Fundamentals in Statistical Pattern Recognition
Classification and regression (4/4) – MLP (2/2)

Dr Sébastien Marcel
Senior researcher
www.idiap.ch/~marcel

Idiap Research Institute
Martigny, Switzerland
www.idiap.ch

May 22, 2013
Outline

1. Recap
2. Back-Propagation
3. Wrapping up
4. Next
Previous lectures

Classification and Regression (4/4)

- k-Nearest Neighbors (k-NN)
- Linear Regression (univariate)
- Linear Regression (multivariate) and the Gradient descent
- Logistic Regression (a classification algorithm !)
- Multi-Layer Perceptron (2/2)
Algorithms for classification and regression

Classification and Regression: Multi-Layer Perceptron

Regression: Multi-Layer Perceptron

Multi-Layer Perceptron (MLP)

MLP is a type of Artificial Neural Network (ANN)
Multi-Layer Perceptron

Putting together several Logistic Regression units ...

$$h^{\text{or}}_{\theta}(x)$$

$$h^{\text{and}}_{\theta}(x)$$

$$h_{\theta}(a)$$
Multi-Layer Perceptron

gives an MLP!
Multi-Layered Perceptron

Solves non-linearly separable problems (XOR)

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>MLP($x_1,x_2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$\approx 0$</td>
<td>$\approx 0$</td>
<td>$\approx 0$ (○)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$\approx 1$</td>
<td>$\approx 0$</td>
<td>$\approx 1$ (×)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$\approx 1$</td>
<td>$\approx 0$</td>
<td>$\approx 1$ (×)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$\approx 1$</td>
<td>$\approx 1$</td>
<td>$\approx 0$ (○)</td>
</tr>
</tbody>
</table>

The first layer performs a non-linear mapping so that data is linearly separable for the layer 2.
Multi-Layered Perceptron

Training with Gradient Descent

\[ \nabla_{w_{ji}} J(\Theta) \]

minimizing any (differentiable) cost \( J(\Theta) \) by computing the partial derivatives (Back-Propagation algorithm)
Outline

1 Recap

2 Back-Propagation
   • Multi-Layer Perceptron: notations
   • Multi-Layer Perceptron: back-propagation
   • Multi-Layer Perceptron: back-propagation in action
   • Multi-Layer Perceptron: training
   • Multi-Layer Perceptron: tricks

3 Wrapping up

4 Next
Outline

1 Recap

2 Back-Propagation
   - Multi-Layer Perceptron: notations
   - Multi-Layer Perceptron: back-propagation
   - Multi-Layer Perceptron: back-propagation in action
   - Multi-Layer Perceptron: training
   - Multi-Layer Perceptron: tricks

3 Wrapping up

4 Next
Multi-Layer Perceptron

Notations

- inputs: \( \mathbf{x} \in \mathbb{R}^{d+1} \) (\( x_0 = 1 \))
- weight \( w_{ji}^l \) between units \( i \) and \( j \) on layer \( l \)
- activation \( a_j^l = g(z) \) in unit \( j \) on layer \( l \)
  \[
g(z) = \frac{1}{1 + e^{-z}}
\]
- hidden units: \( \mathbf{a}^1 \in \mathbb{R}^{h+1} \) (\( a_0^1 = 1 \))
- output: \( \mathbf{a}^2 \in \mathbb{R}^{o+1} \) (\( a_0^1 \) not used)
Outline

1 Recap

2 Back-Propagation
   - Multi-Layer Perceptron: notations
   - Multi-Layer Perceptron: back-propagation
   - Multi-Layer Perceptron: back-propagation in action
   - Multi-Layer Perceptron: training
   - Multi-Layer Perceptron: tricks

3 Wrapping up

4 Next
Back-propagation

Pre-requisite 1: the “chain rule”

\[ a = f(b) \quad b = g(c) \]

\[ \frac{\partial a}{\partial c} = \frac{\partial a}{\partial b} \cdot \frac{\partial b}{\partial c} \]

\[ c \rightarrow b = g(c) \rightarrow a = f(b) \]
Pre-requisite 2: the “sum rule”

\[ a = f(b, c) \quad b = g(d) \quad c = h(d) \]

\[
\frac{\partial a}{\partial d} = \frac{\partial a}{\partial b} \frac{\partial b}{\partial d} + \frac{\partial a}{\partial c} \frac{\partial c}{\partial d} = \frac{\partial f(b, c)}{\partial b} g'(d) + \frac{\partial f(b, c)}{\partial c} h'(d)
\]
Back-propagation

Partial derivatives of the cost $J(\Theta)$

\[
\frac{\partial J(\Theta)}{\partial w^l_{ji}} = ?
\]
Back-propagation

Partial derivatives of the cost $J(\Theta)$

$$\frac{\partial J(\Theta)}{\partial w^l_{ji}} = \frac{\partial J(\Theta)}{\partial z^l_j} \frac{\partial z^l_j}{\partial w^l_{ji}}$$
Back-propagation

Partial derivatives of the cost $J(\Theta)$

$$\frac{\partial J(\Theta)}{\partial w^l_{ji}} = ?$$

$$= \frac{\partial J(\Theta)}{\partial z^l_j} \frac{\partial z^l_j}{\partial w^l_{ji}}$$

$$= \frac{\partial J(\Theta)}{\partial a^l_j} \frac{\partial a^l_j}{\partial z^l_j} \frac{\partial z^l_j}{\partial w^l_{ji}}$$

Applying the “chain rule” twice

$$a = f(b) \quad b = g(c)$$

$$\frac{\partial a}{\partial c} = \frac{\partial a}{\partial b} \frac{\partial b}{\partial c}$$
Back-propagation

Let's look at the terms separately

**Term 3:**
\[ \frac{\partial z_j^l}{\partial w_{ji}^l} \]

**Term 2:**
\[ \frac{\partial a_j^l}{\partial z_j^l} \]

**Term 1:**
\[ \frac{\partial J(\Theta)}{\partial a_j^l} \]
Term 3 \( \frac{\partial z^l_j}{\partial w^l_{ji}} \)

Reminder: \( z^l_j = (w^l_j)^\top a^{l-1} \)

\[
\frac{\partial z^l_j}{\partial w^l_{ji}} = \frac{\partial}{\partial w^l_{ji}} (w^l_j)^\top a^{l-1} = ?
\]
Term 3: \( \frac{\partial z'_j}{\partial w^l_{ji}} \)

Reminder: \( z'_j = (w'_j)^\top a'^{l-1} \)

\[
\begin{align*}
\frac{\partial z'_j}{\partial w^l_{ji}} &= \frac{\partial}{\partial w^l_{ji}} (w'_j)^\top a'^{l-1} \\
&= ? \\
&= a'^{l-1}_i
\end{align*}
\]
Term 2 $\frac{\partial a^l_j}{\partial z^l_j}$

Reminder: $a^l_j = g(z^l_j)$

$$\frac{\partial a^l_j}{\partial z^l_j} = \frac{\partial g(z^l_j)}{\partial z^l_j}$$

$$= ?$$
Back-propagation

Term 2 \( \frac{\partial a_j^l}{\partial z_j^l} \)

Reminder: \( a_j^l = g(z_j^l) \)

\[
\frac{\partial a_j^l}{\partial z_j^l} = \frac{\partial g(z_j^l)}{\partial z_j^l} = g'(z_j^l)
\]
Back-propagation

Term 1 \( \frac{\partial J(\Theta)}{\partial a^l_j} \)

2 cases: \( l = L \) (output units) or \( l \neq L \) (hidden units)

Output Unit \( (l = L) \)

Reminder: \( a^l_j = a^L_1 = h(\Theta)(x_n) \) (1 output for a given example \( n \))

\[
\frac{\partial J(\Theta)}{\partial a^l_j} = \frac{\partial}{\partial a^L_1} \left[ \frac{1}{2} \sum_{n=1}^{N} (h(\Theta)(x_n) - y_n)^2 \right]
\]

= ?
Recap

Back-Propagation

Wrapping up

Back-propagation

Term 1 $\frac{\partial J(\Theta)}{\partial a_j^l}$

2 cases: $l = L$ (output units) or $l \neq L$ (hidden units)

Output Unit ($l = L$)

Reminder: $a_j^l = a_1^L = h_\Theta(x_n)$ (1 output for a given example $n$)

$$
\frac{\partial J(\Theta)}{\partial a_j^l} = \frac{\partial}{\partial a_1^L} \left[ \frac{1}{2} \sum_{n=1}^{N} (h_\Theta(x_n) - y_n)^2 \right] \\
= \ ? \\
= h_\Theta(x_n) - y_n
$$
Back-propagation

Term 1 $\frac{\partial J(\Theta)}{\partial a_l^j}$

2 cases: $l = L$ (output units) or $l \neq L$ (hidden units)

Hidden Units ($l \neq L$)

$$\frac{\partial J(\Theta)}{\partial a_l^j} =$$
Back-propagation

**Term 1** \( \frac{\partial J(\Theta)}{\partial a_j^l} \)

2 cases: \( l = L \) (output units) or \( l \neq L \) (hidden units)

**Hidden Units** \( (l \neq L) \)

\[
\frac{\partial J(\Theta)}{\partial a_j^l} = \sum_{k=1}^{u(l+1)} \frac{\partial J(\Theta)}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial a_j^l}
\]

**Applying the “sum rule”**

\[
\frac{\partial a}{\partial d} = \frac{\partial a}{\partial b} \frac{\partial b}{\partial d} + \frac{\partial a}{\partial c} \frac{\partial c}{\partial d} = \frac{\partial f(b, c)}{\partial b} g'(d) + \frac{\partial f(b, c)}{\partial c} h'(d)
\]

with \( a = f(b, c), \ b = g(d) \) and \( c = h(d) \)
Back-propagation

Term 1 \( \frac{\partial J(\Theta)}{\partial a_j^l} \) \( (l \neq L) \)

\[
\frac{\partial J(\Theta)}{\partial a_j^l} = u(l+1) \sum_{k=1}^{K} \frac{\partial J(\Theta)}{\partial z_{k}^{l+1}} \frac{\partial z_{k}^{l+1}}{\partial a_j^l}
\]
Back-propagation

Term 1 $\frac{\partial J(\Theta)}{\partial a^l_j} (l \neq L)$

Applying the “chain rule”
Term 1 $\frac{\partial J(\Theta)}{\partial a_j^l}$ ($l \neq L$)

Let's denote $\delta_j^l = \frac{\partial J(\Theta)}{\partial a_j^l}$

$$\frac{\partial J(\Theta)}{\partial a_j^l} = \sum_{k=1}^{u(l+1)} \frac{\partial J(\Theta)}{\partial a_k^{l+1}} \frac{\partial a_k^{l+1}}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial a_j^l}$$

$$\delta_j^l = \sum_{k=1}^{u(l+1)} \delta_k^{l+1} g'(z_k^{l+1}) w_{kj}^{l+1}$$

with $\frac{\partial a_{k}^{l+1}}{\partial z_{k}^{l+1}} = g'(z_{k}^{l+1})$, $\frac{\partial z_{k}^{l+1}}{\partial a_{j}^{l}} = w_{kj}^{l+1}$ and $z_j^l = (w_j^l)^\top a^{l-1}$
Let’s put the terms together

Term 3: \[ \frac{\partial z_j^l}{\partial w_{ji}^l} \]

Term 2: \[ \frac{\partial a_j^l}{\partial z_j^l} \]

Term 1: \[ \frac{\partial J(\Theta)}{\partial a_j^l} \]
Let’s put the terms together

Let’s denote \( d_{ji}^l = \frac{\partial J(\Theta)}{\partial w_{ji}^l} \)

\[
\frac{\partial J(\Theta)}{\partial w_{ji}^l} = d_{ji}^l = \frac{\partial J(\Theta)}{\partial a_j^l} \frac{\partial a_j^l}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{ji}^l} = \delta_j^l g'(z_j^l) a_{i}^{l-1}
\]

with

- \( l \neq L \): \( \delta_j^l = \sum_{k=1}^{u(l+1)} \delta_{k}^{l+1} g'(z_k^{l+1}) w_{kj}^{l+1} \)
- \( l = L \): depends on the cost \( J(\Theta) \)
Back-propagation

Cost functions $J(\Theta)$

- **Mean Squared Error:**
  $$J(\Theta) = \frac{1}{2} \sum_{n=1}^{N} (h_\Theta(x_n) - y_n)^2$$
  $$\delta^L_j = \delta^L_1 = h_\Theta(x_n) - y_n$$

- **Log-likelihood:**
  $$J(\Theta) = -\frac{1}{N} \sum_{n=1}^{N} [y_n \log h_\Theta(x_n) + (1 - y_n) \log(1 - h_\Theta(x_n))]$$
  $$\delta^L_j = \delta^L_1 = \frac{1}{N} \frac{h_\Theta(x_n) - y_n}{h_\Theta(x_n)(1 - h_\Theta(x_n))}$$
Outline

1 Recap

2 Back-Propagation
   • Multi-Layer Perceptron: notations
   • Multi-Layer Perceptron: back-propagation
   • Multi-Layer Perceptron: back-propagation in action
   • Multi-Layer Perceptron: training
   • Multi-Layer Perceptron: tricks

3 Wrapping up

4 Next
Back-propagation in action

**Initial weights (random)**

- Input nodes:
  - $x_1$
  - $x_2$

- Hidden layer 1:
  - $a^1_1$
  - $a^1_2$

- Output layer:
  - $a^2_1$

Weights:
- $w^1_{10} = 1.1$
- $w^1_{11} = 0.7$
- $w^1_{12} = 1.3$
- $w^1_{21} = -0.3$
- $w^1_{22} = 0.6$
- $w^2_{10} = 0.3$
- $w^2_{11} = -0.6$
- $w^2_{12} = 1.2$
- $w^2_{20} = 2.3$
- $w^2_{21} = -0.3$
- $w^2_{22} = 0.6$

Activation function:
- $a^j_j = g(z^j_j)$
- $g(z) = \tanh(z)$

Output:
- $a^L = a^2_1 = z^2_1$

(linear output)
Back-propagation in action

Forward pass 1 (propagation): \( x_i = (0.8, -0.8) \) \( y_i = -0.5 \)

\[
\begin{align*}
    z_1^1 &= 1.1 + 0.7 \times 0.8 + 1.3 \times (-0.8) \\
    a_1^1 &= g(z_1^1) = \tanh(0.62) = 0.55 \\
    z_2^1 &= 2.3 - 0.3 \times 0.8 + 0.6 \times (-0.8) \\
    a_2^1 &= g(z_2^1) = \tanh(1.58) = 1.00 \\
    z_2^2 &= 0.3 - 0.6 \times 0.55 + 1.2 \times 1.00 \\
    a_1^2 &= z_1^2 = 1.175 \\
    J_i(\Theta) &= (1.175 + 0.5)^2 = 2.805
\end{align*}
\]
Back-propagation in action

Backward pass: computing $\delta_j^l = \frac{\partial J(\Theta)}{\partial a_j^l}$ (back-propagation)

**Output units:**

$$\delta_j^L = \delta_1^2 = 1.175 + 0.5 = 1.675$$

**Hidden units:**

$$\delta_j^l = \sum_{k=1}^{u(l+1)} \delta_k^{l+1} g'(z_k^{l+1}) w_{kj}^{l+1} \ (l \neq L)$$

$$\delta_0^1 = \delta_1^2 g'(z_1^2) w_{10}^2 = 1.675 \times 0.3 = 5.025$$

$$\delta_1^1 = \delta_1^2 g'(z_1^2) w_{11}^2 = 1.675 \times -0.6 = -1.005$$

$$\delta_2^1 = \delta_1^2 g'(z_1^2) w_{12}^2 = 1.675 \times 1.2 = 2.01$$

$$g'(z_j^2) = g'(z_j^L) = 1 \ (\text{linear output } L = 2)$$
Back-propagation in action

Backward pass: computing $d_{kj}^l = \frac{\partial J(\Theta)}{\partial w_{kj}^l}$ (back-propagation)

Output layer:

\[
\begin{align*}
    d_{k0}^l &= \delta_k^l g'(z_k^l) a_j^{l-1} \\
    d_{10}^2 &= \delta_1^2 g'(z_1^2) a_0^1 \\
    &= 1.675 \times 1 \times 1 = 1.675 \\
    d_{11}^2 &= \delta_1^2 g'(z_1^2) a_1^1 \\
    &= 1.675 \times 1 \times 0.55 = 0.92125 \\
    d_{12}^2 &= \delta_1^2 g'(z_1^2) a_2^1 \\
    &= 1.675 \times 1 \times 1.0065 = 1.6858 \\
\end{align*}
\]

$g'(z_j^l) = 1$ ($l = L$)
Back-propagation in action

Backward pass: computing $d_{kj}^l = \frac{\partial J(\Theta)}{\partial w_{kj}^l}$ (back-propagation)

Hidden layer:

$d_{10}^l = \delta_k^l g'(z_k^l) a_j^{l-1}$
$d_{10}^1 = \delta_1^1 g'(z_1^1) x_0$
$d_{11}^1 = \delta_1^1 g'(z_1^1) x_1$
$d_{12}^1 = \delta_1^1 g'(z_1^1) x_2$

$g'(z_j^l) = 1 - \tanh(z_j^l)^2$ ($l \neq L$)
Back-propagation in action

Backward pass: computing $d^l_{kj} = \frac{\partial J(\Theta)}{\partial w'_{kj}}$ (back-propagation)

Hidden layer:

- $d^l_{kj} = \delta^l_k g'(z^l_k) a^l_{j-1}$
- $d^1_{20} = \delta^1_2 g'(z^1_2) x_0$
  $= 2.1 \times (1 - g(1.58)) \times 1 = -0.0137$
- $d^1_{21} = \delta^1_2 g'(z^1_2) x_1$
  $= -0.0137 \times 0.8 = -0.01096$
- $d^1_{22} = \delta^1_2 g'(z^1_2) x_2$
  $= -0.0137 \times -0.8 = 0.01096$

$g'(z^l_j) = 1 - \tanh(z^l_j)^2$ ($l \neq L$)
Back-propagation in action

Weight update: \((w^l_{kj})^{\text{new}} = w^l_{kj} - \alpha d^l_{kj} \ (\alpha = 0.1)\)

Output layer:

\[
\begin{align*}
(w^2_{10})^{\text{new}} &= w^2_{10} - 0.1 \times d^2_{10} \\
&= 0.3 - 0.1675 = 0.1325 \\
(w^2_{11})^{\text{new}} &= w^2_{11} - 0.1 \times d^2_{11} \\
&= -0.6 - 0.092125 = -0.692125 \\
(w^2_{12})^{\text{new}} &= w^2_{12} - 0.1 \times d^2_{12} \\
&= 1.2 - 0.16858 = 1.03142
\end{align*}
\]
Back-propagation in action

Weight update: \((w_{kj}^l)^{new} = w_{kj}^l - \alpha d_{kj}^l\) \((\alpha = 0.1)\)

Hidden layer:

\[
\begin{align*}
(w_{10}^1)^{new} &= w_{10}^1 - 0.1 \times d_{10}^1 \\
&= 1.1 + 0.0447 = 1.1447 \\
(w_{11}^1)^{new} &= w_{11}^1 - 0.1 \times d_{11}^1 \\
&= 0.7 + 0.03576 = 0.73576 \\
(w_{12}^1)^{new} &= w_{12}^1 - 0.1 \times d_{12}^1 \\
&= 1.3 - 0.03576 = 1.26424
\end{align*}
\]
Back-propagation in action

Weight update: \( (w'_{kj})^{\text{new}} = w'_{kj} - \alpha d'_{kj} (\alpha = 0.1) \)

Hidden layer:

\[
\begin{align*}
(w'_{20})^{\text{new}} &= w^1_{20} - 0.1 \times d^1_{20} \\
&= 2.3 + 0.00137 = 2.30137 \\
(w'_{21})^{\text{new}} &= w^1_{21} - 0.1 \times d^1_{21} \\
&= -0.3 + 0.001096 = -0.298904 \\
(w'_{22})^{\text{new}} &= w^1_{22} - 0.1 \times d^1_{22} \\
&= 0.6 - 0.001096 = 0.598904
\end{align*}
\]
Back-propagation in action

Forward pass 2 (propagation)

\[
\begin{align*}
  z_1^1 &= 1.1447 + 0.73576 \times 0.8 + 1.26424 \times -0.8 \\
  a_1^1 &= g(z_1^1) = \tanh(0.721916) = 0.6252 \\
  z_2^1 &= 2.30137 - 0.2989 \times 0.8 + 0.5989 \times -0.8 \\
  a_2^1 &= g(z_2^1) = \tanh(1.58313) = 1.0074 \\
  z_1^2 &= 0.1325 - 0.692 \times 0.6252 + 1.0314 \times 1.0074 \\
  a_1^2 &= z_1^2 = 0.738 \\
  J_i(\Theta) &= (0.738 + 0.5)^2 = 1.532
\end{align*}
\]
Outline

1 Recap

2 Back-Propagation
   - Multi-Layer Perceptron: notations
   - Multi-Layer Perceptron: back-propagation
   - Multi-Layer Perceptron: back-propagation in action
   - Multi-Layer Perceptron: training
   - Multi-Layer Perceptron: tricks

3 Wrapping up

4 Next
Training

Gradient Descent “Batch”

1. Initialize the weights randomly $w_{ji} \neq 0$
2. For each iteration until convergence
   1. For each example $(x_n, y_n)$
      1. Forward pass: compute $h_\Theta(x_n)$
      2. Backward pass: compute derivatives $\delta_j^l$ and $d_{kj}^l$
      3. Accumulate derivatives: $D_{kj}^l + = d_{kj}^l$
   2. Update parameters $\Theta$:
      - $(w_{kj}^l)^{new} = w_{kj}^l - \alpha \frac{1}{N} D_{kj}^l + \lambda w_{kj}^l$ (with regularization)
      - $(w_{k0}^l)^{new} = w_{k0}^l - \alpha \frac{1}{N} D_{k0}^l$ (bias term)
Gradient Descent “Stochastic”

1. Initialize the weights randomly $w_{ji} \neq 0$
2. For each iteration until convergence
   - pick an example $(x_n, y_n)$ at random
   - Forward pass: compute $h_{\Theta}(x_n)$
   - Backward pass: compute derivatives $\delta_j^l$ and $d_{kj}^l$
   - Update parameters $\Theta$:
     - $(w_{kj}^l)^{new} = w_{kj}^l - \alpha \frac{1}{N} d_{kj}^l + \lambda w_{kj}^l$ (with regularization)
     - $(w_{k0}^l)^{new} = w_{k0}^l - \alpha \frac{1}{N} d_{k0}^l$ (bias term)
Training

“Batch” vs “Stochastic”

- “Stochastic” faster than “Batch”
- “Stochastic” often provides better solutions
- Acceleration techniques such as conjugate gradient only for “batch”
- Conditions of convergence well understood for “batch”
Outline

1 Recap

2 Back-Propagation
   - Multi-Layer Perceptron: notations
   - Multi-Layer Perceptron: back-propagation
   - Multi-Layer Perceptron: back-propagation in action
   - Multi-Layer Perceptron: training
   - Multi-Layer Perceptron: tricks

3 Wrapping up

4 Next
Tricks

- Shuffle the examples ("Stochastic" only)
- Decorrelate the input data (perform PCA)
- Normalize the input data to zero mean and unit standard deviation
- Replace the sigmoid by tanh and in particular $1.7159 \tanh\left(\frac{2}{3}z\right)$
- Replace the target values
  - $\{0.1, 0.9\}$ instead of $\{0.0, +1.0\}$ for sigmoid,
  - $\{-0.8, 0.8\}$ instead of $\{-1.0, +1.0\}$ for tanh,
- Initialize the weights $w_{ij}^{l} \sim \mathcal{N}(0, \frac{1}{\sqrt{v_j^l}})$ ($v_j^l$ is the fan-in)
- Others: momentum, adaptive learning rates, ...
Outline

1. Recap
2. Back-Propagation
3. Wrapping up
4. Next
Multi-Layer Perceptron Training

Gradient Descent “Stochastic”

1. Initialize the weights randomly $w^l_{ji} \neq 0$
2. For each iteration until convergence
   1. Pick an example $(x_n, y_n)$ at random
   2. Forward pass: compute $h_\Theta(x_n)$
   3. Backward pass: compute derivatives $\delta_j^l$ and $d_{kj}^l$
   4. Update parameters $\Theta$:
      - $(w_{kj}^l)^{new} = w_{kj}^l - \alpha \frac{1}{N} d_{kj}^l + \lambda w_{kj}^l$ (with regularization)
      - $(w_{k0}^l)^{new} = w_{k0}^l - \alpha \frac{1}{N} d_{k0}^l$ (bias term)
Recap

Back-Propagation

Wrapping up

Next

Multi-Layer Perceptron Training

Forward pass (propagation): \( x_i = (0.8, -0.8) \), \( y_i = -0.5 \)

\[
\begin{align*}
\mathbf{w}_1^{10} &= 1.1 \\
\mathbf{w}_1^{11} &= 0.7 \\
\mathbf{w}_1^{12} &= 1.3 \\
\mathbf{w}_2^{10} &= 0.3 \\
\mathbf{w}_2^{11} &= -0.6 \\
\mathbf{w}_2^{12} &= -0.3 \\
\mathbf{w}_2^{20} &= 2.3
\end{align*}
\]

\[
\begin{align*}
\mathbf{z}_1 &= 1.1 + 0.7 \times 0.8 + 1.3 \times -0.8 \\
\mathbf{a}_1 &= g(z_1) = \tanh(0.62) = 0.55 \\
\mathbf{z}_2 &= 2.3 - 0.3 \times 0.8 + 0.6 \times -0.8 \\
\mathbf{a}_2 &= g(z_2) = \tanh(1.58) = 1.00 \\
\mathbf{z}_1^2 &= 0.3 - 0.6 \times 0.55 + 1.2 \times 1.00 \\
\mathbf{a}_1^2 &= \mathbf{z}_1^2 = 1.175
\end{align*}
\]

MSE cost:

\[
J_i(\Theta) = (1.175 + 0.5)^2 = 2.805
\]
Multi-Layer Perceptron Training

Backward pass: computing $d_{kj}^l = \frac{\partial J(\Theta)}{\partial w_{kj}^l}$ (back-propagation)

\[\delta_j^l = \sum_{k=1}^{u_l+1} \delta_{k}^{l+1} g'(z_k^{l+1}) w_{kj}^{l+1}\]

\[d_{kj}^l = \delta_k^l g'(z_k^l) a_j^{l-1}\]

\[(w_{kj}^l)_{\text{new}} = w_{kj}^l - \alpha d_{kj}^l\]
Outline

1. Recap
2. Back-Propagation
3. Wrapping up
4. Next
Next lecture

Classification and Regression (4/4)
- k-Nearest Neighbors (k-NN)
- Linear Regression (univariate)
- Linear Regression (multivariate) and the Gradient descent
- Logistic Regression (a classification algorithm !)
- Multi-Layer Perceptron

Dimensionality reduction and clustering
★ Principal Component Analysis (PCA)
★ Linear Discriminant Analysis (LDA)
★ k-Means
Thank you for your attention

Sébastien Marcel
Idiap Research Institute
Martigny, Switzerland
marcel AT idiap DOT ch
www.idiap.ch/∼marcel