CS 480/680

8/3/26

Lecture 14

Adaptive η

Conv Nets

Res Nets

Q2 on 3/10
11:31 in class

LV - posted

DC 2502 ←

Sol 5 - Monday

Lecture Notes IV – Neural Networks, Part 2

Marina Meilă
mmp@uwaterloo.ca

With Thanks to Pascal Poupart & Gautam Kamath
Cheriton School of Computer Science
University of Waterloo

February 8, 2026

Can we mimic 2-nd order methods "cheaply"? (T,L)

- "2-nd derivative" is Hessian $\frac{\partial^2 \mathcal{L}}{\partial \mathbb{W}^2}$ a $p \times p$ matrix
- We want to get the benefits of 2-nd order in $\mathcal{O}(p)$ time.[4]
- Let $g \in \mathbb{R}^p$ denote a gradient or stochastic gradient (p is still the number of parameters we are training)
- **Momentum (Heavy ball method)**

$$\mathbb{W}^{t+1} \leftarrow \mathbb{W}^t - \gamma \eta g^t + (1-\gamma) \underbrace{(\mathbb{W}^t - \mathbb{W}^{t-1})}_{\text{previous step}} \tag{27}$$

- (many variations exist, e.g. Nesterov method)
- Adaptive learning rates (coming next)

[4] The factor n or n' is ignored, because it does not affect what we do.

Adaptive Learning Rates (LR) – **S,T**

η = learning rate

- Typical for SGD
- The LR adapts over iteration t and parameter w_i
- Goal η "normalized" with a factor that avoids large updates

- $g_i^t = \frac{\partial \mathcal{L}}{\partial w_i}(\mathbb{W}^t)$
- Let $G_{i,t} = \sum_{t'=1}^{t}(g_i^{t'})^2$ = sum of squares of gradients for weight i
- AdaGrad algorithm

$$w_i^{t+1} \leftarrow w_i^t - \frac{\eta}{\sqrt{G_{i,t} + \epsilon}} g_i^t. \tag{28}$$

new $\eta = \eta^t$

- Problem stepsize can't increase if needed

$\eta^t = \frac{\eta}{\text{something}(t)}$

why?

Theory of SGD

$\eta^t \sim \frac{1}{t}$

$G_{i,t} = \sum_{t'=1}^{t}(g_i^{t'})^2$ ← for weight i in $\mathbb{W}$

↗ with t

Adaptive Learning Rates (LR) – **S,T**

- Typical for SGD
- The LR adapts over iteration t and parameter w_i
- Goal η "normalized" with a factor that avoids large updates

- $g_i^t = \frac{\partial \mathcal{L}}{\partial w_i}(\mathbb{W}^t)$
- Let $G_{i,t} = \sum_{t=1}^{t}(g_i^{t'})^2$ = sum of squares of gradients for weight i
- AdaGrad algorithm

$$w_i^{t+1} \leftarrow w_i^t - \frac{\eta}{\sqrt{G_{i,t} + \epsilon}} g_i^t. \tag{28}$$

- Problem stepsize can't increase if needed
- RMSProp "momentum" (forgetting) $G_{i,t} = 0.9G_{i,t-1} + 0.1(g_i^t)^2$

Adaptive Learning Rates (LR) – **S,T**

- Typical for SGD
- The LR adapts over iteration t and parameter w_i
- Goal η "normalized" with a factor that avoids large updates

- $g_i^t = \frac{\partial \mathcal{L}}{\partial w_i}(\mathbb{W}^t)$
- Let $G_{i,t} = \sum_{t=1}^{t}(g_i^{t'})^2$ = sum of squares of gradients for weight i
- AdaGrad algorithm

$$w_i^{t+1} \leftarrow w_i^t - \frac{\eta}{\sqrt{G_{i,t}+\epsilon}} g_i^t . \quad (28)$$

- Problem stepsize can't increase if needed
- RMSProp "momentum" (forgetting) $G_{i,t} = 0.9 G_{i,t-1} + 0.1 (g_i^t)^2$
- Adam momentum and RMSProp
  - Hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$

$$m_i^t = \beta_1 m_i^{t-1} + (1-\beta_1) g_i^t \quad \text{momentum for } g \quad (29)$$

$$v_i^t = \beta_2 v_i^{t-1} + (1-\beta_2)(g_i^t)^2 \quad \text{RMSProp, momementum for } g^2 \quad (30)$$

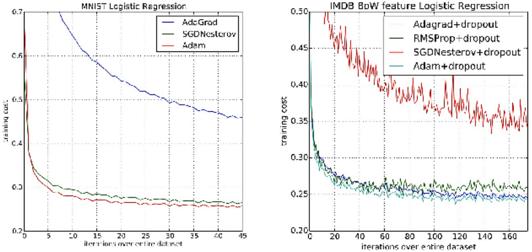
$$\hat{m}_i^t = \frac{m_i^t}{1-\beta_1^t} \quad \hat{v}_i^t = \frac{v_i^t}{1-\beta_2^t} \quad \text{decay} \quad (31)$$

$$w_i^t \leftarrow w_i^{t-1} - \frac{\eta}{\sqrt{\hat{v}_i^t}+\epsilon} \hat{m}_i^t \quad (32)$$

Comparisons – AdaGrad, RMSProp, ADAM

Empirical Comparison

- From Kingma & Ba (ICLR-2015):



UNIVERSITY OF WATERLOO

NN Architectures

Lecture Notes V – Residual and Convolutional Neural Networks

Marina Meilă
mmp@uwaterloo.ca

With Thanks to Pascal Poupart & Gautam Kamath
Cheriton School of Computer Science
University of Waterloo

February 8, 2026

Convolutional networks (Convnets)

Residual networks (Resnets)

Some (convolutional) neural network breakthroughs

Reading HTF Ch.: 11.3 Neural networks, Murphy Ch.: (16.5 neural nets), Bach Ch.: –, Deep Learning Book (Goodfellow, Bengio, Courville) 6.1-4, ResNet 7.6, ConvNet 9., Autoencoders 14.1, Dive Into Deep Learning 4.1-4.3.

ConvNets – Convolutional Networks

- **discrete convolution** let $f, g : \mathbb{Z} \to \mathbb{R}$
  $\mathbb{Z} =$ all integers

$$(f * g)(t) \; = \sum_{i \in \mathbb{Z}} f(t-i) g(i) \tag{1}$$

- convolution as **Toeplitz** matrix vector multiplication
- in ConvNets, $\mathbb{Z}$ is replaced by $1 : m$, f is **padded with 0's**
  - g is a (smoothing) kernel
  - i.e. $g(i) = g(-i) > 0$ and $|\operatorname{supp} g| = 2s + 1 \ll m$, $\sum_i g(i) = 1$
- Convolutional layer $f \leftarrow x$ input, $g \leftarrow w$ weights, s output

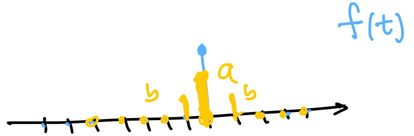
$$s(t) = \sum_{i=t-s}^{t+s} w_i s(t-i) \tag{2}$$

- Pooling
  a symmetric function like $\max, \sum, \ldots$

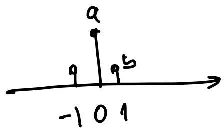
$f, g : \mathbb{R} \to \mathbb{R}$

$h = f * g \quad h(x) = \int_{-\infty}^{\infty} f(t) g(x-t) dt$

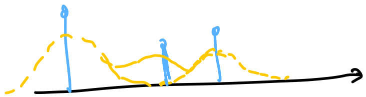
Ex1



g = filter



f*g = ?



"smooths f"

$$(f*g)(t) = \sum_i f(t-i)\,g(i)$$

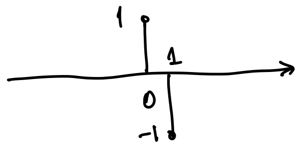
$i \in \{0,1,-1\}$

$$= f(t)\cdot g(0) + f(t-1)\cdot g(1) + f(t+1)\,g(-1) =$$

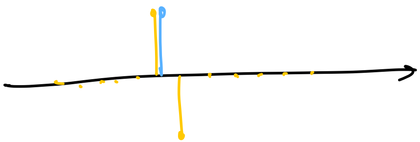
$$= a\,f(t) + b\,\big(f(t-1) + f(t+1)\big)$$

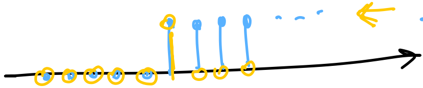
$a + 2b = 1$

Ex2



$$(f*g)(t) = f(t) - f(t-1)$$





← f is "step fction"

↑ location of step

Discrete convolution

- Discrete convolution

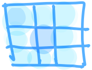
$$y(i) = \sum_{t=-\infty}^{\infty} x(t)w(i-t)$$

z
values of (hidden) units
g = weights

- Multidimensional convolution

$$y(i,j) = \sum_{t_1=-\infty}^{\infty} \sum_{t_2=-\infty}^{\infty} x(t_1,t_2)w(i-t_1,j-t_2)$$

images pixel i,j



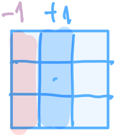
w = smoothing

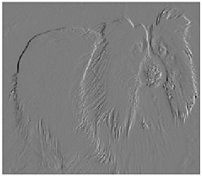
UNIVERSITY OF WATERLOO

Example: Edge Detection

- Consider a grey scale image
- Detect vertical edges: $y(i,j) = x(i,j) - x(i-1,j)$

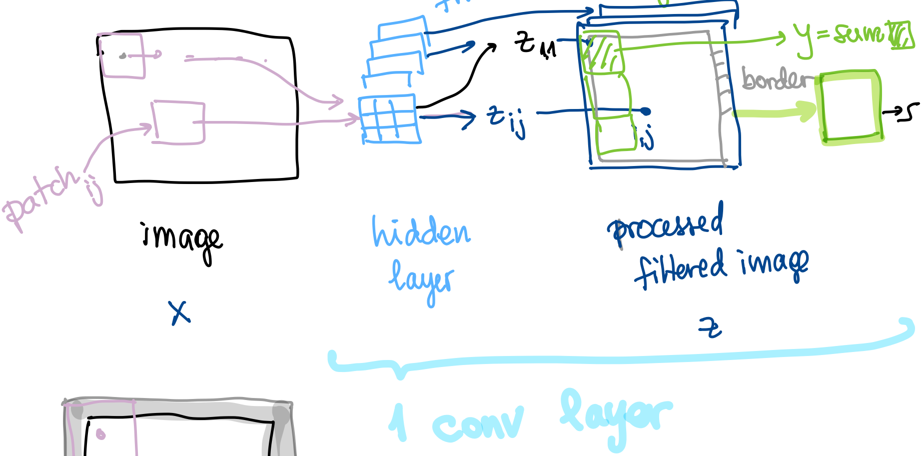
$$w(i-t_1, j-t_2) = \begin{cases} 1 & t_1 = i, t_2 = j \\ -1 & t_1 = i-1, t_2 = j \\ 0 & \text{otherwise} \end{cases}$$

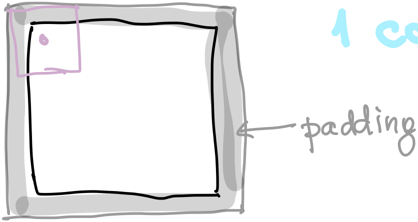




UNIVERSITY OF WATERLOO

Convolution for feature extraction





Pooling

- Pooling: **commutative** mathematical operation that combines several units
- Examples:
  - max, sum, product, average, Euclidean norm, etc.

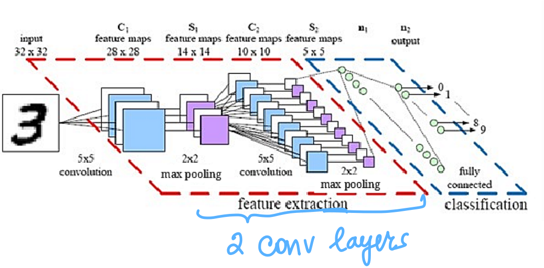
- Commutative property (order does not matter):

"symmetric" (if >2 arguments)

Ex.: $\max(a, b) = \max(b, a)$

UNIVERSITY OF WATERLOO

Example: Digit Recognition

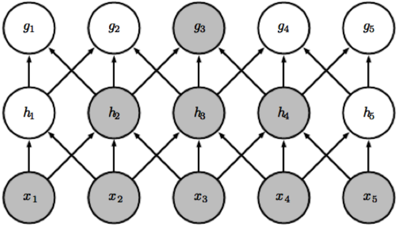


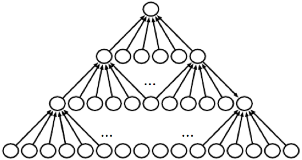
Parameters

- **# of filters:** integer indicating the # of filters applied to each window. = patch of image
- **kernel size:** tuple (width, height) indicating the size of the window. e.g. 5x5
- **Stride:** tuple (horizontal, vertical) indicating the horizontal and vertical shift between each window.
- **Padding:** "valid" or "same". Valid indicates no input padding. (border) Same indicates that the input is padded with a border of zeros to ensure that the output has the same size as the input.

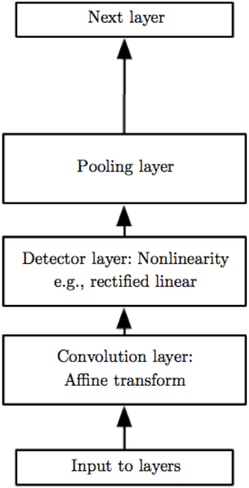
Benefits

- Sparse interactions
  - Fewer connections
- Parameter sharing
  - Fewer weights
- Locally equivariant representation
  - Locally invariant to translations
  - Handle inputs of varying length





from www.deeplearningbook.org Chapter 9



Training

- Convolutional neural networks are trained in the same way as other neural networks
  - E.g., backpropagation

- Weight sharing:
  - Combine gradients of shared weights into a single gradient

Architecture design

- What is the preferred filter size?
- VGG (Visual Geometry Group at Oxford, 2014): stack of small filters is often preferred to a single large filter
  - Fewer parameters
  - Deeper network
- Picture

Resnets – Residual networks

Idea What is the "simplest" input-output function? $f_0(x) = x$

- Hence, a NN layer should learn the difference w.r.t. identity f_0

$$x_{l+1} = B_l \phi(W_l x_l) + x_l \tag{3}$$

Generalization DenseNet

- Layer l gets inputs from $l-1, l-2, \ldots$

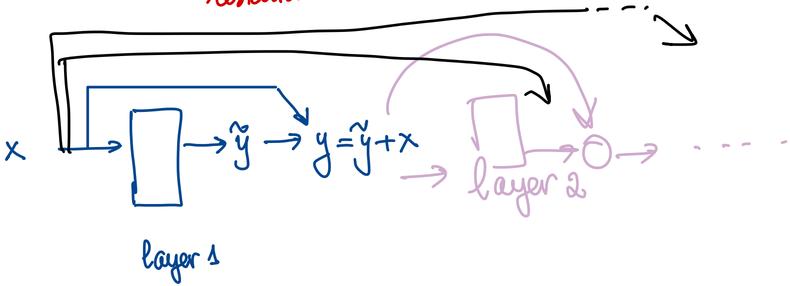
Residual Networks

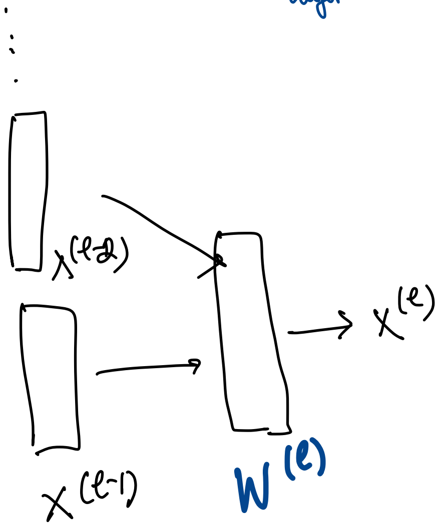
- Problem: very deep networks can perform worse than shallower networks (due to local optima & other stationary non-optimal points)
- Solution [He et al., 2015]: introduce **residual connections** (a.k.a. skip connections) to make blocks optional
- Picture:

Res Nets

Idea "Neutral" nn "$f(x) = x$"

$$f(x) = x + \underbrace{\text{neural net}(x)}_{\text{residual}} \iff \text{learn } f(x) - x$$





"Dense Net" x input to layer 1, 2, ⋯ K

⇕

$\ell - K, \cdots \ell - 1 \Longrightarrow$ input to layer ℓ