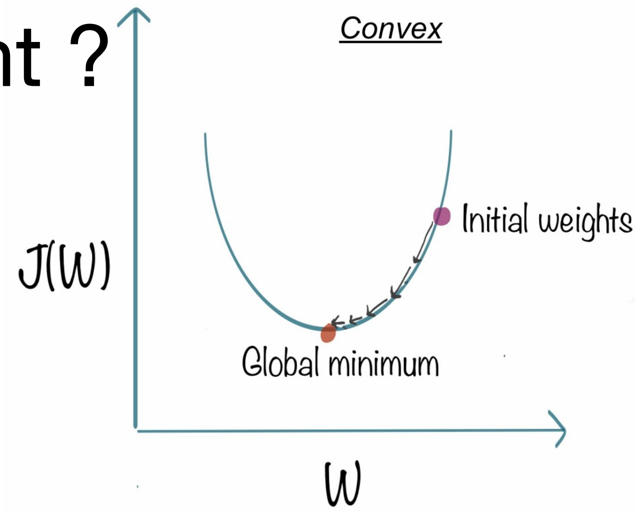
Loss Function Summary

Group 12
2021/03/07

Laksara Chandrasiri E/15/048 - e15048@eng.pdn.ac.lk
Sewwandi Nisansala E/15/243 - e15243@eng.pdn.ac.lk
Sonali Prasadika E/15/271 - e15271@eng.pdn.ac.lk
Rashmi Ireshe E/15/363 - r.i.thilakarathne@eng.pdn.ac.lk

What is a lost function ?

Why it’s so important ?



Paper

Delving Deep into Rectifiers:

Surpassing Human-Level Performance on ImageNet Classification

Authors:

Kaiming He
Xiangyu Zhang
Shaoqing Ren
Jian Sun

What are Rectified Linear Unit (ReLU) ?

Definition

Generic form of Rectifier Linear Function

$$f(x_i) = \begin{cases} x_i, & \text{if } x_i > 0 \\ a_i x_i, & \text{if } x_i \leq 0 \end{cases}$$

ReLU: when $a_i = 0$

$f(x_i) = max(0, x_i)$

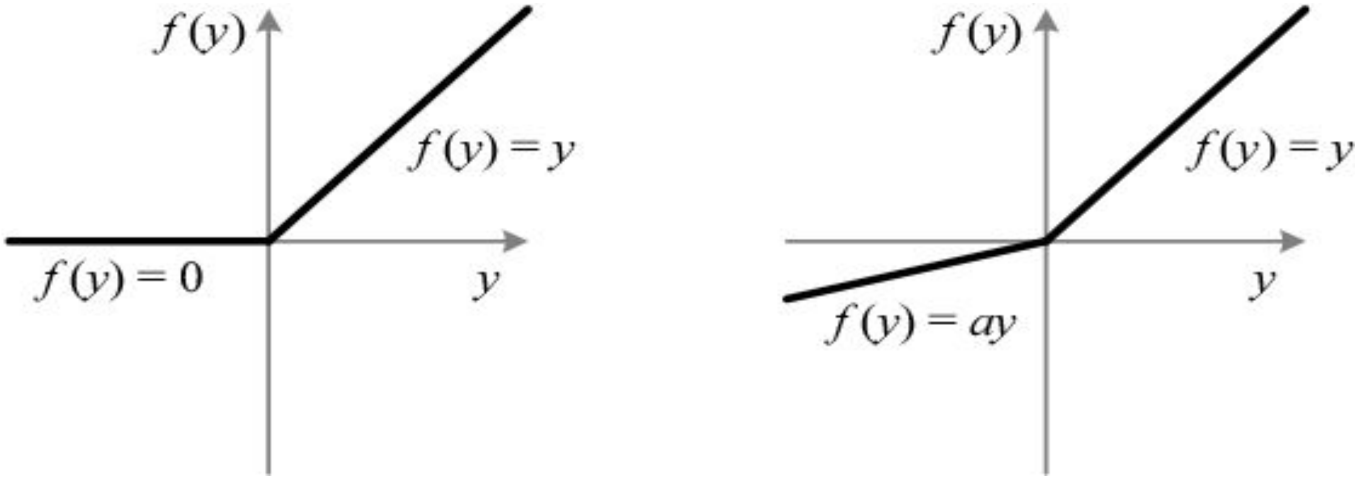
PReLU: when a_i is a learnable parameter

$f(x_i) = max(0, x_i) + a_i min(0, x_i)$

LReLU: Leaky ReLU, when $a_i = 0$

$f(x_i) = max(0, x_i) + 0.01 min(0, x_i)$

ReLU Vs PReLU



ReLU vs. PReLU. For PReLU, the coefficient of the negative part is not constant and is adaptively learned.

Optimization

- Trained using backpropagation
- Optimized simultaneously with other layers

The gradient of a_i for one layer:

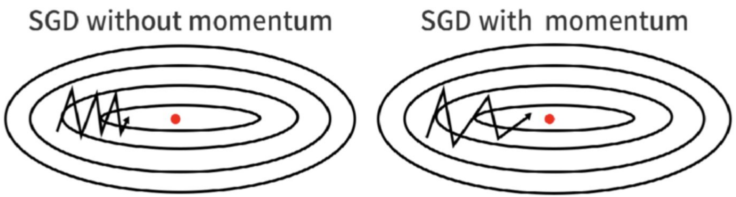
$$\frac{\partial \varepsilon}{\partial a_i} = \sum_{y_i} \frac{\partial \varepsilon}{\partial f(y_i)} \frac{\partial f(y_i)}{\partial a_i}$$

$\partial \varepsilon$: Objective function

$\frac{\partial \varepsilon}{\partial f(y_i)}$: Gradient propagated from the deeper layer

$\frac{\partial f(y_i)}{\partial a_i}$: Gradient of the activation $= \begin{cases} 0, & if \;\; y_i > 0 \\ y_i, & if \;\; y_i \leq 0 \end{cases}$

Momentum



Adopt The Momentum Method When Updating -

$$\Delta a_i := \mu \Delta a_i + \epsilon \frac{\partial \varepsilon}{\partial a_i}$$

μ : Momentum

ϵ : Learning Rate

Initial a_i : 0.25

- For the channel-shared variant, the gradient of a_i

$$\frac{\partial \varepsilon}{\partial a} = \sum_i \sum_{y_i} \frac{\partial \varepsilon}{\partial f(y_i)} \frac{\partial f(y_i)}{\partial a}$$

$\sum_i$: sums over all channels of the layer

Paper

ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION

Authors:

Diederik P. Kingma

Jimmy Lei Ba

The name Adam is derived from adaptive moment estimation

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 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd 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’S UPDATE RULE

Adam’s update rule is its careful choice of stepsizes

$$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon) \text{ (Update parameters)}$$

Step size have two upper bounds

$$|\Delta_t| \leq \alpha \cdot (1 - \beta_1) / \sqrt{1 - \beta_2} \text{ in the case } (1 - \beta_1) > \sqrt{1 - \beta_2},$$

and

$$|\Delta_t| \leq \alpha$$

Best Default values by the Authors

α - 0.001

β1 - 0.9

β2 - 0.999

ϵ - 10^{-8}

INITIALIZATION BIAS CORRECTION

Focused on the Initial steps of the algorithm

$$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \text{ (Update biased first moment estimate)}$$
$$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \text{ (Update biased second raw moment estimate)}$$

Initial steps m(t-1) and v(t-1) values are almost zero (m0 = 0, v0 = 0)

mt and vt are heavily biased to (1- **β**).gt on initial algorithm

$$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t) \text{ (Compute bias-corrected first moment estimate)}$$
$$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t) \text{ (Compute bias-corrected second raw moment estimate)}$$

Correcting the moving average by removing the bias from moving average

Paper

Gradient Based Learning Applies to Document Recognition

Authors:
YANN LECUN
L´EON BOTTOU
YOSHUA BENGIO
PATRICK HAFFNER

E/15/048 Laksara

Maximum Likelihood Estimation Criterion

$$E(W) = \frac{1}{P}\sum_{p=1}^{P} y_{D^p}(Z^p, W)$$

E/15/048 Laksara

Two Properties

- Allow the parameters of the RBF to adapt, has a trivial, but totally unacceptable, solution

- There is no competition between the classes.

Improved Loss Function

$$E(W) = \frac{1}{P}\sum_{p=1}^{P}\left(y_{D^p}(Z^p, W) + \log\left(e^{-j} + \sum_{i} e^{-y_i(Z^p, W)}\right)\right).$$

Back Propagation to compute the gradient of loss function

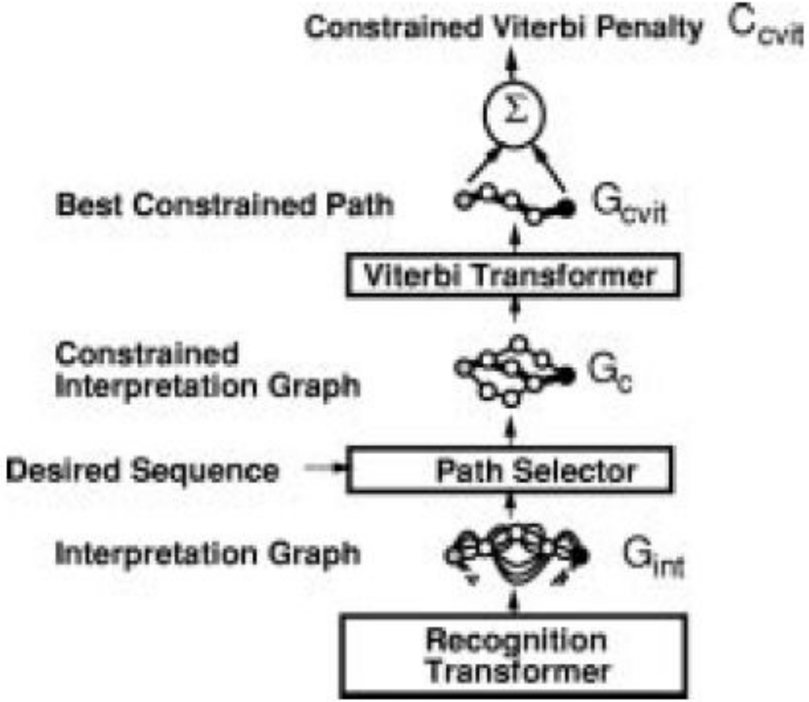


Fig. 19. Viterbi training GTN architecture for a character string recognizer based on HOS.

Paper

Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion

Authors:

Pascal Vincent
Hugo Larochelle
Isabelle Lajoie
Yoshua Bengio
Pierre-Antoine Manzagol

Loss Function $$L(\mathbf{x},\mathbf{z}) \propto -\log p(\mathbf{x}|\mathbf{z}).$$

$$L(x, z) = L_2(x, z) = C(s^2)||x-z||^2 \text{ -----------> } 1$$

Paper

Image Style Transfer Using Convolutional Neural Networks

Authors:
Leon A. Gatys
Alexander S. Ecker
Matthias Bethge

Source :https://rn-unison.github.io/articulos/style_transfer.pdf

Content Representation

$$\mathcal{L}_{\text{content}}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F_{ij}^{l} - P_{ij}^{l} \right)^2$$

$$\frac{\partial \mathcal{L}_{\text{content}}}{\partial F_{ij}^{l}} = \begin{cases} \left(F^{l} - P^{l} \right)_{ij} & \text{if } F_{ij}^{l} > 0 \\ 0 & \text{if } F_{ij}^{l} < 0 \, , \end{cases}$$

Style Representation

$$E_l = \frac{1}{4N_l^2M_l^2}\sum_{i,j}\left(G_{ij}^l - A_{ij}^l\right)^2$$

$$\mathcal{L}_{\text{style}}(\vec{a},\vec{x}) = \sum_{l=0}^{L} w_l E_l,$$

Total Loss Function

$$\mathcal{L}_{\text{total}}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{\text{content}}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{\text{style}}(\vec{a}, \vec{x})$$

Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning

Authors:
Yarin Gal
Zoubin Ghahramani

Dropout as a Bayesian Approximation

$$\mathcal{L}_{\text{dropout}} := \frac{1}{N}\sum_{i=1}^{N} E(\mathbf{y}_i, \widehat{\mathbf{y}}_i) + \lambda \sum_{i=1}^{L} \left(||\mathbf{W}_i||_2^2 + ||\mathbf{b}_i||_2^2 \right)$$

PyTorch: An Imperative Style, High-Performance Deep Learning Library

Authors:
- Adam Paszke
- Sam Gross
- Francisco Massa

NN Implementation

- Neural network architecture can be easily implemented with PyTorch.
- Neural network models are usually represented as classes that compose individual layers.

A custom layer used as a building block for a simple neural network

```python
class LinearLayer(Module):
    def __init__(self, in_sz, out_sz):
        super().__init__()
        t1 = torch.randn(in_sz, out_sz)
        self.w = nn.Parameter(t1)
        t2 = torch.randn(out_sz)
        self.b = nn.Parameter(t2)

    def forward(self, activations):
        t = torch.mm(activations, self.w)
        return t + self.b
```

```python
class FullBasicModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv = nn.Conv2d(1, 128, 3)
        self.fc = LinearLayer(128, 10)

    def forward(self, x):
        t1 = self.conv(x)
        t2 = nn.functional.relu(t1)
        t3 = self.fc(t1)
        return nn.functional.softmax(t3)
```

Simplified training of a generative adversarial networks

```
discriminator = create_discriminator()
generator = create_generator()
optimD = optim.Adam(discriminator.parameters())
optimG = optim.Adam(generator.parameters())

def step(real_sample):
  # (1) Update Discriminator
  errD_real = loss(discriminator(real_sample), real_label)
  errD_real.backward()
  fake = generator(get_noise())
  errD_fake = loss(discriminator(fake.detach(), fake_label)
  errD_fake.backward()
  optimD.step()
  # (2) Update Generator
  errG = loss(discriminator(fake), real_label)
  errG.backward()
  optimG.step()
```

References

1) Vincent, Pascal, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." *Journal of machine learning research* 11, no. 12 (2010).
2) LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86, no. 11 (1998): 2278-2324.
3) Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "Image style transfer using convolutional neural networks." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2414-2423. 2016.
4) Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen et al. "Pytorch: An imperative style, high-performance deep learning library." *arXiv preprint arXiv:1912.01703* (2019).
5) Gal, Yarin, and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." In *international conference on machine learning*, pp. 1050-1059. PMLR, 2016.
6) Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).
7) He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." In Proceedings of the IEEE international conference on computer vision, pp. 1026-1034. 2015.

Thank You !!!