文章的主要结构算法：

主要对自己阅读过的几篇论文做一下总结：

- Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks
- Virtual Adversarial Training: a Regularization Method for Supervised and Semi-supervised Learning
- Temporal Ensembling for Semi-Supervised Learning
- Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results
- mixup: BEYOND EMPIRICAL RISK MINIMIZATION
- Interpolation Consistency Training forSemi-Supervised Learning
- MixMatch: A Holistic Approach to Semi-supervise learning
- ReMixMatch: Semi-Supervised Learning with Distribution Alignment and Augmentation Anchoring
- FixMatch: Simplifying Semi-Supervised Learning withConsistency and Confidence

这几篇文章是我最近学习这方面资料所已经读过和将要读的一些文章，先看理论知识，然后了解基本的思路之后，再研究代码的实现问题。

算法1：Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks

2013年的一篇文章。

基本算法思路

Pseudo-label是将未标注样本预测来率最大的目标类作为看作真实的标签（也就是经过CNN前向传播，最终softmax输出的最大概率对应的类别作为其pseudo label）。

在训练的过程中，每次权重更新之后，重新计算无标注样本的伪标签作为真实的标签，然后使用与有监督训练分类过程中的交叉熵损失函数，同时对于有标注样本与无标注样本的损失函数进行加权处理。

$$L=\frac{1}{n}\sum_{m=1}^{n}\sum_{i=1}^{C}L(y_i^m,f_i^m)+\alpha(t)\frac{1}{n'}\sum_{m=1}^{n'}\sum_{i=1}^{C}L(y_i'^m,f_i'^m), \tag{15}$$

alpha时调节二者权重的系数，如果alpha太大，那么消除了有标注样本的作用; 如果太小，那么半监督学习几乎没有意义，因此设置alpha为下面的分段调整的形式，能够得到更好的效果。

$$\alpha(t)=\begin{cases}0 & t<T_1\\ \frac{t-T_1}{T_2-T_1}\alpha_f & T_1\le t<T_2\\ \alpha_f & T_2\le t\end{cases} \tag{16}$$

pseudo-label起作用的一些理论依据

- **决策边界应该穿过数据分布的低密度区域**

正常的直觉也是这样，当分类决策边界位于数据集的样本低密度区域时，算法能够较好的将数据分割开来，算法分类更加合理，分类准确率更高。

cluster assumption 在这篇2005年的论文中说明了理论。

- **熵正则化**

*Entropy Regularization*在这篇2006年的文章中说明，应用熵正则化可以从无标注数据的最大后验概率估计中受益，通过最小化类概率的条件熵，可以在不进行密度建模的情况下，实现低密度区域的分离。

$$H(y|x')=-\frac{1}{n'}\sum_{m=1}^{n'}\sum_{i=1}^{C}P(y_i^m=1|x'^m)\log P(y_i^m=1|x'^m) \tag{17}$$

在这个式子中，熵是未重叠的类别的度量，熵越小，随着类别重叠的越小，数据样本的密度就越小。

最大化后验概率估计：

$$C(\theta,\lambda)=\sum_{m=1}^{n}\log P(y^m|x^m;\theta)-\lambda H(y|x';\theta) \quad (18)$$

在这个式子中，第一项时有标注样本的概率最大，第二项时对于无标注样本来说的上最小，使得数据点的密度最小。由此可以得到更好的泛化能力。

- **以伪标签作为熵正则化进行训练**

分别分析上边的式子（18）与式子（15），18的第一项对应15的第一项，18的第二项对应于15的第二项，因此伪标签的方法等价与熵正则化。

使用伪标签的训练，使得无标注数据样本的熵变得更小，使得决策边界处于数据样本密度更低的位置，分类泛化性能更好。

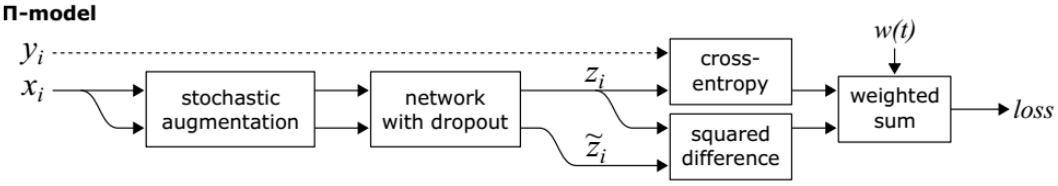
算法2：Temporal Ensembling for Semi-Supervised Learning

2017年的文章

算法思想

文中主要提出了两个模型第一个是 Π model,网络结构以及算法伪代码如下图所示。

- **Π Model**



根据以上结构图简要介绍算法的操作流程。

1. 训练的过程中，对于一个输入图片经过随机噪声及dropout的网络之后，得到两个不同的输出Zi与Zi‘。
2. 损失函数有两部分组成，第一部分评估有标注样本的交叉熵损失函数（仅处理有标注样本），第二部分惩罚两个输入Zi与Zi’的不一致（MSE,处理所有的样本，包括有标注与无标注）

3. 损失函数两项的权值采用加权的形式，加权系数为w(t)，这个系数随时间变化，在训练刚开始时，我们希望有标注样本占据主要部分，因此交叉熵损失比重应该大，随着训练过程的进行，第二种损失的比重逐渐变大。

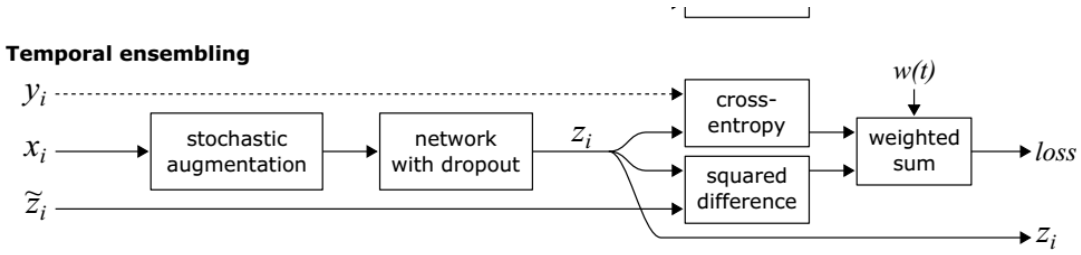
Algorithm 1 Π-model pseudocode.

Require: x_i = training stimuli
Require: L = set of training input indices with known labels
Require: y_i = labels for labeled inputs $i \in L$
Require: $w(t)$ = unsupervised weight ramp-up function
Require: $f_\theta(x)$ = stochastic neural network with trainable parameters θ
Require: $g(x)$ = stochastic input augmentation function
for t in $[1, num\_epochs]$ **do**
  for each minibatch B **do**
    $z_{i \in B} \leftarrow f_\theta(g(x_{i \in B}))$ ▷ evaluate network outputs for augmented inputs
    $\tilde{z}_{i \in B} \leftarrow f_\theta(g(x_{i \in B}))$ ▷ again, with different dropout and augmentation
    $loss \leftarrow -\frac{1}{|B|} \sum_{i \in (B \cap L)} \log z_i[y_i]$ ▷ supervised loss component
      $+ w(t) \frac{1}{C|B|} \sum_{i \in B} ||z_i - \tilde{z}_i||^2$ ▷ unsupervised loss component
    update θ using, e.g., ADAM ▷ update network parameters
  end for
end for
return θ

- **Temporal ensembling**

这个算法模型与Π model基本一致，针对Π Model作了一些改进。

1. 由于Π Model在每个step需要推理两次模型才能得到Zi与Zi‘，实际上并不需要进行两次推理，这样产生冗余，减慢了推理速度
2. 在Temporal ensembling模型中，使用一次推理就可以得到Zi，而Zi‘由之前一段时间得到的Z指数移动平均得到，Zi由本来自当前迭代时间内产生。Zi’保存了之前的信息，这样能更好的消除扰动稳定当前值。



Algorithm 2 Temporal ensembling pseudocode. Note that the updates of Z and $\tilde{z}$ could equally well be done inside the minibatch loop; in this pseudocode they occur between epochs for clarity.

Require: x_i = training stimuli
Require: L = set of training input indices with known labels
Require: y_i = labels for labeled inputs $i \in L$
Require: α = ensembling momentum, $0 \leq \alpha < 1$
Require: $w(t)$ = unsupervised weight ramp-up function
Require: $f_\theta(x)$ = stochastic neural network with trainable parameters θ
Require: $g(x)$ = stochastic input augmentation function

$Z \leftarrow \mathbf{0}_{[N \times C]}$ ▷ initialize ensemble predictions
$\tilde{z} \leftarrow \mathbf{0}_{[N \times C]}$ ▷ initialize target vectors
for t in $[1, num\_epochs]$ **do**
  for each minibatch B **do**
    $z_{i \in B} \leftarrow f_\theta(g(x_{i \in B}, t))$ ▷ evaluate network outputs for augmented inputs
    $loss \leftarrow -\frac{1}{|B|} \sum_{i \in (B \cap L)} \log z_i[y_i]$ ▷ supervised loss component
      $+ w(t) \frac{1}{C|B|} \sum_{i \in B} ||z_i - \tilde{z}_i||^2$ ▷ unsupervised loss component
    update θ using, e.g., ADAM ▷ update network parameters
  end for
  $Z \leftarrow \alpha Z + (1 - \alpha) z$ ▷ accumulate ensemble predictions
  $\tilde{z} \leftarrow Z / (1 - \alpha^t)$ ▷ construct target vectors by bias correction
end for
return θ

算法3：Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results

2018年的文章

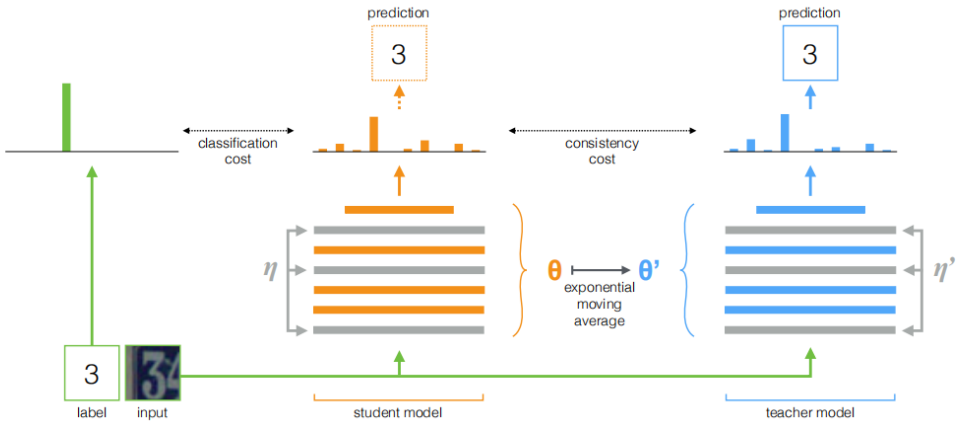
论文主要针对temporal ensambling的一些**缺点**做出了改进，temporal ensambling对于每个训练样本的**label预测做指数移动平均**并且惩罚对于同一个目标预测的不一致的项，使用这种方式target每一个个epoch仅能更改一次，这对于大数据集与在线学习就不能使用，因此在这个算法中，作者不采用对于label prediction进行指数移动平均，而是对于**教师模型的权重参数**进行指数移动平均（EMA），这样就可以每个step更新一次，不仅解决了大数据集与在线学习的问题，而且也提高了识别的精度。

算法思想

1. 由于数据标注的成本太高，因此考虑采用正则化的方式来利用未标注的样本来降低深度学习模型的过拟合
2. 对于图像分类模型来说，当输入的图像稍微有变化时，通常认为时同一个输入，希望模型能够输出预测相同的结果，因此可以通过在模型的输入中加入噪声（数据增广data augmentation）实现，也可以采用诸如dropout等正则化策略来实现。
3. 由于无标注样本的分类损失没有定义，因此对于其自身进行正则化并不会有主动半监督学习。针对这种情况对于每个样本的有噪声与无噪声的情况进行评估，并且在这两种情况的预测应用一致性损失（MSE）。
4. 应用学生教师模型，学生模型进行正常的预测，教师模型生成目标的预测，然后学生模型用来做预测，单是由于教师模型本身存在偏差，所生成的target本身也会存在大量的预测偏差，如果赋予权重较大，那么一致性损失就会非常大，出现误预测。可以通过提升target的质量来提升效果

5. 两种方法来提升target的质量，一种时VAT中使用的方法，通过巧妙的选择扰动，而不是直接诶添加加性或者乘性噪声; 第二种方案是仔细巧妙的选择教师模型而不是简单的学生模型的复制（本论文采用的方法）。
6. 目标是不进行额外的训练从学生模型中获得更好的教师模型。由于softmax的输出不会在训练数据之外产生更加精确的预测，通常在推理阶段给模型加入噪声会缓解这种情况（类似TTA），这种方式temporal ensambling算法已经证实会产生良好的结果。
7. 本文的算法在temporal sampling的基础上，采用EMA来对教师模型的权重进行更新，在大数据集以及在线学习上有更好可用之处

算法介绍



如上图所示，使用student model的指数平均权重来更新teacher model的模型。权重的平均值不仅改善了顶层的输出，而且改善了所有层的输出，因此会产生更好的结果。

对比temporal ensambling算法，有两方面的优势：第一，由于教师模型产生更加准确的target label，因此，在教师与学生模型之间的反馈过程会收敛的更快，准确率更高。第二，在大型数据集与在线学习上扩展应用较好。

损失函数采用一致性损失（均方误差损失MSE）：

$$J(\theta) = \mathbb{E}_{x,\eta',\eta}\left[\|f(x,\theta',\eta') - f(x,\theta,\eta)\|^2\right]$$

权重的移动平均：

$$\theta'_t = \alpha\theta'_{t-1} + (1-\alpha)\theta_t$$

算法4：Virtual Adversarial Training: a Regularization Method for Supervised and Semi-supervised Learning

2018年的文章

算法思想

本文提出了一种新的正则化的方法，`vat`是一种基于熵最小化出发的正则方法。vat提出了一种对于给定输入评估模型输出条件分布局部光滑性的对抗损失。这种对抗损失不需要输出label的信息因此可以用于半监督学习。

为了提升模型的泛化能力(鲁棒性），对于输入样本加入扰动，然后对于同一个样本鼓励模型去获得同样的输出（**一致性正则化**）。

如果添加各向同性的随机扰动和对于输入图像进行随机的数据增广，那么预测器在模型的某些方向的扰动效果非常差，这就是对抗扰动的方向，所以需要添加合适的扰动（对抗扰动）。对抗扰动采用能够使模型的输出分布产生更大变动的方向。对于NN模型，其中负梯度方向是模型的损失下降最快的方向，那即负梯度方向上模型优化最快， 为了对模型的输出分布产生最大的改变，正梯度方向也就是模型梯度下降最慢的方向定为对抗扰动方向。

文中提出虚拟对抗扰动主要是针对无标注数据集而言，即使没有标签也能够找出虚拟对抗扰动的方向，定义LDS（Local Distributional Smoothness）为针对虚拟对抗方向的模型基于散度的分布鲁棒性，文中提出了一种新颖的训练方法，使用有效的近似来最大化模型的似然，同时在每个训练输入数据点上提升模型的`LDS`，这中方法就是vat。

VAT的优点：

1. 应用于半监督学习
2. 适用于任何针对输入与参数我们可以评估梯度的参数模型
3. 超参数少
4. 参数不变正则化

针对第二个优点，要解决优化问题，首先就要求解一个优化问题，从而去得到虚拟对抗的方向，例如对于NN来说，我们必须要求出输入相对于输出的梯度，虚拟对抗扰动对于这个问题提出了一个简单的高校的解决方法，可以应用到NN中。

针对第四个优点，VAT不同于传统的正则化，对于Lp正则化来说，线性模型可以比较好的的控制模型的参数得到较好的效果，然而对于非线性程度高的模型，这种正则化策略变得不可控，效果不好，需要参数不变的正则化，VAT就是这样的正则化策略。

算法细节

对抗训练：

$$L_{\mathrm{adv}}(x_l, \theta) := D\left[q(y|x_l), p(y|x_l + r_{\mathrm{adv}}, \theta)\right]$$
$$\text{where } r_{\mathrm{adv}} := \arg\max_{r;\|r\|\leq\epsilon} D\left[q(y|x_l), p(y|x_l + r, \theta)\right],$$

D是散度，表示两个分布的相似程度。通常对于特定的扰动得不到精确的radv，通常可以采用D对r的线性近似得到radv，对于l2正则化

$$r_{\mathrm{adv}} \approx \epsilon \frac{g}{\|g\|_2}, \text{ where } g = \nabla_{x_l} D\left[h(y; y_l), p(y|x_l, \theta)\right]$$

l无穷正则化：

$$r_{\mathrm{adv}} \approx \epsilon \mathrm{sign}(g),$$

虚拟对抗训练，与对抗训练的不同点在于虚拟对抗训练需要处理大龄无标注样本的半监督学习问题，因此其散度变为：

$$D\left[q(y|x_*), p(y|x_* + r_{\mathrm{qadv}}, \theta)\right]$$
$$\text{where } r_{\mathrm{qadv}} := \arg\max_{r;\|r\|\leq\epsilon} D\left[q(y|x_*), p(y|x_* + r, \theta)\right],$$

对于无标注样本，没有直接的标签信息，因此，我们采取了用当前近似值p(y|x,θ)代替q(y|x)的策略当带标签的训练样本数量很大时，p(y|x,θ)应该接近q(y|x)。使用从p(y|x,θ)概率生成的虚拟标签代替用户不知道的标签，并根据虚拟标签计算对抗方向。

因此在这篇文章中采用现在的估计值p(y|x,θ^)取代q(y|x).

$$\mathrm{LDS}(x_*, \theta) := D\left[p(y|x_*, \hat{\theta}), p(y|x_* + r_{\mathrm{vadv}}, \theta)\right]$$
$$r_{\mathrm{vadv}} := \arg\max_{r;\|r\|_2\leq\epsilon} D\left[p(y|x_*, \hat{\theta}), p(y|x_* + r)\right],$$

其中的x*，包含有标注与无标注的样本。LDS对于在每个输入数据点的现在的模型定义了一个负的评估标准，LDS的减小标志着模型在输入点更加的平滑，正则化项是所有数据点的LDS的平均值。

$$\mathcal{R}_{\mathrm{vadv}}(\mathcal{D}_l, \mathcal{D}_{ul}, \theta) := \frac{1}{N_l + N_{ul}} \sum_{x_* \in \mathcal{D}_l, \mathcal{D}_{ul}} \mathrm{LDS}(x_*, \theta).$$

那么整个的目标函数就是：

$$\ell(\mathcal{D}_l, \theta) + \alpha \mathcal{R}_{\text{vadv}}(\mathcal{D}_l, \mathcal{D}_{ul}, \theta),$$

其中，第一项是有标注数据的交叉熵。

算法5：mixup: BEYOND EMPIRICAL RISK MINIMIZATION

2018年的文章

监督学习中，采用对于训练数据的最小化平均损失，也就是经验损失最小化（ERM），这种方案对于NN来说就是简单的记忆训练数据的样本特征，这还总记忆的特地那会导致学习机的泛化能力变差（这部分文中有分析，可以参考原文）。但是当面临新的数据分布时（没有出现过的数据，例如测试集），ERM不能保证学习机具有很强的泛化能力。

后来的研究方法为了增强学习机的泛化能力，采用了数据增广的策略来实现邻域风险最小化（Vicinal Risk Minimization），利用训练数据集的扰动来增强学习机的表现能力。但是一般的数据增广仅仅在同类间对数据进行处理，没有进行类间数据的融合操作。

本文的算法提出，采用mixup对不同类的数据进行数据增广。

对于不同类别的数据进行线性的操作混合。

$$\begin{aligned} \tilde{x} &= \lambda x_i + (1-\lambda) x_j, && \text{where } x_i, x_j \text{ are raw input vectors} \\ \tilde{y} &= \lambda y_i + (1-\lambda) y_j, && \text{where } y_i, y_j \text{ are one-hot label encodings} \end{aligned}$$

简单的操作最后效果非常好，而且随后的半监督学习中很多方法都加上了这种方法。

算法6： Interpolation Consistency Training forSemi-Supervised Learning

2019年的文章

算法介绍

提出了一个应用于半监督学习的算法——Interpolation Consistency Training（本质就是上边介绍算法mixup与mean—teacher的集合），利用类似于mixup的方法，设计插值一致性训练。

miup的公式：

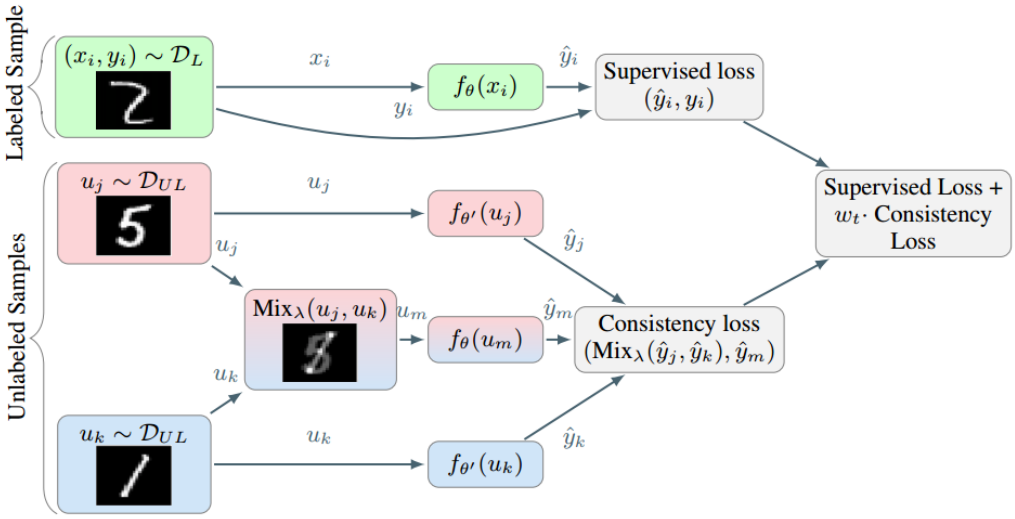
$$\text{Mix}_\lambda(a, b) = \lambda \cdot a + (1-\lambda) \cdot b,$$

这个算法就是训练一个分类器预测其在无标注样本点的插值的连续预测结果：

$$f_\theta(\text{Mix}_\lambda(u_j, u_k)) \approx \text{Mix}_\lambda(f_{\theta'}(u_j), f_{\theta'}(u_k)),$$

其中theta'为学生模型的参数的移动平均。

算法的网络结构：



类似mean-teacher的算法操作流程，不过多了插值的一个步骤。

算法流程：

Algorithm 1 The Interpolation Consistency Training (ICT) Algorithm

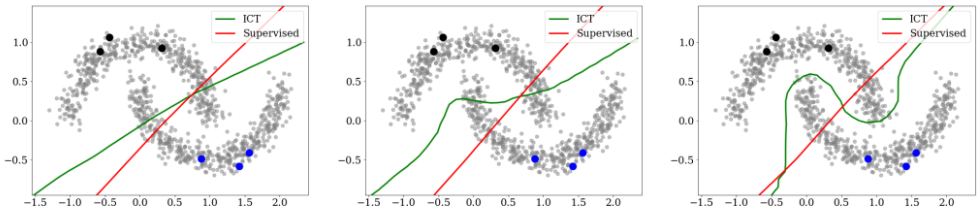
Require: $f_\theta(x)$: neural network with trainable parameters θ
Require: $f_{\theta'}(x)$ mean teacher with θ' equal to moving average of θ
Require: $\mathcal{D}_L(x, y)$: collection of the labeled samples
Require: $\mathcal{D}_{UL}(x)$: collection of the unlabeled samples
Require: α: rate of moving average
Require: $w(t)$: ramp function for increasing the importance of consistency regularization
Require: T: total number of iterations
Require: Q: random distribution on [0,1]
Require: $\text{Mix}_\lambda(a, b) = \lambda a + (1 - \lambda)b$.
for $t = 1, \ldots, T$ **do**
Sample $\{(x_i, y_i)\}_{i=1}^B \sim \mathcal{D}_L(x, y)$ ▷ Sample labeled minibatch
$L_S = \text{CrossEntropy}(\{(f_\theta(x_i), y_i)\}_{i=1}^B)$ ▷ Supervised loss (cross-entropy)
Sample $\{u_j\}_{j=1}^U, \{u_k\}_{k=1}^U \sim \mathcal{D}_{UL}(x)$ ▷ Sample two unlabeled examples
$\{\hat{y}_j\}_{j=1}^U = \{f_{\theta'}(u_j)\}_{j=1}^U, \{\hat{y}_k\}_{k=1}^U = \{f_{\theta'}(u_k)\}_{k=1}^U$ ▷ Compute fake labels
Sample $\lambda \sim Q$ ▷ sample an interpolation coefficient
$(u_m = \text{Mix}_\lambda(u_j, u_k), \hat{y}_m = \text{Mix}_\lambda(\hat{y}_j, \hat{y}_k))$ ▷ Compute interpolation
$L_{US} = \text{ConsistencyLoss}(\{(f_\theta(u_m), \hat{y}_m)\}_{m=1}^U)$ ▷ e.g., mean squared error
$L = L_S + w(t) \cdot L_{US}$ ▷ Total Loss
$g_\theta \leftarrow \nabla_\theta L$ ▷ Compute Gradients
$\theta' = \alpha\theta' + (1 - \alpha)\theta$ ▷ Update moving average of parameters
$\theta \leftarrow \text{Step}(\theta, g_\theta)$ ▷ e.g. SGD, Adam
end for
return θ

文中提到的一些理论知识

1. 聚类假设

如果两个样本分别术语两个聚类，那么他们更可能术语两个类，这也等价于前边几个算法中提到的低密度分离假设，即学习机的分类决策边界穿过分类样本点的数据分布密度更小的位置。这也是一致性正则化的内容。

一致性正则化是指对应同一个输入样本的一个小小的扰动，那么学习机的预测依然保持原来的预测一致。这也就是满足低密度分离假设。



(a) After 100 updates (b) After 500 updates (c) After 1000 updates

Figure 1: Interpolation Consistency Training (ICT) applied to the "two moons" dataset, when three labels per class (large dots) and a large amount of unlabeled data (small dots) is available. When compared to supervised learning (red), ICT encourages a decision boundary traversing a low-density region that would better reflect the structure of the unlabeled data. Both methods employ a multilayer perceptron with three hidden ReLU layers of twenty neurons.

2. 一致性正则化的策略

实施一致性正则化的策略，往往采用给输入样本加入随机的扰动，但那是随机扰动效果往往不尽人意，因此在VAT算法中，采用了能够使得算法的预测输出产生最大改变的扰动。然而这种扰动的产生方法需要计算预测输出关于输入的梯度，对于很大的NN模型来说计算复杂度太高。而且也有文章证明，这种方法会损害模型的泛化能力。

为了克服随机扰动的不稳定性，VAT中扰动的缺点。本文提出的ICT是一种高效的一致性正则化策略。采用无标注样本u1与u2的插值作为扰动。 这种策略简单高效。

3. 插值策略的有效性

为什么这种插值的策略有效呢？因为只有在决策边界的数据点才更夹有效，因为对于这些点加入扰动之后，可能把这个点变为另外一类，可以提高模型的决策难度，提升泛化能力。

对于选中进行插值的两个数据点u1与u2,其能够出现以下三种情况：

1. 处于同一个聚类中
2. 位于不同的聚类，但是属于相同的类
3. 位于不同的聚类中，属于不同的类

出现3的概率最大，当选择一个无标注样本点位于决策边界处，其插值的方向朝着低密度区，那么插值之后的插值点可能位于另一个聚类，因此这种插值的策略对于一致性正则化非常有利。

由于在有监督训练中mixup强迫模型在两类样本线性变化，使得决策边界远离类边界，因此这里借用mixup的思想。

算法7：MixMatch: A Holistic Approach to Semi-supervise learning

2019年的文章

算法的思想

这个算法就是之前一些常用的半监督学习的paradigm（范式）的整合。

半监督学习常用的paradigm的整合，常用的方法有：

- 熵最小化

这个方法的介绍在 `算法1：pesudo-label` 的文章中已经介绍过相关的概念，主要是利用了决策边界穿过数据点密度较小的区域这个假设。在pesudo label中，作者通过将高置信度的预测作为训练数据的target并且利用target作为无标注数据的label应用交叉熵损失，来隐式的实现熵最小化。

本文中作者通过对无标注样本的target的分布应用shapen函数来隐式的实现交叉熵。

- 一致性正则化

主要思想就是模型对于同样的样本应该输出同样的预测，即使样本遭受一些扰动。在temporal emsembling中作者通过对于target输出值进行移动平均，依据输出值Zi与Zi‘的误差来实现一致性正则化，mean-teacher算法中，作者通过指数移动平均来更新模型的参数，从而间接实现一致性正则化。

mixmatch使用标注的data augmentation的方法实现一致性正则化。

- 一般正则化

一般正则化的主要目的是提升模型的泛化能力，降低过拟合。

mixmatch利用l2正则化来乘法模型的参数，实现权重衰减。也是用mixup的方法作为数据增广。

算法流程

输入同等数量的有标注样本x与无标注样本u，其中x带有one-hot的标签信息，然后利用数据增广产生增光后的X’（其中含有label），增广后的U‘（其中含有猜测的label），对于有标注的X’与带有猜测label的U’分别计算loss，然后整合loss，得到最终的unified loss。

$$\mathcal{X}', \mathcal{U}' = \text{MixMatch}(\mathcal{X}, \mathcal{U}, T, K, \alpha)$$

$$\mathcal{L}_{\mathcal{X}} = \frac{1}{|\mathcal{X}'|} \sum_{x,p \in \mathcal{X}'} \text{H}(p, \text{p}_{\text{model}}(y \mid x; \theta))$$

$$\mathcal{L}_{\mathcal{U}} = \frac{1}{L|\mathcal{U}'|} \sum_{u,q \in \mathcal{U}'} \|q - \text{p}_{\text{model}}(y \mid u; \theta)\|_2^2$$

$$\mathcal{L} = \mathcal{L}_{\mathcal{X}} + \lambda_{\mathcal{U}} \mathcal{L}_{\mathcal{U}}$$

1. **数据增广**

对有标注与无标注的样本均使用数据增广。对于无标注的样本生成K个增光后的图像，利用这些增广后的K个图像生成猜测的label。

Algorithm 1 MixMatch takes a batch of labeled data $\mathcal{X}$ and a batch of unlabeled data $\mathcal{U}$ and produces a collection $\mathcal{X}'$ (resp. $\mathcal{U}'$) of processed labeled examples (resp. unlabeled with guessed labels).

Input: Batch of labeled examples and their one-hot labels $\mathcal{X} = \big((x_b, p_b); b \in (1, \dots, B)\big)$, batch of unlabeled examples $\mathcal{U} = \big(u_b; b \in (1, \dots, B)\big)$, sharpening temperature T, number of augmentations K, Beta distribution parameter α for MixUp.
for $b = 1$ **to** B **do**
  $\hat{x}_b = \text{Augment}(x_b)$ *// Apply data augmentation to* x_b
  for $k = 1$ **to** K **do**
    $\hat{u}_{b,k} = \text{Augment}(u_b)$ *// Apply* k^{th} *round of data augmentation to* u_b
  end for
  $\bar{q}_b = \frac{1}{K} \sum_k \text{p}_{\text{model}}(y \mid \hat{u}_{b,k}; \theta)$ *// Compute average predictions across all augmentations of* u_b
  $q_b = \text{Sharpen}(\bar{q}_b, T)$ *// Apply temperature sharpening to the average prediction (see eq. (7))*
end for
$\hat{\mathcal{X}} = \big((\hat{x}_b, p_b); b \in (1, \dots, B)\big)$ *// Augmented labeled examples and their labels*
$\hat{\mathcal{U}} = \big((\hat{u}_{b,k}, q_b); b \in (1, \dots, B), k \in (1, \dots, K)\big)$ *// Augmented unlabeled examples, guessed labels*
$\mathcal{W} = \text{Shuffle}\big(\text{Concat}(\hat{\mathcal{X}}, \hat{\mathcal{U}})\big)$ *// Combine and shuffle labeled and unlabeled data*
$\mathcal{X}' = \big(\text{MixUp}(\hat{\mathcal{X}}_i, \mathcal{W}_i); i \in (1, \dots, |\hat{\mathcal{X}}|)\big)$ *// Apply* MixUp *to labeled data and entries from* $\mathcal{W}$
$\mathcal{U}' = \big(\text{MixUp}(\hat{\mathcal{U}}_i, \mathcal{W}_{i+|\hat{\mathcal{X}}|}); i \in (1, \dots, |\hat{\mathcal{U}}|)\big)$ *// Apply* MixUp *to unlabeled data and the rest of* $\mathcal{W}$
return $\mathcal{X}', \mathcal{U}'$

2. **label guessing**

对于无标注样本的K个增广样本的预测值进行平均，得到:

$$\bar{q}_b = \frac{1}{K} \sum_{k=1}^{K} \text{p}_{\text{model}}(y \mid \hat{u}_{b,k}; \theta)$$

然后利用sharpen函数隐式的实现熵的最小化。

$$\text{Sharpen}(p, T)_i := p_i^{\frac{1}{T}} \Big/ \sum_{j=1}^{L} p_j^{\frac{1}{T}}$$

其中p是一个输入的数据分布，在此处就是qb，T是温度超参数，当T趋近于0时，分布趋近于Driac（one-hot）。

3. **Mixup**

首先整理所有的无标注与有标注的样本，qb为预测的label。

$$\hat{\mathcal{X}} = \big((\hat{x}_b, p_b); b \in (1, \dots, B)\big)$$
$$\hat{\mathcal{U}} = \big((\hat{u}_{b,k}, q_b); b \in (1, \dots, B), k \in (1, \dots, K)\big)$$

与过去的工作不同治理对于有标注与无标注样本进行混和，为了使用分离版本的loss（有标注与无标注样本分别计算），使用修正版本的mixup。

$$\lambda \sim \mathrm{Beta}(\alpha, \alpha)$$
$$\lambda' = \max(\lambda, 1-\lambda)$$
$$x' = \lambda' x_1 + (1-\lambda') x_2$$
$$p' = \lambda' p_1 + (1-\lambda') p_2$$

在原始版本的mixup中， lamda'等于lamda。此处alpha为超参数。然后记录无标注样本与有标注样本出现在一个batch中的顺序，用于计算分离的loss。

超参数的论文最优设置可以参考原始论文。

算法8： ReMixMatch: Semi-Supervised Learning with Distribution Alignment and Augmentation Anchoring

2020年的文章

算法思想

这篇文章是mixmatch算法的改进版本。

主要引入了Distribution Alignment和Augmentation Anchoring来改进mixmatch。同时使用交叉熵代替MSE，使用CTAugmentaion数据增广策略。

Distribution Alignment作用是强迫无标注数据的边缘分布与真实标签的边缘分布一致。

Augmentation Anchoring的作用是对于无标注样本引入多个更强的数据增广产生多个增广版本的样本，来促使模型输出与统一个无标注样本的弱增广版本的结果一致。

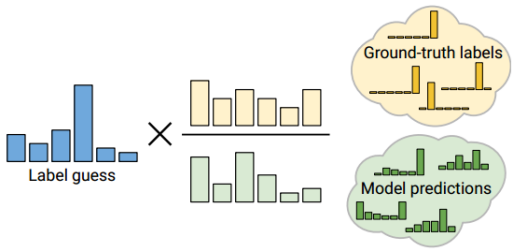


Figure 1: Distribution alignment. Guessed label distributions are adjusted according to the ratio of the empirical ground-truth class distribution divided by the average model predictions on unlabeled data.

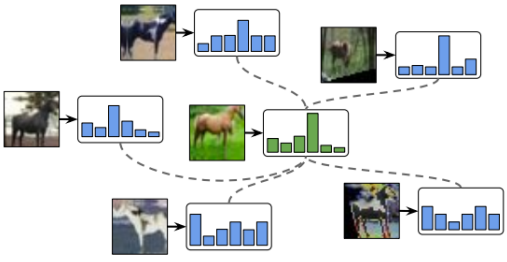


Figure 2: Augmentation anchoring. We use the prediction for a weakly augmented image (green, middle) as the target for predictions on strong augmentations of the same image (blue).

Distribution Alignment（分布对齐）的主要作用就是令模型整体的预测输出分布与真实label的边缘分布相匹配。最大化输入输出之间的互信息，利用这个理论，分布对齐直接加入mixmatch中来修正guessed label。

Augmentation Anchoring取代了mixmatch中的一致性正则化。对于给定的无标注样本，首先产生一个弱增强的数据，然后产生多个强增强的数据。弱增强的数据产生的预测作为所有强增强数据的guessed label。其中，对于强增强数据的操作，本文采用了CTAugmentation。

算法流程伪代码如下：

Algorithm 1 ReMixMatch algorithm for producing a collection of processed labeled examples and processed unlabeled examples with label guesses (cf. Berthelot et al. (2019) Algorithm 1.)

Input: Batch of labeled examples and their one-hot labels $\mathcal{X} = \big((x_b, p_b) : b \in (1, \dots, B)\big)$, batch of unlabeled examples $\mathcal{U} = \big(u_b : b \in (1, \dots, B)\big)$, sharpening temperature T, number of augmentations K, Beta distribution parameter α for MixUp.
for $b = 1$ **to** B **do**
  $\hat{x}_b = \text{StrongAugment}(x_b)$ // *Apply strong data augmentation to* x_b
  $\hat{u}_{b,k} = \text{StrongAugment}(u_b); k \in \{1, \dots, K\}$ // *Apply strong data augmentation* K *times to* u_b
  $\tilde{u}_b = \text{WeakAugment}(u_b)$ // *Apply weak data augmentation to* u_b
  $q_b = p_{\text{model}}(y \mid \tilde{u}_b; \theta)$ // *Compute prediction for weak augmentation of* u_b
  $q_b = \text{Normalize}(q_b \times p(y) / \tilde{p}(y))$ // *Apply distribution alignment*
  $q_b = \text{Normalize}\big(q_b^{1/T}\big)$ // *Apply temperature sharpening to label guess*
end for
$\hat{\mathcal{X}} = \big((\hat{x}_b, p_b); b \in (1, \dots, B)\big)$ // *Augmented labeled examples and their labels*
$\hat{\mathcal{U}}_1 = \big((\hat{u}_{b,1}, q_b); b \in (1, \dots, B)\big)$ // *First strongly augmented unlabeled example and guessed label*
$\hat{\mathcal{U}} = \big((\hat{u}_{b,k}, q_b); b \in (1, \dots, B), k \in (1, \dots, K)\big)$ // *All strongly augmented unlabeled examples*
$\hat{\mathcal{U}} = \hat{\mathcal{U}} \cup \big((\tilde{u}_b, q_b); b \in (1, \dots, B)\big)$ // *Add weakly augmented unlabeled examples*
$\mathcal{W} = \text{Shuffle}\big(\text{Concat}(\hat{\mathcal{X}}, \hat{\mathcal{U}})\big)$ // *Combine and shuffle labeled and unlabeled data*
$\mathcal{X}' = \big(\text{MixUp}(\hat{\mathcal{X}}_i, \mathcal{W}_i); i \in (1, \dots, |\hat{\mathcal{X}}|)\big)$ // *Apply* MixUp *to labeled data and entries from* $\mathcal{W}$
$\mathcal{U}' = \big(\text{MixUp}(\hat{\mathcal{U}}_i, \mathcal{W}_{i+|\hat{\mathcal{X}}|}); i \in (1, \dots, |\hat{\mathcal{U}}|)\big)$ // *Apply* MixUp *to unlabeled data and the rest of* $\mathcal{W}$
return $\mathcal{X}', \mathcal{U}', \hat{\mathcal{U}}_1$

输入输出互信息：

$$\begin{aligned}\mathcal{I}(y; x) &= \iint p(y, x) \log \frac{p(y, x)}{p(y)p(x)} \, \mathrm{d}y \, \mathrm{d}x \\ &= \mathcal{H}(\mathbb{E}_x[p_{\text{model}}(y|x; \theta)]) - \mathbb{E}_x[\mathcal{H}(p_{\text{model}}(y|x; \theta))]\end{aligned}$$

第二项是熵最小化，鼓励模型的输出具有更小的熵（对一个类别的输出的置信度最高），第一项其目标是鼓励模型在整个训练集平均预测每个类别的频率相同，这被称为公平性，

在mixmatch中采用sharpen的操作实现熵的最小化，因此鉴于上边的考虑，考虑引入一个公平性的原则。上边互信息公式中的第一项暗示了它应该以相同的频率预测每个标签，但如果数据集中类别的边缘概率分布p(y)并不是均匀的，那这个目标就不一定有效了。

为了不引入额外的超参数与额外的loss项，采用Distribution Alignment的方法。算法伪代码中的`6 7 8`实现相关的操作。

细节

1. 为何不采用Auto Augmentation?

作者尝试了直接在mixmatch中直接尝试了auto-augmentation的强增强方法，然而算法不收敛，猜想可能的原因就是mixmatch中使用了平均的操作，而强增强的策略可能预测产生不同的值，因此平均以后的结果不一定是有意义的值。

同时由于AutoAugmentation是在有监督训练中进行搜索得到（强化学习），需要大量的标注数据，然而在半监督学习中，有标注样本的数目有限。不能哈斯用这种增强方法。

2. CTAugmentation

因此文中作者使用控制理论的思想设计出CTAugmentaion，而无需任何形式的基于强化学习的训练。（这个增强算法不了解具体怎么操作，随后得去看看是否有代码）

3. 最终模型的损失函数：

第一项为有标注样本的交叉熵损失，第二项为mixup后的无标注样本的交叉熵损失，第三项为强增强样本的交叉熵损失

第四项为rotation loss，这里借鉴了自监督的思想，将样本之后作为四分类损失（旋转角度为0,90,180,270,四种）

$$\sum_{x,p\in\mathcal{X}'} \mathrm{H}(p, p_{\mathrm{model}}(y|x;\theta)) + \lambda_{\mathcal{U}} \sum_{u,q\in\mathcal{U}'} \mathrm{H}(q, p_{\mathrm{model}}(y|u;\theta))$$
$$+\lambda_{\hat{\mathcal{U}}_1} \sum_{u,q\in\hat{\mathcal{U}}_1} \mathrm{H}(q, p_{\mathrm{model}}(y|u;\theta)) + \lambda_r \sum_{u\in\hat{\mathcal{U}}_1} \mathrm{H}(r, p_{\mathrm{model}}(r|\,\mathrm{Rotate}(u,r);\theta))$$

算法9： FixMatch: Simplifying Semi-Supervised Learning withConsistency and Confidence

2020年的文章

算法思路

这是一个非常简单的算法。实际上是pseudo label（伪标签） 和 consistence regularization（一致性正则化）。

1. 利用弱数据增广图像生成伪标签
2. 利用阈值，保留预测置信度更高的伪标签
3. 将产生的伪标签作为label对强增广的图像利用正常交叉熵进行训练

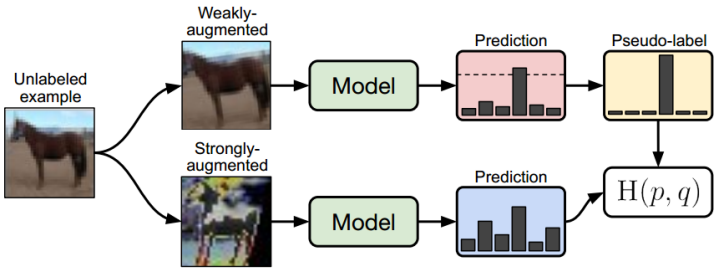


Figure 1: Diagram of FixMatch, our proposed semi-supervised learning algorithm. First, a weakly-augmented version of an unlabeled image (top) is fed into the model to obtain its predictions (red box). When the model assigns a probability to any class which is above a threshold (dotted line), the prediction is converted to a one-hot pseudo-label. Then, we compute the model's prediction for a strongly augmented version of the same image (bottom). The model is trained to make its prediction on the strongly-augmented version match the pseudo-label via a standard cross-entropy loss.

损失函数由两部分组成，有监督训练的交叉熵，无监督交叉熵

有监督交叉熵针对有标注样本，对有标注样本不采用强数据增广，仅仅采用弱数据增广。

$$\ell_s = \frac{1}{B}\sum_{b=1}^{B} \mathrm{H}(p_b, p_{\mathrm{m}}(y \mid \alpha(x_b)))$$

无监督交叉熵针对无标注样本：

$$\ell_u = \frac{1}{\mu B}\sum_{b=1}^{\mu B} \mathbb{1}(\max(q_b) \geq \tau)\, \mathrm{H}(\hat{q}_b, p_{\mathrm{m}}(y \mid \mathcal{A}(u_b)))$$

最终的loss依然采用二者的加权。

更多技术文章请点击查看