第四章 自然语言分类任务

扫码试看/订阅

《NLP 实战高手课》视频课程

4.1 重新审视 Word Embedding

内容概述

- Negative Sampling
- Word Embedding 的一些问题
- 早期解决 Word Embedding 问题的一些方法

Negative Sampling

Word Embedding 的问题

早期解决 Word Embedding 问题的一些方法

- 加入 POS
- Context Embedding
  - https://www.aclweb.org/anthology/K16-1006.pdf
- 通过翻译加入上下文因素
  - https://papers.nips.cc/paper/7209-learned-in-translation-contextualized-word-vectors.pdf

补充材料

- 关于 Negative Sampling 的推导：
  - https://arxiv.org/pdf/1402.3722.pdf

4.2 深度迁移学习模型：从 ELMo 到 BERT

内容概述

- 深度迁移学习模型的核心思想
- ELMo
- BERT

深度迁移学习的核心思想

- 解决 Word Embedding 不足
- 充分利用无标注的数据
- 使用较深的模型

ELMo

- 架构和任务
- 使用方法

BERT

- 架构和任务
- 使用方法

4.3 深度迁移学习模型：RoBERTa、XLNet、ERNIE 和 T5

内容综述

- RoBERTa
- XLNet
- ERNIE
- T5

RoBERTa

- 核心思想：
  - 通过更好地训练 BERT 可以达到超过其他新的预训练语言模型的效果
- 核心改动：
  - 更大的 Batch Size
  - 去掉 Next Sentence Prediction
  - 采用更大的预训练语料
  - Dynamic Masking

XLNet

- 主要改动
  - Permutation Language Modeling
  - Transformer-XL
- XLNet 还在很多地方进行了改动

ERNIE

- 核心思想：
  - 使用 Multi-task Learning 提升模型效果

T5

- 核心思想：
  - 将 Multi-task Learning 的任务变为 Seq2Seq 的任务
  - 测试了各类预训练语言模型设定的有效性

4.4 深度迁移学习模型：ALBERT 和 ELECTRA

内容综述

- 核心思想
- ALBERT
- ELECTRA

核心思想

- 只要是增加预训练模型的复杂度，人们往往可以得到更好的效果
- 但是：
  - 更大的模型意味着更大的算力消耗
  - 更大的模型意味着预训练语言模型过程的更大消耗
- 结论：应该在同等复杂度上进行模型比较

ALBERT

- 核心思想
  - 权重共享
  - 输入层的优化
  - Sentence Order Prediction

ELECTRA

- 核心思想：采用对抗训练提升模型训练效果
  - 通过 MLM 训练 Generator
  - Discriminator 负责区分 Generator 生成的 token 是否被替代
- 其他改进：
  - 采用权重共享

4.7 优化器：Adam 和 AdamW

内容综述

- 推进优化器文献进展的思想
- Adam 以及 Adam 的实现（PyTorch）
- Weight Decay 和 AdamW
- Weight Decay 和 Normalization

Adam 以及 Adam 的实现（PyTorch）

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector

$m_0 \leftarrow 0$ (Initialize 1^{st} moment vector)
$v_0 \leftarrow 0$ (Initialize 2^{nd} moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
  $t \leftarrow t + 1$
  $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
  $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
  $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
  $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
  $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
  $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

Adam 的实现

- https://github.com/pytorch/pytorch/blob/master/torch/optim/adam.py

AdamW

Algorithm 2 Adam with L_2 regularization and Adam with decoupled weight decay (AdamW)

given $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, \lambda \in \mathbb{R}$
initialize time step $t \leftarrow 0$, parameter vector $\boldsymbol{\theta}_{t=0} \in \mathbb{R}^n$, first moment vector $\boldsymbol{m}_{t=0} \leftarrow \boldsymbol{0}$, second moment vector $\boldsymbol{v}_{t=0} \leftarrow \boldsymbol{0}$, schedule multiplier $\eta_{t=0} \in \mathbb{R}$
repeat
  $t \leftarrow t + 1$
  $\nabla f_t(\boldsymbol{\theta}_{t-1}) \leftarrow \text{SelectBatch}(\boldsymbol{\theta}_{t-1})$ ▷ select batch and return the corresponding gradient
  $\boldsymbol{g}_t \leftarrow \nabla f_t(\boldsymbol{\theta}_{t-1}) + \lambda \boldsymbol{\theta}_{t-1}$
  $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\boldsymbol{g}_t$ ▷ here and below all operations are element-wise
  $\boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1 - \beta_2)\boldsymbol{g}_t^2$
  $\hat{\boldsymbol{m}}_t \leftarrow \boldsymbol{m}_t / (1 - \beta_1^t)$ ▷ β_1 is taken to the power of t
  $\hat{\boldsymbol{v}}_t \leftarrow \boldsymbol{v}_t / (1 - \beta_2^t)$ ▷ β_2 is taken to the power of t
  $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$ ▷ can be fixed, decay, or also be used for warm restarts
  $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta_t \left(\alpha \hat{\boldsymbol{m}}_t / (\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon) + \lambda \boldsymbol{\theta}_{t-1} \right)$
until *stopping criterion is met*
return optimized parameters $\boldsymbol{\theta}_t$

Weight Decay 和 Normalization

补充阅读材料

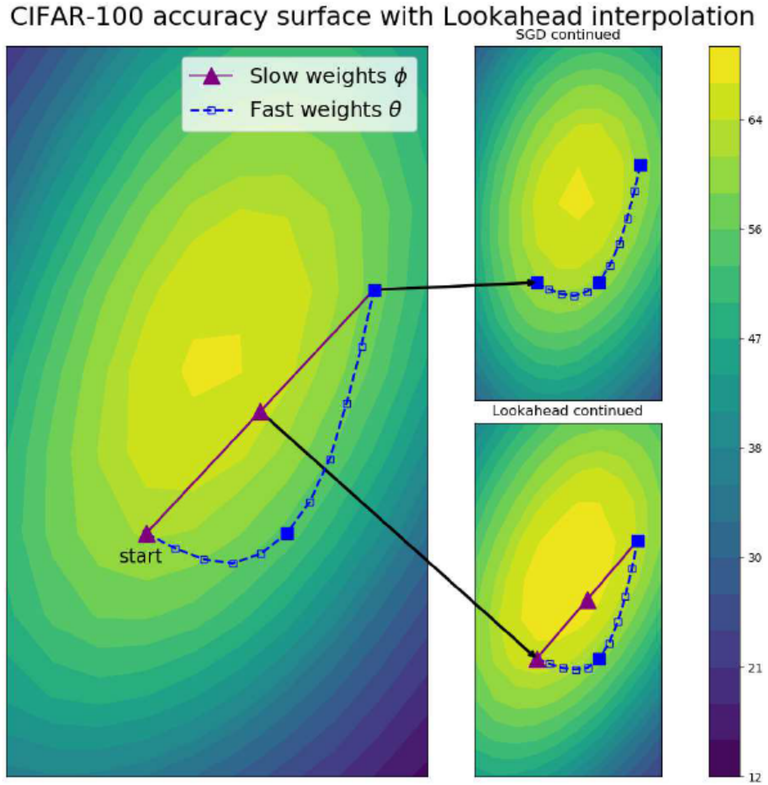
- Adam：https://arxiv.org/pdf/1412.6980.pdf
- AdamW：https://arxiv.org/pdf/1711.05101.pdf
- Weight Decay 和 Batch Normalization 之间的关系：
  - https://blog.janestreet.com/l2-regularization-and-batch-norm/

4.8 优化器：Lookahead，Radam 和 Lamb

内容综述

- Lookahead
- RAdam
- Lamb

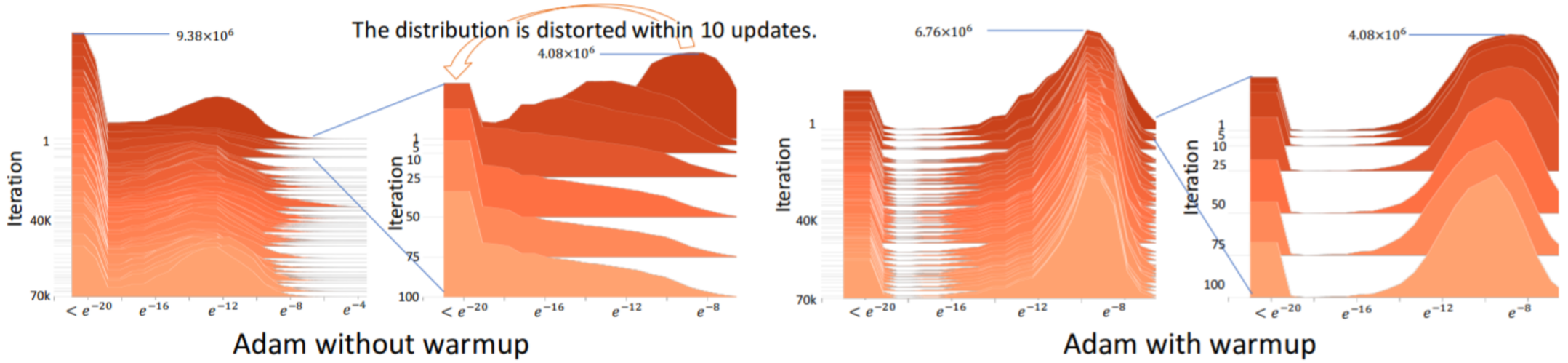
Lookahead



Algorithm 1 Lookahead Optimizer:

Require: Initial parameters ϕ_0, objective function L
Require: Synchronization period k, slow weights step size α, optimizer A
for $t = 1, 2, \ldots$ **do**
  Synchronize parameters $\theta_{t,0} \leftarrow \phi_{t-1}$
  for $i = 1, 2, \ldots, k$ **do**
    sample minibatch of data $d \sim \mathcal{D}$
    $\theta_{t,i} \leftarrow \theta_{t,i-1} + A(L, \theta_{t,i-1}, d)$
  end for
  Perform outer update $\phi_t \leftarrow \phi_{t-1} + \alpha(\theta_{t,k} - \phi_{t-1})$
end for
return parameters ϕ

RAdam


The distribution is distorted within 10 updates.
9.38×10^6
4.08×10^6
6.76×10^6
4.08×10^6
Iteration
1
40K
70k
1
5
10
25
50
75
100
< e^−20
e^−16
e^−12
e^−8
e^−4
Adam without warmup
Adam with warmup

Lamb

Algorithm 1 LARS

Input: $x_1 \in \mathbb{R}^d$, learning rate $\{\eta_t\}_{t=1}^T$, parameter $0 < \beta_1 < 1$, scaling function ϕ, $\epsilon > 0$
Set $m_0 = 0$
for $t = 1$ **to** T **do**
  Draw b samples S_t from $\mathbb{P}$
  Compute $g_t = \frac{1}{|\mathcal{S}_t|} \sum_{s_t \in \mathcal{S}_t} \nabla \ell(x_t, s_t)$
  $m_t = \beta_1 m_{t-1} + (1 - \beta_1)(g_t + \lambda x_t)$
  $x_{t+1}^{(i)} = x_t^{(i)} - \eta_t \frac{\phi(\|x_t^{(i)}\|)}{\|m_t^{(i)}\|} m_t^{(i)}$ for all $i \in [h]$
end for

Algorithm 2 LAMB

Input: $x_1 \in \mathbb{R}^d$, learning rate $\{\eta_t\}_{t=1}^T$, parameters $0 < \beta_1, \beta_2 < 1$, scaling function ϕ, $\epsilon > 0$
Set $m_0 = 0$, $v_0 = 0$
for $t = 1$ **to** T **do**
  Draw b samples S_t from $\mathbb{P}$.
  Compute $g_t = \frac{1}{|\mathcal{S}_t|} \sum_{s_t \in \mathcal{S}_t} \nabla \ell(x_t, s_t)$.
  $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$
  $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
  $m_t = m_t / (1 - \beta_1^t)$
  $v_t = v_t / (1 - \beta_2^t)$
  Compute ratio $r_t = \frac{m_t}{\sqrt{v_t} + \epsilon}$
  $x_{t+1}^{(i)} = x_t^{(i)} - \eta_t \frac{\phi(\|x_t^{(i)}\|)}{\|r_t^{(i)} + \lambda x_t^{(i)}\|} (r_t^{(i)} + \lambda x_t^{(i)})$
end for

一些 trick

- 大部分时间 AdamW 是一个很好的优化器
- Lookahead 有很大概率提升准确性
- Radam 在一些情况下可以提升准确性
- Lamb 是一个非常危险的优化器
- 可以将一些优化器进行结合
  - https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer

补充材料

- Lookahead：https://arxiv.org/pdf/1907.08610.pdf
- Radam：https://arxiv.org/pdf/1908.03265.pdf
- Lamb： https://arxiv.org/pdf/1904.00962.pdf

4.9 如何通过多重 loss 提升模型的准确率

内容综述

- loss 的设计
- 多重 loss 的使用
- Focal Loss
- 在 Huggingface Transformer 中增加多重 loss

loss 的设计

- 对于非标准任务，loss 的设计是至关重要的
- 对于标准任务，通常可以通过叠加多个 loss 实现训练的提升

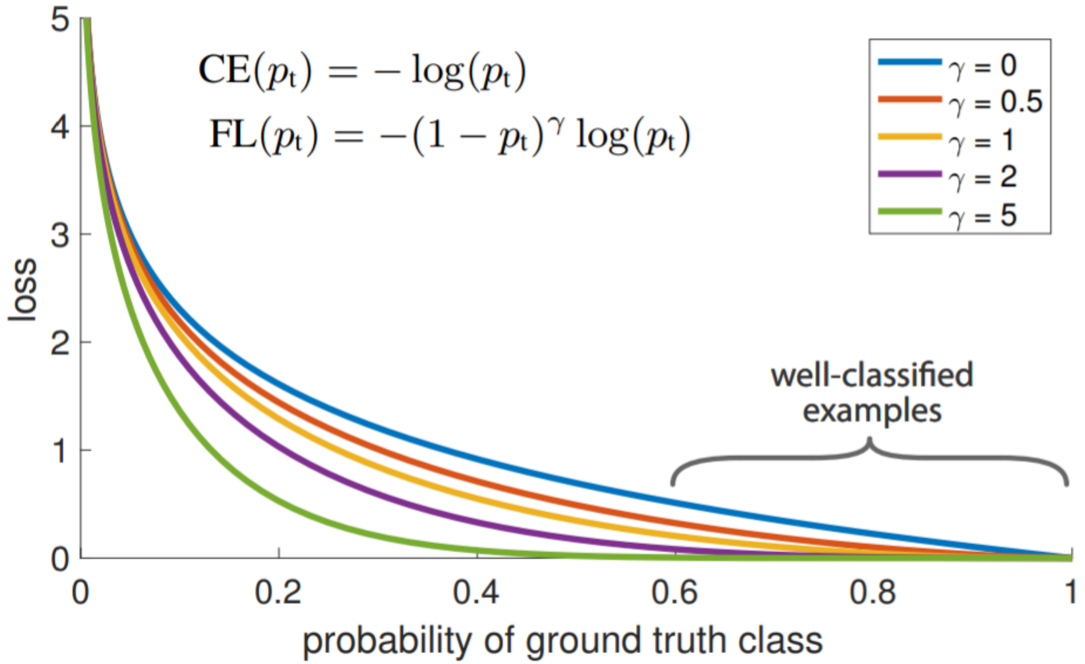
神经网络训练的基本目标和多重 loss 的使用

- 基本目标：
  - 快速收敛
  - 精度提高
- 理论上 -> 两者共存
- 实际上 -> 常常矛盾

神经网络训练的基本目标和多重 loss 的使用

- 多重 loss 的使用
  - 初期 -> 提高收敛速度
  - 后期 -> 提高模型精度
- Intuition -> 不同的 loss 擅长于捕捉不同的损失
- 重点和难点：不同 loss 之间的比例

Focal Loss


$\mathrm{CE}(p_t) = -\log(p_t)$
$\mathrm{FL}(p_t) = -(1-p_t)^{\gamma}\log(p_t)$
γ = 0
γ = 0.5
γ = 1
γ = 2
γ = 5
well-classified examples
loss
probability of ground truth class
0
1
2
3
4
5
0
0.2
0.4
0.6
0.8
1

在 Huggingface Transformer 中增加 loss

补充材料

- Focal Loss Paper：https://arxiv.org/pdf/1708.02002.pdf
- Focal Loss 实现：https://github.com/clcarwin/focal_loss_pytorch

扫码试看/订阅

《NLP 实战高手课》视频课程