),\ \ k = 1, 2, \cdots, M$$

- **Dropout is used to reduce *Overfitting* by reducing complex co-adaptations on hidden representations of training data.**
- Dropout is similar to **bagging** since at each iterations we are training a different sub-model.
- Dropout is different from bagging because in dropout the weights of the model are shared but in bagging we have different models and each of them have different weights.

Pseudo-Label

- We pick up the class label which has the maximum predicted probability and assign it for the chosen unlabeled sample.

$$y_i' = \begin{cases} 1 & \text{if } i = \text{argmax}_{i'}\ f_{i'}(x) \\ 0 & \text{otherwise} \end{cases}$$

- The pre-trained network is trained in a supervised fashion with labeled and unlabeled data simultaneously.
- **For unlabeled data, $Pseudo - Labels$ are recalculated for every weight update and are used for the same loss function of supervised learning task.**
- The overall loss function is given by:

$$L = \frac{1}{n}\sum_{m=1}^{n}\sum_{i=1}^{C} L\left(y_i^m, f_i^m\right) + \alpha(t)\frac{1}{n'}\sum_{m=1}^{n'}\sum_{i=1}^{C} L\left(y_i'^m, f_i'^m\right)$$

- here n is the number of mini-batch in labeled data, n' for unlabeled data, f_i^m is the output units if m samples in labeled data, $y^m_i is the label for labeled data of m samples, $f_i'^m$ is the output unit of unlabeled data, $y_i'^m$ is the pseudo-label of unlabeled data, $\alpha(t)$ is a coefficient balancing the supervised loss and unsupervised loss.
- We need to choose proper value for $\alpha(t)$ because:
  - If $\alpha(t)$ is too high, it disturbs training for labeled data.
  - If $\alpha(t)$ is too low, we cannot benefit from unlabeled data.
  - So we perform *Deterministic-Annealing*

Deterministic Annealing:

$$\alpha(t) = \begin{cases} 0 & t < T_1 \\ \frac{t-T_1}{T_2-T_1}\alpha_f & T_1 \leq t < T_2 \\ \alpha_f & T_2 \leq t \end{cases}$$

- We slowly increase the value of $\alpha(t)$ in Deterministic Annealing, because the slow increase is expected to avoid poor local-minima.

Why Pseudo-Labels work?

- ***Low-Density Separation between Classes:***
  - The cluster assumption states that the decision boundary should lie in low-density regions to improve generalization performance.
  - 2 works on training neural networks using manifold-learning
    - **Semi-Supervised Embedding**: Uses embedding-based regularizer to imporve the generalization performance of DNN. Because neighbours of a data sample have similar activations with the sample, by embedding-based penalty term, it's more likely that data samples in high-density region have the same label.
    - **Manifold Tangent Classifier**: Encourages the network output to be insensitive to variations in the directions of low-dimensional manifold.
- ***Entropy Regularization:***
  - used to benefit from unlabled data when using maximum a posteriori esimation.
  - **We can model low density separation between classes without any modeling of the density by minimizing the conditional entropy of class probabilities for unlabeled data**

$$H\left(y|x'\right) = -\frac{1}{n'}\sum_{m=1}^{n'}\sum_{i=1}^{C} P\left(y_i^m = 1|x'^m\right)\log P\left(y_i^m = 1|x'^m\right)$$

  - Here n' is the number of unlabeled data, C is the number of classes, y_i^m is the unknown label for the m^{th} unlabeled sample, x'^m is the input vector of the m^{th} unlabeled sample.
  - ***The entropy is a measure of class overlap. As class overlap decreases, the density of data points get lower at the decision boundary.***
  - The MAP estimate is defined as the maximizer of the posterior distribution:

$$C(\theta, \lambda) = \sum_{m=1}^{n} \log P\left(y^m | x^m; \theta\right) - \lambda H\left(y | x'; \theta\right)$$

- Here n is the number of labeled data, x^m is the m^{th} labaled sample, λ is the coefficient balancing the two terms.
- **By maximizing the conditional log-likelihood of labeled data (the first term) and minimizing the entropy of the unlabeled data(the second term), we can get better generalization performance using unlabeled data.**