Basics of deep learning
part 2
By
Christian Desrosiers
How to train the network?
Forward pass

\[ \vec{x} \]
Forward pass

\[ W^{[0]} \bar{x} \]
Forward pass

\[ \sigma\left(W^{[0]}\bar{x}\right) \]
Forward pass

\[ W^{[1]} \sigma(W^{[0]} \bar{x}) \]
Forward pass

\[ \sigma(W^{[1]} \sigma(W^{[0]} \tilde{x})) \]
Forward pass

\[ \tilde{w}^T \sigma(W^{[1]} \sigma(W^{[0]} \tilde{x})) \]
Forward pass

\[ y_w(\tilde{x}) = \sigma(\tilde{w}^T \sigma(W^{[1]} \sigma(W^{[0]} \tilde{x}))) \]
Forward pass

\[ l(y_W(\bar{x}), t) = -t \ln(y_W(\bar{x})) - (1 - t) \ln(1 - y_W(\bar{x})) \]
How to optimize the network?

1- From

\[ W = \arg \min_W = \sum_{n=1}^N l(y_W(\tilde{x}_n), t_n) + \lambda R(W) \]

Choose a regularization function

\[ R(W) = \|W\|_1 \text{ or } \|W\|_2 \]
How to optimize the network?

2- Choose a loss $l(y_w(\tilde{x}_n), t_n)$ for example

- Hinge loss
- Cross entropy

Do not forget to adjust the output layer with the loss you have choosen.

$\text{cross entropy} \Rightarrow \text{Softmax}$
How to optimize the network?

3- Compute the gradient of the loss with respect to each parameter

\[
\frac{\partial}{\partial W_{a,b}^{[c]}} \left( \sum_{n=1}^{N} l(y_W(\tilde{x}_n), t_n) + \lambda R(W) \right)
\]

and launch a gradient descent algorithm to update the parameters.

\[
W_{a,b}^{[c]} := W_{a,b}^{[c]} - \eta \frac{\partial}{\partial W_{a,b}^{[c]}} \left( \sum_{n=1}^{N} l(y_W(\tilde{x}_n), t_n) + \lambda R(W) \right)
\]
How to optimize the network?

\[
y_w(\bar{x}) = \sigma(\bar{w}^{[2]} \sigma(W^{[1]} \sigma(W^{[0]} \bar{X})))
\]

\[
\partial \left( \sum_{n=1}^{N} l(y_w(\bar{x}_n), t_n) + \lambda R(W) \right)_{\partial W^{[c]}_{a,b}}
\]
\[
y_W(\bar{x}) = \sigma(\tilde{w}^T \sigma(W[1] \sigma(W[0] \bar{x})))
\]
\[
l(y_W(\bar{x}), t) = -t \ln(y_W(\bar{x})) - (1 - t) \ln(1 - y_W(\bar{x}))
\]
$A = W^{[0]} \tilde{x}$
\[ A = W^{[0]} \bar{x} \]
\[ B = \sigma(A) \]
$A = W^{[0]} \tilde{x}$

$B = \sigma(A)$

$C = W^{[1]} B$
\[ A = W^{[0]} \tilde{x} \]
\[ B = \sigma(A) \]
\[ C = W^{[1]} B \]
\[ D = \sigma(C) \]
\[ A = W^{[0]} \ddot{x} \]

\[ B = \sigma(A) \]

\[ C = W^{[1]} B \]

\[ D = \sigma(C) \]

\[ E = \tilde{w}^T D \]
\[ A = W^{[0]} \tilde{x} \]
\[ B = \sigma(A) \]
\[ C = W^{[1]} B \]
\[ D = \sigma(C) \]
\[ E = \tilde{w}^T D \]
\[ y_w(\tilde{x}) = \sigma(\tilde{w}^T \sigma(W^{[1]} \sigma(W^{[0]} \tilde{x}))) \]
\[ y_w(\tilde{x}) = \sigma(E) \]
\[ A = W^{[0]} \tilde{x} \]
\[ B = \sigma(A) \]
\[ C = W^{[1]} B \]
\[ D = \sigma(C) \]
\[ E = \tilde{w}^T D \]
\[ y_w(\tilde{x}) = \sigma(E) \]
\[ l(y_w(\tilde{x}), t) \]
\[ y_w(\tilde{x}) = \sigma(\tilde{w}^T \sigma(W^{[1]} \sigma(W^{[0]} \tilde{x}))) \]
\[
\frac{\partial}{\partial W^{[l]}} \left( \sum_{n=1}^{N} l(y_{W}(\bar{x}_n), t_n) \right)
\]?

Chain rule
Chain rule recap

\[ f(u) = u^2 \]
\[ u(v) = 2v \]
\[ v(x) = \frac{1}{x} \]

\[ \frac{\partial f}{\partial x} = ? \]

\[ \frac{\partial f}{\partial x} = \frac{\partial f}{\partial u} \times \frac{\partial u}{\partial v} \times \frac{\partial v}{\partial x} \]

\[ = 2u \times 2 \times \left( -\frac{1}{x^2} \right) \]
\[ A = W^{[0]} \bar{x} \]
\[ B = \sigma(A) \]
\[ C = W^{[1]} B \]
\[ D = \sigma(C) \]
\[ E = \tilde{w}^T D \]
\[ y_w(\bar{x}) = \sigma(E) \]
\[ l(y_w(\bar{x}), t) \]

\[
\frac{\partial(l(y_w(\bar{x}), t))}{\partial W^{[0]}} = \frac{\partial(l(y_w(\bar{x}), t))}{\partial y_w(\bar{x})} \frac{\partial(y_w(\bar{x}))}{\partial E} \frac{\partial(E)}{\partial D} \frac{\partial(D)}{\partial C} \frac{\partial(C)}{\partial B} \frac{\partial(B)}{\partial A} \frac{\partial(A)}{\partial W^{[0]}}
\]
Back propagation

\[ \frac{\partial (l(y_w(\bar{x}), t))}{\partial W^{[0]}} = \frac{\partial (l(y_w(\bar{x}), t))}{\partial y_w(\bar{x})} \frac{\partial (y_w(\bar{x}))}{\partial E} \frac{\partial (E)}{\partial D} \frac{\partial (D)}{\partial C} \frac{\partial (C)}{\partial B} \frac{\partial (B)}{\partial A} \frac{\partial (A)}{\partial W^{[0]}} \]
Activation functions
Activation functions

\[ \sigma(x) = \frac{1}{1 + e^{-x}} \]

3 Problems:

- Gradient saturates when input is large
- Not zero centered
- \textit{exp}() is an expensive operation
Activation functions

**Tanh**($x$)

- Output is zero-centered 🎉
- Small gradient when input is large 😞

[LeCun et al., 1991]
Activation functions

ReLU(x) = max(0, x)

- Large gradient for \( x > 0 \)
- Super fast
- Output non centered at zero
- No gradient when \( x < 0 \)

ReLU(x)
(Rectified Linear Unit)

[Krizhevsky et al., 2012]
Activation functions

\[ \text{LReLU}(x) = \max(0.01x, x) \]

- no gradient saturation 😊
- Super fast 😊
- 0.01 is a hyperparameter 😞

[Mass et al., 2013]
[He et al., 2015]
Activation functions

\[ \text{PReLU}(x) = \max(\alpha x, x) \]

- no gradient saturation 😊
- Super fast 😊
- \(\alpha\) learned with backprop 😊

[Mass et al., 2013]
[He et al., 2015]
In practice

• By default, people use ReLU.

• Try Leaky ReLU / PReLU

• Try tanh but might be sub-optimal

• Do not use sigmoid except at the output of a 2 class net.
How to classify an image?
How to classify an image?
Many parameters (7850 in Layer 1)
Too many parameters
(655,370 in Layer 1)

256x256
Waaayy too many parameters
(160M in Layer 1)

256x256x256

https://ml4a.github.io/ml4a/neural_networks/
Full connections are too many

150-D input vector with 150 neurons in Layer 1 => 22,500 parameters!!
No full connection

150-D input vector with 150 neurons in Layer 1 => 450 parameters!!
Share weights

1- Learning convolution filters!
2- Small number of parameters = can make it deep!

150-D input vector with 150 neurons in Layer 1 ⇒ 3 parameters!!
\[ \sigma(-0.23 + 0.34 - 1.2) = 0.25 \]
\( \sigma(-0.17 + 0.8 - 0.84) = 0.45 \)
Convolution!
Each neuron of layer 1 is connected to 3x3 pixels, layer 1 has 9 parameters!!
Convolution operation

\[ F = \sigma(x \ast W^{[0]}) \]
5-feature map
convolution layer
K-feature map
convolution layer
Conv layer 1

POOLING LAYER
Pooling layer

**Goals**

- Reduce the spatial resolution of feature maps
- Lower memory and computation requirements
- Provide partial invariance to position, scale and rotation

2 Class CNN

\[ l(y_W(\tilde{x}), t) = -t \ln(y_W(\tilde{x})) - (1 - t) \ln(1 - y_W(\tilde{x})) \]
K Class CNN

Conv layer 1 → Pool layer 1 → Conv layer 2 → Pool layer 2 → Fully connected layers

\[ l(y_W(\tilde{x}), t) = -t \ln(y_W(\tilde{x})) - (1 - t) \ln(1 - y_W(\tilde{x})) \]
Nice example from the literature

Learn image-based characteristics

http://web.eecs.umich.edu/~honglak/icml09-ConvolutionalDeepBeliefNetworks.pdf
Batch processing
\( \hat{x} = (0.4, -1.0) \)
\( \hat{w} = [2.0, -3.6, 0.5] \)
\[ \vec{x} = (0.4, -1.0) \]
\[ \vec{w} = [2.0, -3.6, 0.5] \]

\[ \vec{w}^T \vec{x} = 2 - 3.6 \times 0.4 - 0.5 = -1.94 \]

\[ y_{\vec{w}}(\vec{x}) = \sigma(-1.94) = \frac{1}{1 + e^{1.94}} = 0.125 \]
\[ \tilde{x} = (0.4, -1.0) \]
\[ \tilde{w} = [2.0, -3.6, 0.5] \]

\[ y_{\tilde{w}}(\tilde{x}_a) = \sigma \begin{pmatrix} 2.0 & -3.6 & 0.5 \end{pmatrix} \begin{pmatrix} 1 \\ 0.4 \\ -1 \end{pmatrix} = 0.125 \]
\[ \tilde{x}_a = (0.4, -1.0), \tilde{x}_b = (-2.1, 3.0) \]
\[ \tilde{w} = [2.0, -3.6, 0.5] \]

\[
y_{\tilde{w}}(\tilde{x}_a) = \sigma \left( 2.0, -3.6, 0.5 \right) \begin{pmatrix} 1 \\ 0.4 \\ -1 \end{pmatrix} = 0.125
\]

\[
y_{\tilde{w}}(\tilde{x}_b) = \sigma \left( 2.0, -3.6, 0.5 \right) \begin{pmatrix} 1 \\ -2.1 \\ 3.0 \end{pmatrix} = 0.99
\]
Mini-batch processing

\[ y_\bar{w}(\bar{x}_a) = \sigma \left( \begin{pmatrix} 1 \\ 1 \\ 0.5 \end{pmatrix} \right) \begin{pmatrix} 2.0 & -3.6 & 0.5 \\ 0.4 & -2.1 \\ -1 & 3.0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ -1 \\ 3.0 \end{pmatrix} = (0.125, 0.99) \]
Mini-batch processing

\[
\begin{bmatrix}
1 & 1 & 1 & 1 \\
-1.1 & 0.5 & 0.4 & -2.1 \\
1.1 & -0.2 & -1.0 & 3.0 \\
\end{bmatrix}
\]

\[
\begin{align*}
\tilde{w}^T \tilde{x} &= 2.0 \\
&= 3.6 \\
&= 0.5 \\
\end{align*}
\]

\[
y_{\tilde{w}}(\tilde{x}_a) = (0.89, 0.2, 0.125, 0.99)
\]
Mini-batch processing
Mini-batch processing

Mini-batch of 4 images

4 predictions
Classical applications of ConvNets

Classification.
Classical applications of ConvNets

Classification.

Classical applications of ConvNets

Image segmentation
Classical applications of ConvNets

Image segmentation

Classical applications of ConvNets

Image segmentation

Fang Liu, Zhaoye Zhou, +3 authors, Deep convolutional neural network and 3D deformable approach for tissue segmentation in musculoskeletal magnetic resonance imaging. in Magnetic resonance in medicine 2018 DOI:10.1002/mrm.26841
Classical applications of ConvNets

Image segmentation

*Brain Tumor Segmentation with Deep Neural Networks*, Medical Image Analysis, Vol 35, 18-31
Classical applications of ConvNets

Localization
Classical applications of ConvNets

Localization

Merci