Advances in Neural Machine Translation

Rico Sennrich, Alexandra Birch, Marcin Junczys-Dowmunt

Institute for Language, Cognition and Computation
University of Edinburgh

November 1 2016

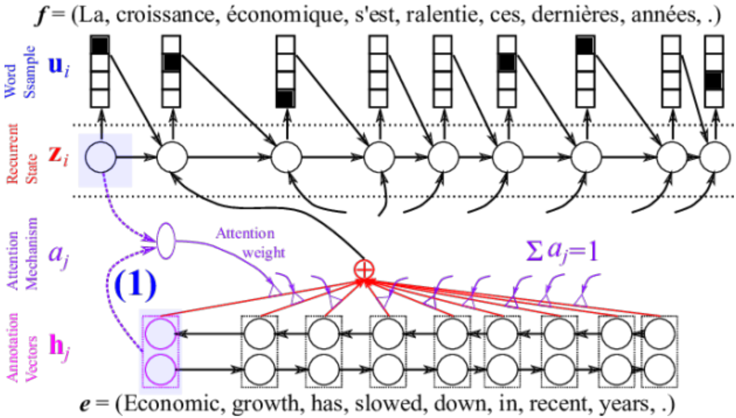
Machine Translation

Why we need MT

- human translation industry: $\approx$ 666 million words / day [Pym et al., 2012]
- MT indudstry: $\gg$ 100 billion words / day [Turovsky, 2016]

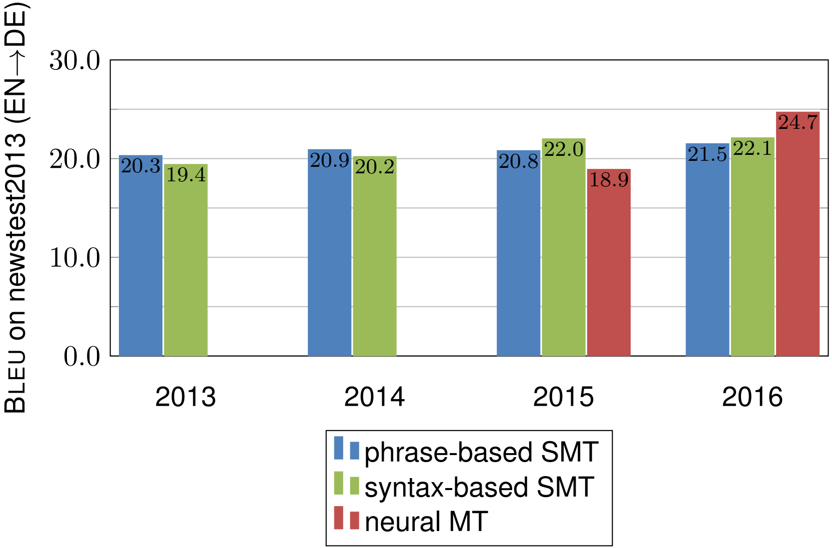
demand for translation for outpaces what is humanly possible to produce
$\rightarrow$ we need fast, high-quality MT

Neural Machine Translation



Kyunghyun Cho
http://devblogs.nvidia.com/parallelforall/introduction-neural-machine-translation-gpus-part-3/

Edinburgh's* WMT results over the years



*NMT 2015 from U. Montréal: `https://sites.google.com/site/acl16nmt/`

neural MT has already moved from academia into production

SYSTRAN announces the launch of its "Purely Neural MT" engine, a revolution for the machine translation market

Google announces Neural Machine Translation to improve Google Translate

WIPO goes Neural

Oct 4, 2016 | 590 views | 41 Likes | 3 Comments

Why neural MT?

- single, end-to-end trained neural network replaces collection of weak features
- good generalization via continuous space representations
  $\rightarrow$ modelling of dependencies over long distances

why now?

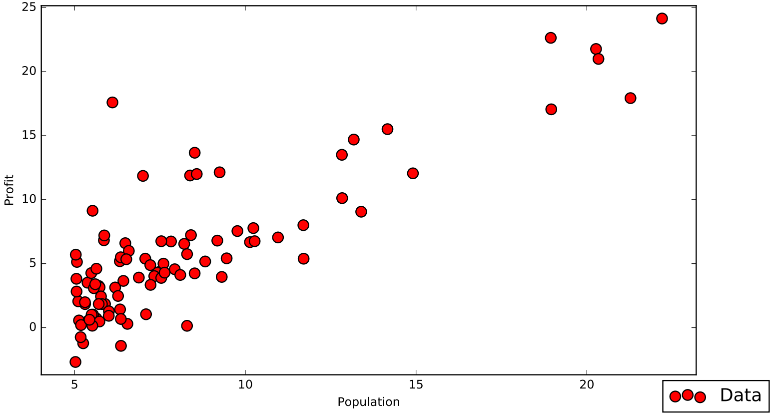
- neural translation dates back to at least the 80s [Allen, 1987]
- large-scale neural MT is now possible thanks to
  - large amounts of training data
  - exponential growth in computational power (GPUs!)
  - algorithmic advances

Linear Regression

Parameters: $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$ Model: $h_\theta(x) = \theta_0 + \theta_1 x$

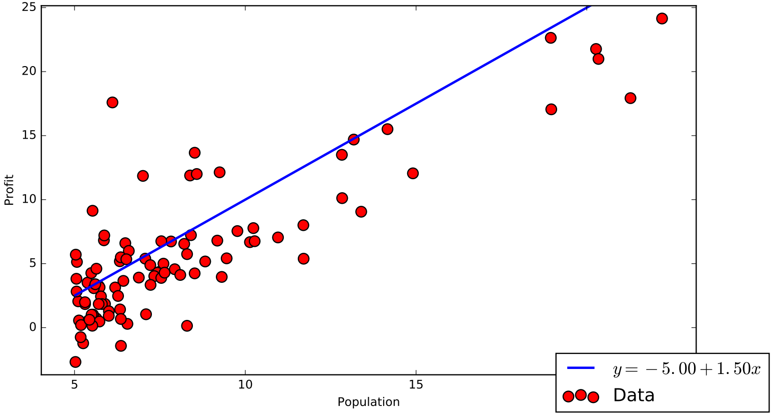
Linear Regression

Parameters: $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$ Model: $h_\theta(x) = \theta_0 + \theta_1 x$



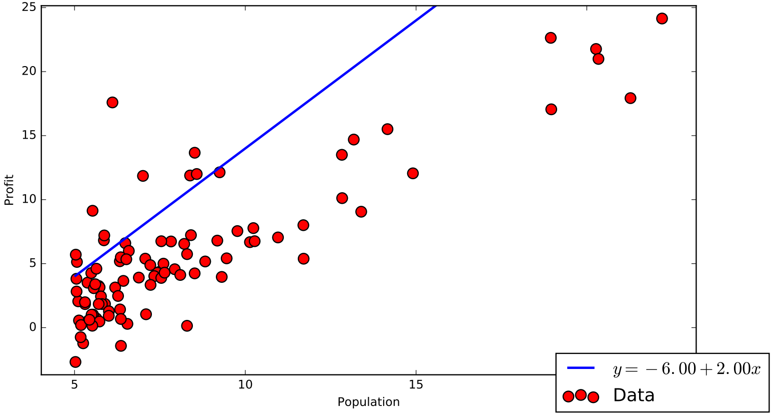
Linear Regression

Parameters: $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$ Model: $h_\theta(x) = \theta_0 + \theta_1 x$



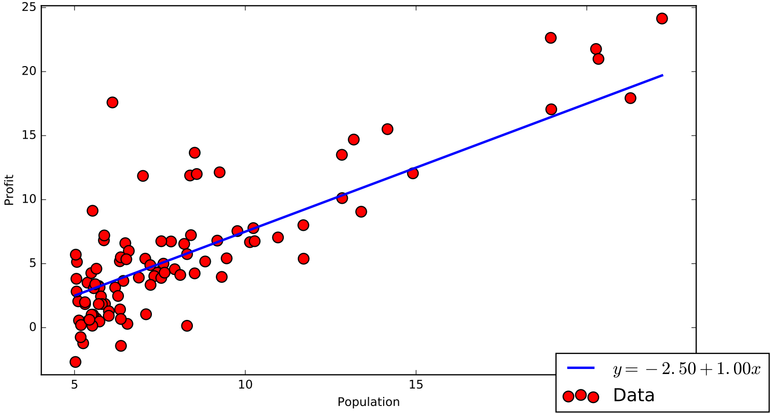
Linear Regression

Parameters: $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$ Model: $h_\theta(x) = \theta_0 + \theta_1 x$



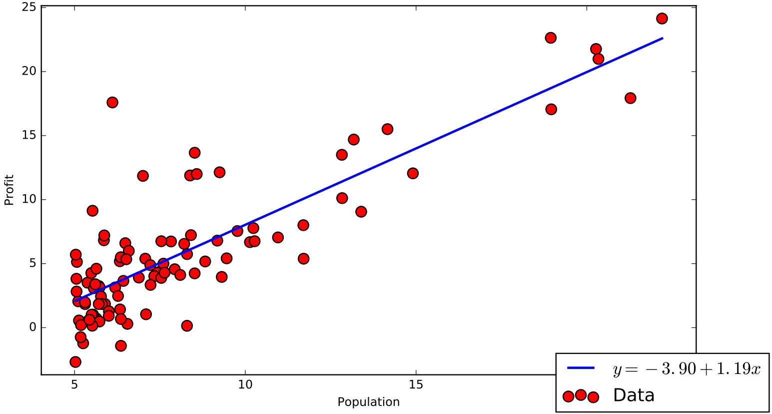
Linear Regression

Parameters: $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$ Model: $h_\theta(x) = \theta_0 + \theta_1 x$



Linear Regression

Parameters: $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$ Model: $h_\theta(x) = \theta_0 + \theta_1 x$



The cost (or loss) function

- We try to find parameters $\hat{\theta} \in \mathbb{R}^2$ such that the cost function $J(\theta)$ is minimal:

$$J : \mathbb{R}^2 \rightarrow \mathbb{R}$$

$$\hat{\theta} = \underset{\theta \in \mathbb{R}^2}{\arg\min}\; J(\theta)$$

The cost (or loss) function

- We try to find parameters $\hat{\theta} \in \mathbb{R}^2$ such that the cost function $J(\theta)$ is minimal:

$$J : \mathbb{R}^2 \to \mathbb{R}$$

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^2} J(\theta)$$

- Mean Square Error:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

The cost (or loss) function

- We try to find parameters $\hat{\theta} \in \mathbb{R}^2$ such that the cost function $J(\theta)$ is minimal:

$$J : \mathbb{R}^2 \to \mathbb{R}$$

$$\hat{\theta} = \underset{\theta \in \mathbb{R}^2}{\arg\min}\ J(\theta)$$

- Mean Square Error:

$$\begin{aligned} J(\theta) &= \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 \\ &= \frac{1}{2m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right)^2 \end{aligned}$$

The cost (or loss) function

- We try to find parameters $\hat{\theta} \in \mathbb{R}^2$ such that the cost function $J(\theta)$ is minimal:

$$J : \mathbb{R}^2 \to \mathbb{R}$$

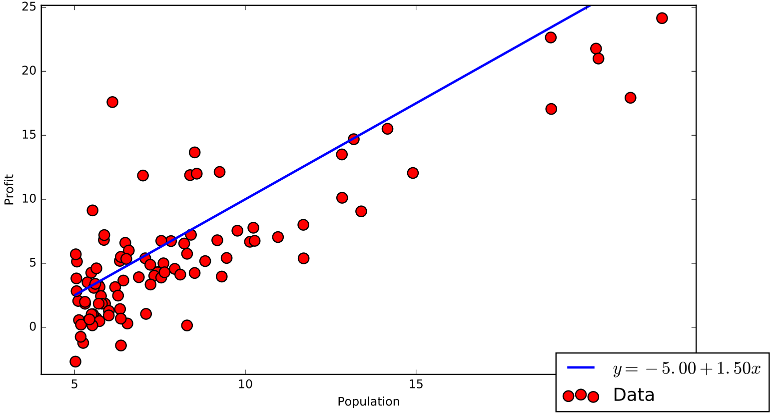
$$\hat{\theta} = \underset{\theta \in \mathbb{R}^2}{\arg\min}\ J(\theta)$$

- Mean Square Error:

$$\begin{aligned} J(\theta) &= \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 \\ &= \frac{1}{2m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right)^2 \end{aligned}$$

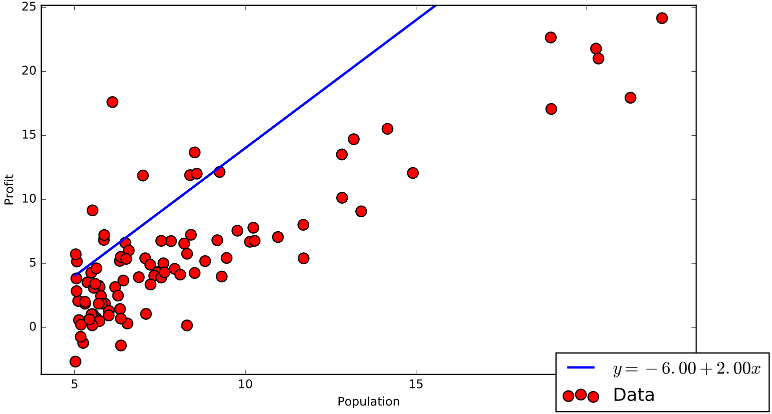
where m is the number of data points in the training set.

The cost (or loss) function



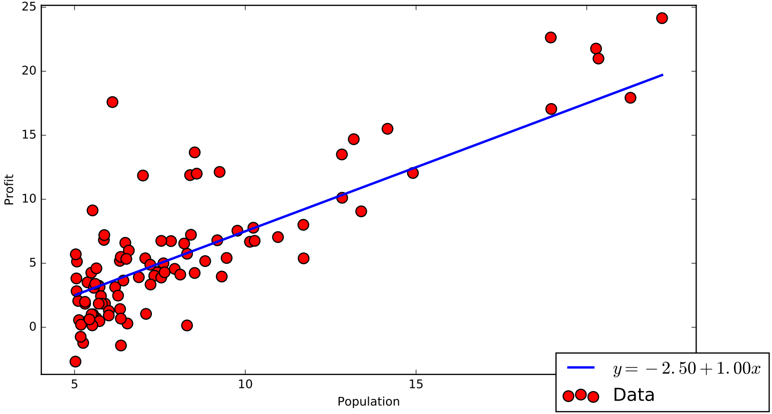
$$J\left(\begin{bmatrix} -5.00 \\ 1.50 \end{bmatrix}\right) = 6.1561$$

The cost (or loss) function



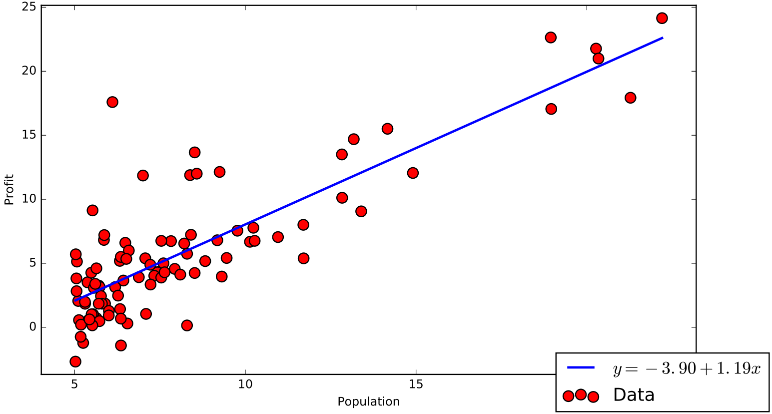
$$J\left(\begin{bmatrix} -6.00 \\ 2.00 \end{bmatrix}\right) = 19.3401$$

The cost (or loss) function



$$J\left(\begin{bmatrix} -2.50 \\ 1.00 \end{bmatrix}\right) = 4.7692$$

The cost (or loss) function



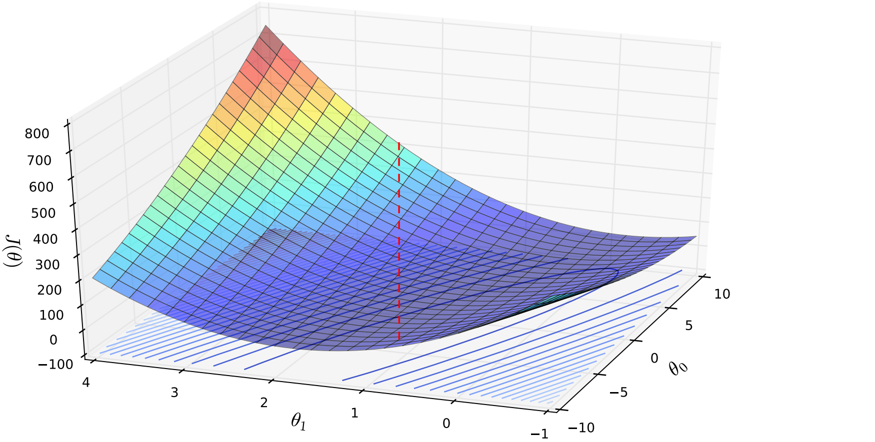
$$J\left(\begin{bmatrix} -3.90 \\ 1.19 \end{bmatrix}\right) = 4.4775$$

The cost (or loss) function

So, how do we find $\hat{\theta} = \underset{\theta \in \mathbb{R}^2}{\arg\min}\ J(\theta)$ computationally?

The cost (or loss) function

So, how do we find $\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^2} J(\theta)$ computationally?



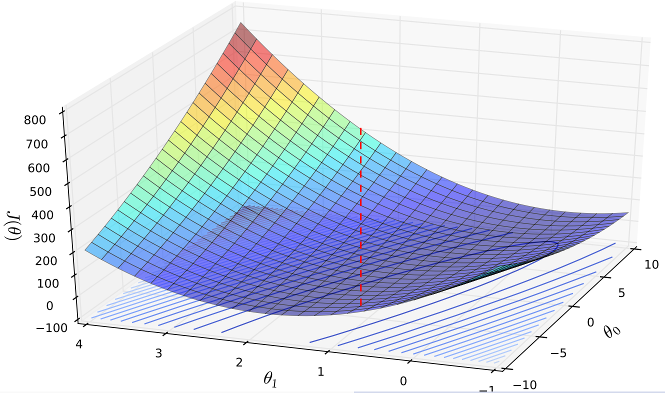
(Stochastic) gradient descent

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \text{ for each } j$$

(Stochastic) gradient descent

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \text{ for each } j$$

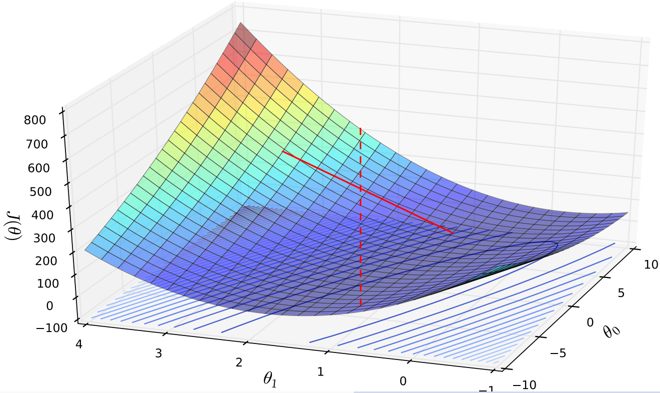
Step 0, $\alpha = 0.01$



(Stochastic) gradient descent

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \text{ for each } j$$

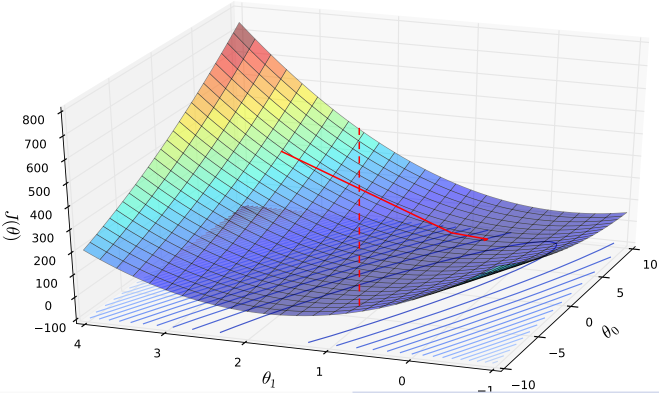
Step 1, $\alpha = 0.01$



(Stochastic) gradient descent

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \text{ for each } j$$

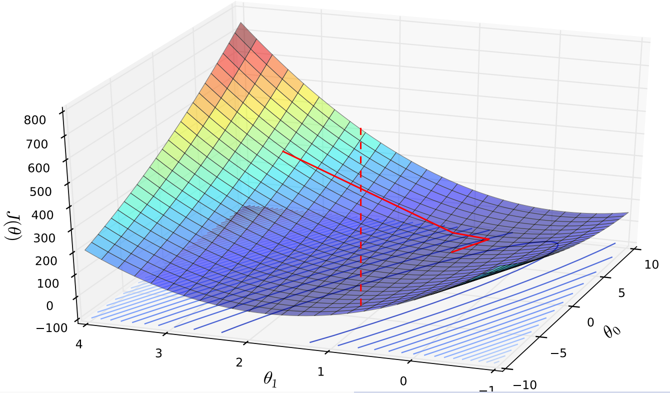
Step 20, $\alpha = 0.01$



(Stochastic) gradient descent

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \text{ for each } j$$

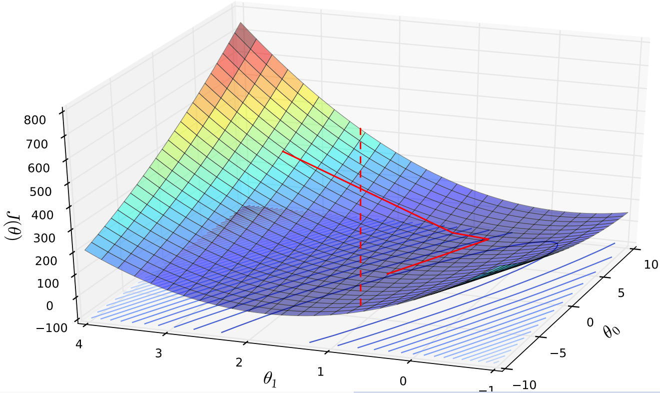
Step 200, $\alpha = 0.01$



(Stochastic) gradient descent

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \text{ for each } j$$

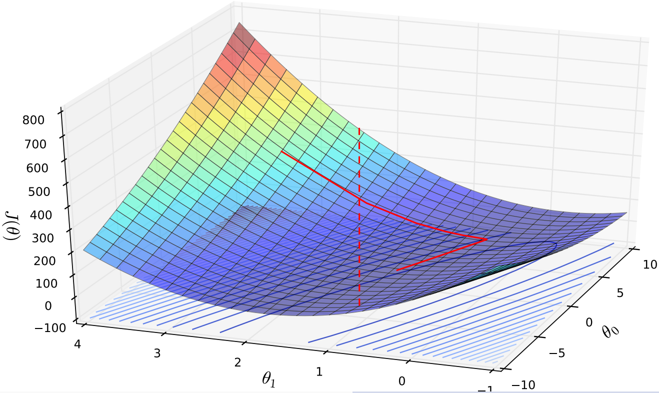
Step 10000, $\alpha = 0.01$



(Stochastic) gradient descent

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \text{ for each } j$$

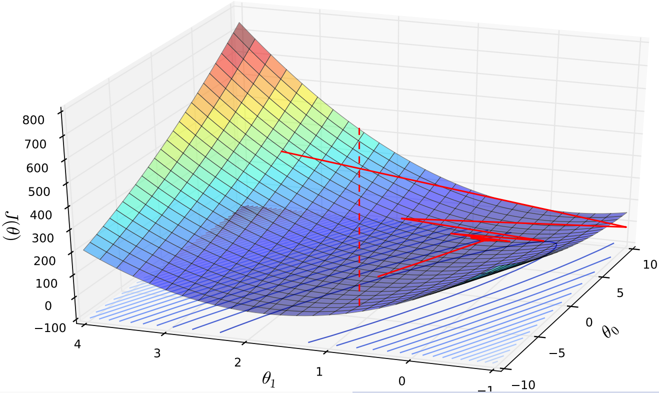
Step 10000, $\alpha = 0.005$



(Stochastic) gradient descent

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \text{ for each } j$$

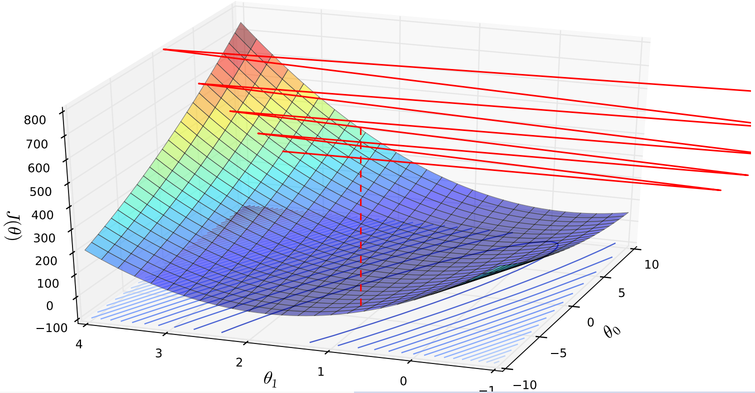
Step 10000, $\alpha = 0.02$



(Stochastic) gradient descent

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \text{ for each } j$$

Step 10, $\alpha = 0.025$



(Stochastic) gradient descent

How do we calculate $\frac{\partial}{\partial \theta_j} J(\theta)$?

(Stochastic) gradient descent

How do we calculate $\frac{\partial}{\partial \theta_j} J(\theta)$?

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \\
&= \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}
\end{aligned}$$

(Stochastic) gradient descent

How do we calculate $\frac{\partial}{\partial \theta_j} J(\theta)$?

$$
\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \\
&= 2 \cdot \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot \frac{\partial}{\partial \theta_j} (h_\theta(x^{(i)}) - y^{(i)}) \\
&= \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot \frac{\partial}{\partial \theta_j} \sum_{i=0}^{n} \theta_i x_i^{(i)} \\
&= \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}
\end{aligned}
$$

The update rule once again

For linear regression we have the following model:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

The update rule once again

For linear regression we have the following model:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

and we repeat until convergence (θ_0 and θ_1 should be updated simultaneously):

$$\begin{aligned}
\theta_0 &:= \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \\
\theta_1 &:= \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x^{(i)}
\end{aligned}$$

To summarize what we have learned

When approaching a machine learning problem, we need:

To summarize what we have learned

When approaching a machine learning problem, we need:

- a suitable model;

To summarize what we have learned

When approaching a machine learning problem, we need:

- a suitable model;
- a suitable cost (or loss) function;

To summarize what we have learned

When approaching a machine learning problem, we need:

- a suitable model;
- a suitable cost (or loss) function;
- an optimization algorithm;

To summarize what we have learned

When approaching a machine learning problem, we need:

- a suitable model;
- a suitable cost (or loss) function;
- an optimization algorithm;
- the gradient(s) of the cost function (if required by the optimization algorithm).

To summarize what we have learned

When approaching a machine learning problem, we need:

- a suitable model; (here: a linear model)
- a suitable cost (or loss) function; (here: mean square error)
- an optimization algorithm; (here: a variant of SGD)
- the gradient(s) of the cost function (if required by the optimization algorithm).

To summarize what we have learned

When approaching a machine learning problem, we need:

- a suitable model; (here: a linear model)
- a suitable cost (or loss) function; (here: mean square error)
- an optimization algorithm; (here: a variant of SGD)
- the gradient(s) of the cost function (if required by the optimization algorithm).

Side note: algorithms for finding the minimum without the gradient

To summarize what we have learned

When approaching a machine learning problem, we need:

- a suitable model; (here: a linear model)
- a suitable cost (or loss) function; (here: mean square error)
- an optimization algorithm; (here: a variant of SGD)
- the gradient(s) of the cost function (if required by the optimization algorithm).

Side note: algorithms for finding the minimum without the gradient

- for linear regession: the normal matrix (exact);

To summarize what we have learned

When approaching a machine learning problem, we need:

- a suitable model; (here: a linear model)
- a suitable cost (or loss) function; (here: mean square error)
- an optimization algorithm; (here: a variant of SGD)
- the gradient(s) of the cost function (if required by the optimization algorithm).

Side note: algorithms for finding the minimum without the gradient

- for linear regession: the normal matrix (exact);
- random search;

To summarize what we have learned

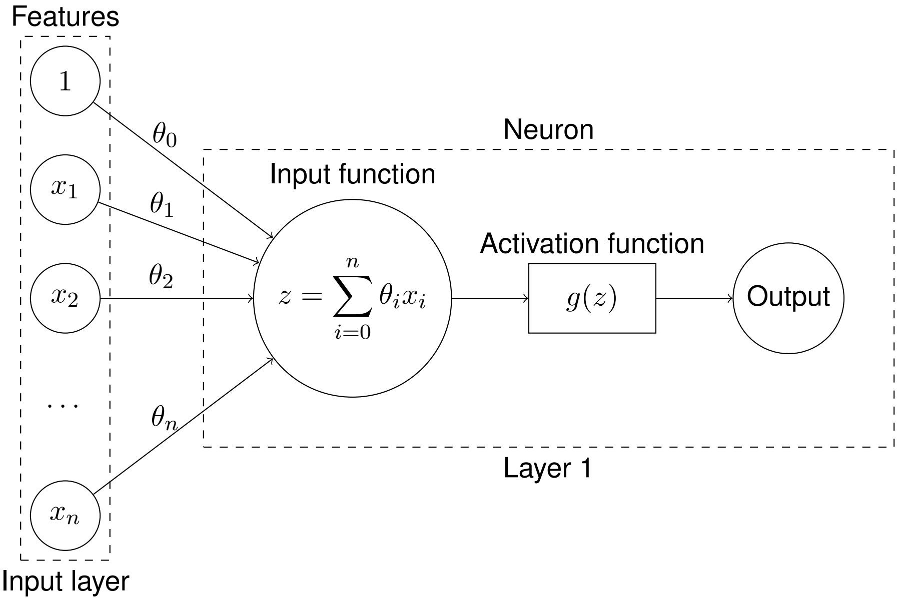
When approaching a machine learning problem, we need:

- a suitable model; (here: a linear model)
- a suitable cost (or loss) function; (here: mean square error)
- an optimization algorithm; (here: a variant of SGD)
- the gradient(s) of the cost function (if required by the optimization algorithm).

Side note: algorithms for finding the minimum without the gradient

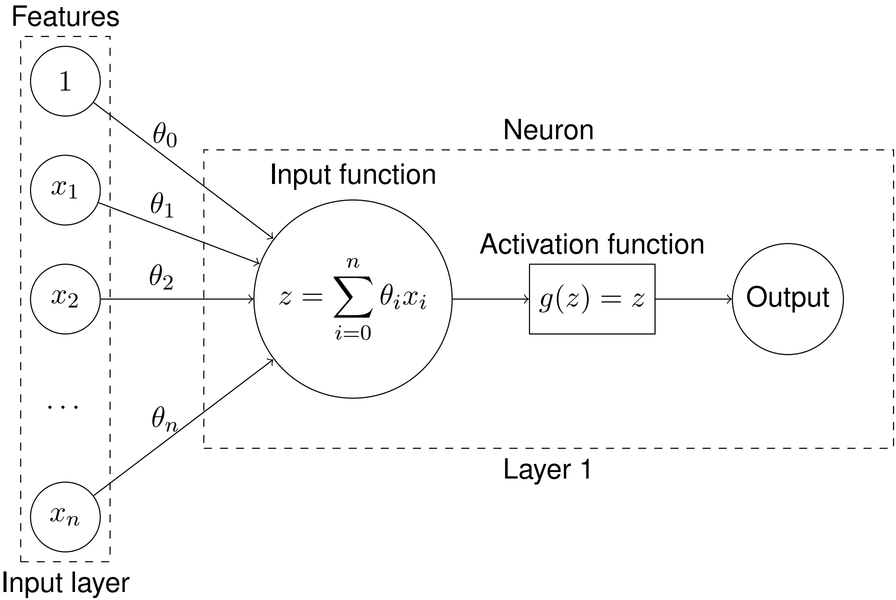
- for linear regession: the normal matrix (exact);
- random search;
- genetic algorithms;
- ...

A neuron

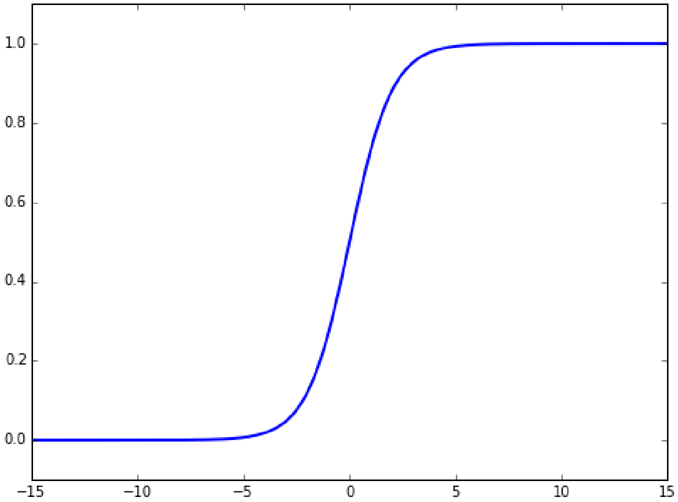


Features
1
x_1
x_2
...
x_n
Input layer
θ_0
θ_1
θ_2
θ_n
Neuron
Input function
$z = \sum_{i=0}^{n} \theta_i x_i$
Activation function
$g(z)$
Output
Layer 1

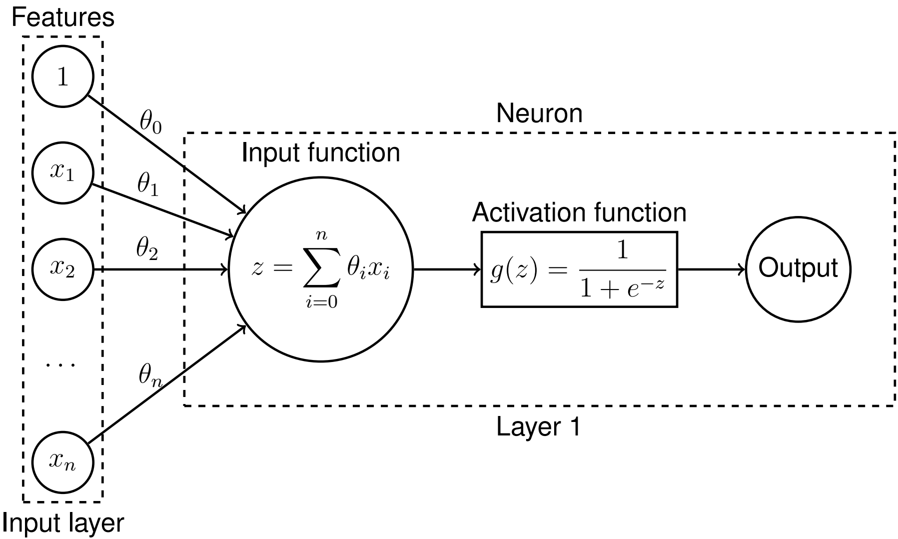
Linear regression and neural networks



The logistic function (remember this one!)



A more typical neuron (binary logistic regression)



Binary logistic regression

- Model:

$$h_\theta(x) = g(\sum_{i=0}^{n} \theta_i x_i) = \frac{1}{1 + e^{-\sum_{i=0}^{n} \theta_i x_i}}$$

Binary logistic regression

- Model:

$$h_\theta(x) = g(\textstyle\sum_{i=0}^{n} \theta_i x_i) = \frac{1}{1 + e^{-\sum_{i=0}^{n} \theta_i x_i}}$$

- Cost function (binary crossentropy):

$$J(\theta) = -\frac{1}{m}[\textstyle\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))]$$

Binary logistic regression

- Model:

$$h_\theta(x) = g(\textstyle\sum_{i=0}^{n} \theta_i x_i) = \dfrac{1}{1 + e^{-\sum_{i=0}^{n} \theta_i x_i}}$$

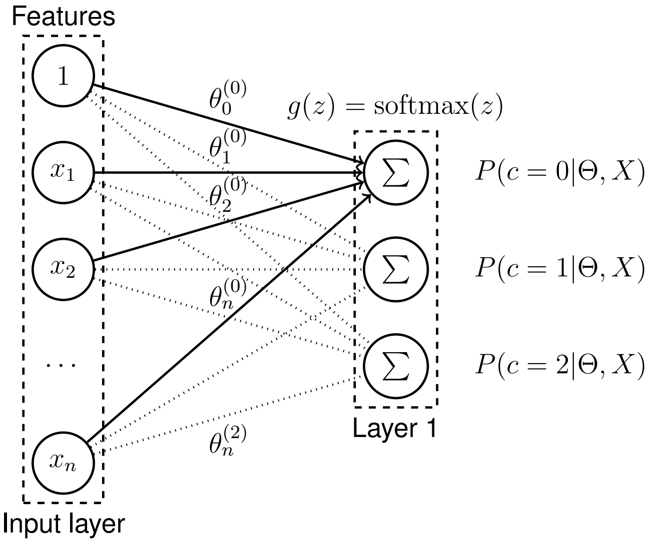
- Cost function (binary crossentropy):

$$J(\theta) = -\frac{1}{m}[\textstyle\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))]$$

- Gradient:

$$\tfrac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Multi-class logistic regression and neural networks



Multi-class logistic regression

- Model: $h_\Theta(x) = [P(k|x,\Theta)]_{k=1,\dots,c} = \mathrm{softmax}(\Theta x)$ where $\Theta = (\theta^{(1)}, \dots, \theta^{(c)})$

Multi-class logistic regression

- Model: $h_\Theta(x) = [P(k|x, \Theta)]_{k=1,\dots,c} = \mathrm{softmax}(\Theta x)$ where $\Theta = (\theta^{(1)}, \dots, \theta^{(c)})$
- Cost function: $J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{c}\delta(y^{(i)}, k)\log P(k|x^{(i)}, \Theta)$ where $\delta(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$

Multi-class logistic regression

- Model: $h_\Theta(x) = [P(k|x,\Theta)]_{k=1,\dots,c} = \mathrm{softmax}(\Theta x)$ where $\Theta = (\theta^{(1)}, \dots, \theta^{(c)})$
- Cost function: $J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{c}\delta(y^{(i)},k)\log P(k|x^{(i)},\Theta)$
  where $\delta(x,y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$
- Gradient: $\dfrac{\partial J(\Theta)}{\partial \Theta_{j,k}} = -\frac{1}{m}\sum_{i=1}^{m}(\delta(y^{(i)},k) - P(k|x^{(i)},\Theta))\, x_j^{(i)}$

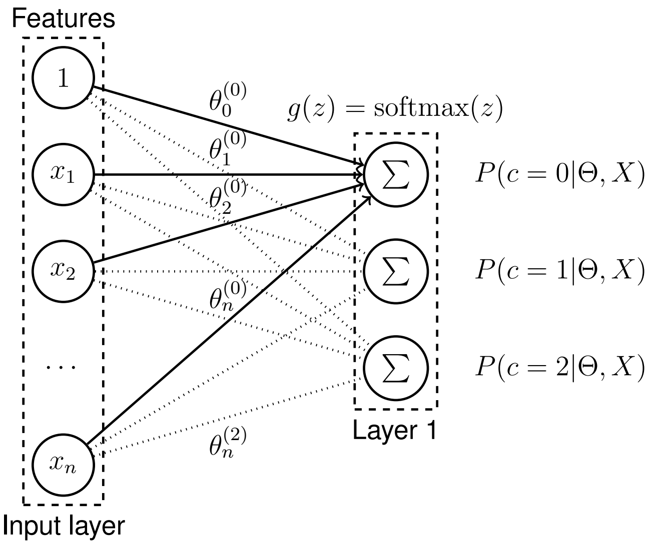
Multi-class logistic regression

- Model: $h_\Theta(x) = [P(k|x,\Theta)]_{k=1,\ldots,c} = \mathrm{softmax}(\Theta x)$ where $\Theta = (\theta^{(1)}, \ldots, \theta^{(c)})$
- Cost function: $J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{c}\delta(y^{(i)},k)\log P(k|x^{(i)},\Theta)$
  where $\delta(x,y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$
- Gradient: $\dfrac{\partial J(\Theta)}{\partial \Theta_{j,k}} = -\frac{1}{m}\sum_{i=1}^{m}(\delta(y^{(i)},k) - P(k|x^{(i)},\Theta))\, x_j^{(i)}$
- May look complicated, but can be looked up!

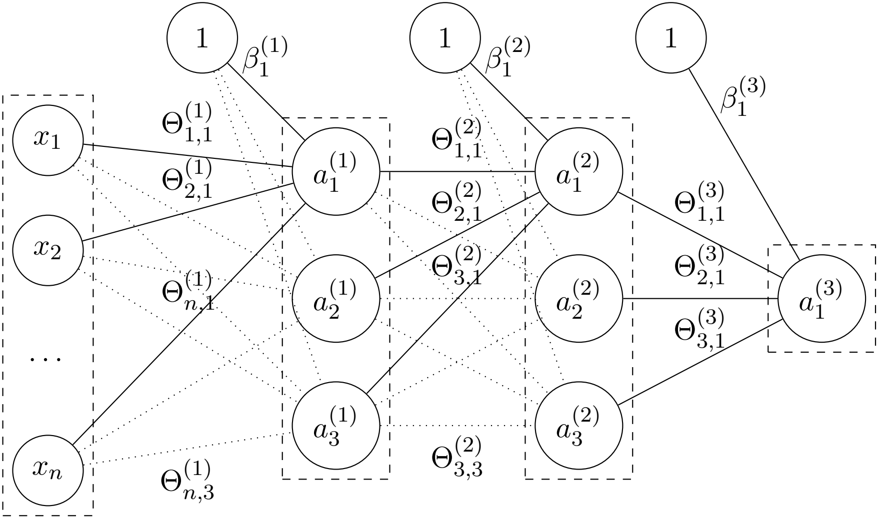
Multi-class logistic regression

- Model: $h_\Theta(x) = [P(k|x,\Theta)]_{k=1,\ldots,c} = \mathrm{softmax}(\Theta x)$ where $\Theta = (\theta^{(1)}, \ldots, \theta^{(c)})$
- Cost function: $J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{c}\delta(y^{(i)},k)\log P(k|x^{(i)},\Theta)$ where $\delta(x,y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$
- Gradient: $\dfrac{\partial J(\Theta)}{\partial \Theta_{j,k}} = -\frac{1}{m}\sum_{i=1}^{m}(\delta(y^{(i)},k) - P(k|x^{(i)},\Theta))\, x_j^{(i)}$
- May look complicated, but can be looked up!

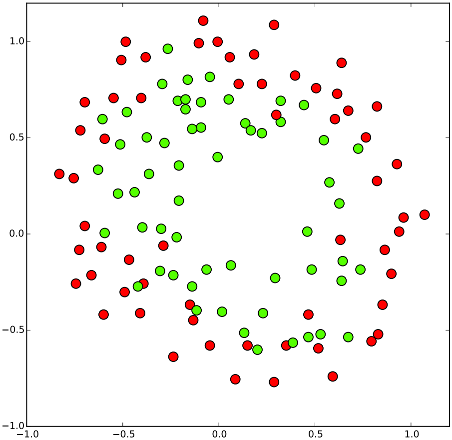
Multi-class logistic regression and neural networks



Deep learning: multi-layer neural networks

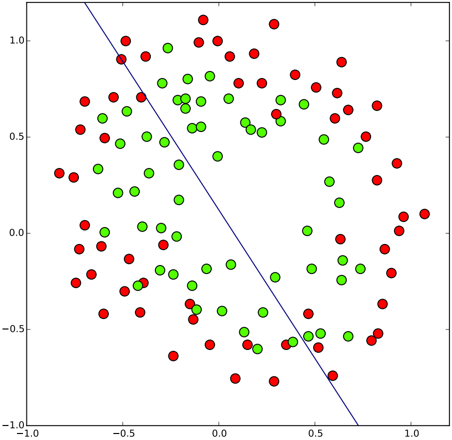


Why multiple-layers?



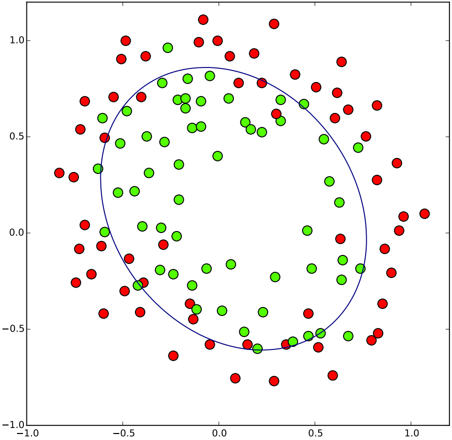
Can a linear model separate these dots?

Why multiple-layers?



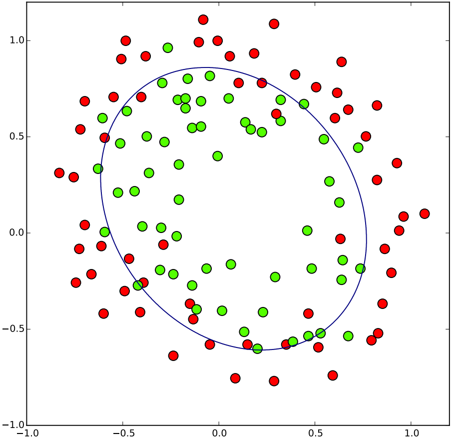
$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Why multiple-layers?



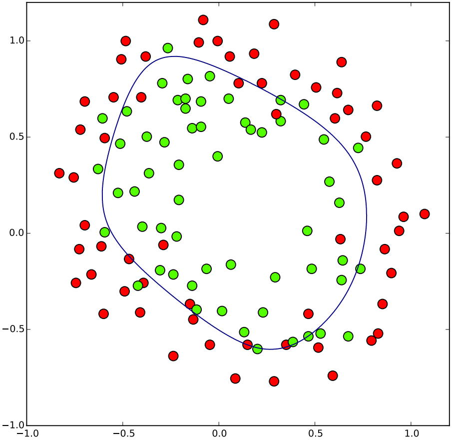
$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1 x_2 + \theta_5 x_2^2$$

Why multiple-layers?



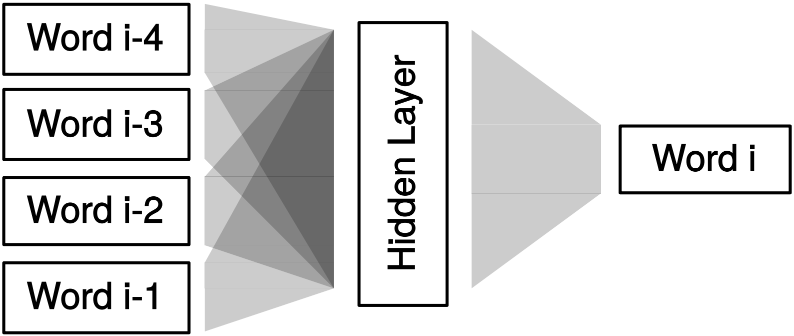
$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 + \theta_5 x_5 \text{ where } x_3 = x_2^2, \ldots$$

Why multiple-layers?



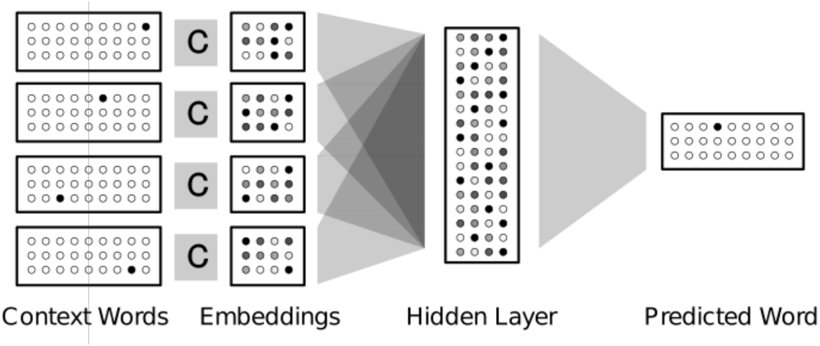
$$h(x) = \sigma(\Theta_2 \sigma(\Theta_1 x)) \text{ where } |\Theta_1| = 3 \times 3, |\Theta_2| = 3 \times 1$$

Feed forward networks language models



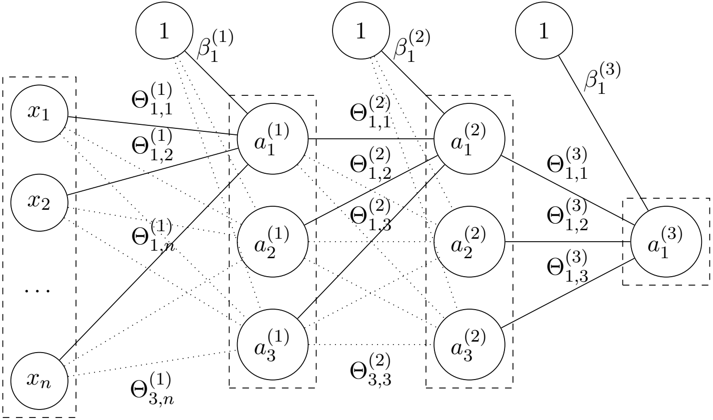
Source: Philipp Koehn, draft chapther on neural machine translation.

Feed forward networks language models



Source: Philipp Koehn, draft chapther on neural machine translation.

Backpropagation – forward step



$$a^{(0)} = x$$

$$z^{(1)} = \Theta^{(1)} a^{(0)} + \beta^{(1)} \qquad z^{(2)} = \Theta^{(2)} a^{(1)} + \beta^{(2)} \qquad z^{(3)} = \Theta^{(3)} a^{(2)} + \beta^{(3)}$$

$$g^{(1)}(x) = \tanh(x) \qquad g^{(2)}(x) = \tanh(x) \qquad g^{(3)}(x) = \tanh(x)$$

$$a^{(1)} = g^{(1)}(z^{(1)}) \qquad a^{(2)} = g^{(2)}(z^{(2)}) \qquad a^{(3)} = g^{(3)}(z^{(3)})$$

The four fundamental equations of Backpropagation

$$
\begin{aligned}
\delta^L &= \nabla_{a^L} J(\Theta) \odot (g^L)'(z^L) && (BP1) \\
\delta^l &= ((\Theta^{l+1})^T \delta^{l+1}) \odot (g^l)'(z^l) && (BP2) \\
\nabla_{\beta^l} J(\Theta) &= \delta^l && (BP3) \\
\nabla_{\Theta^l} J(\Theta) &= a^{l-1} \odot \delta^l && (BP4)
\end{aligned}
$$

The Backpropagation Algorithm

For one training example (x,y):

- Input: Set the activations of the input layers $a^0 = x$
- Forward step: for $l = 1, \ldots, L$ calculate

$$z^l = \Theta^{(l)} a^{l-1} + \beta^l \text{ and } a^l = g^l(z^l)$$

- Output error δ^L: calculate vector

$$\delta^L = \nabla_{a^L} J(\Theta) \odot (g^L)'(z^L)$$

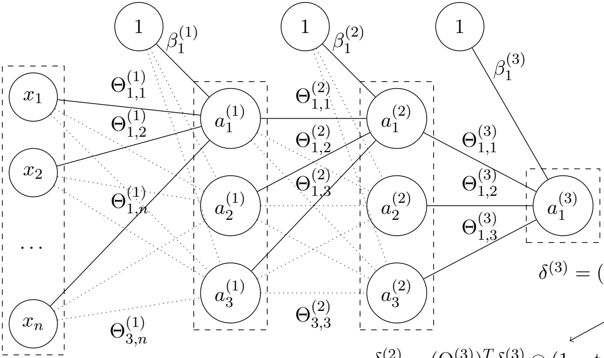
- Error backpropagation: for $l = L-1, L-2, \ldots, 1$ calculate

$$\delta^l = ((\Theta^{l+1})^T \delta^{l+1}) \odot (g^l)'(z^l)$$

- Gradients:

$$\nabla_{\Theta^l} J(\Theta) = a^{l-1} \odot \delta^l \text{ and } \nabla_{\beta^l} J(\Theta) = \delta^l$$

Backpropagation – backward step



$$\delta^{(3)} = (a^{(3)} - y) \odot (1 - \tanh^2(z^{(3)}))$$

$$\delta^{(2)} = (\Theta^{(3)})^T \delta^{(3)} \odot (1 - \tanh^2(z^{(2)}))$$

$$\delta^{(1)} = (\Theta^{(2)})^T \delta^{(2)} \odot (1 - \tanh^2(z^{(1)}))$$

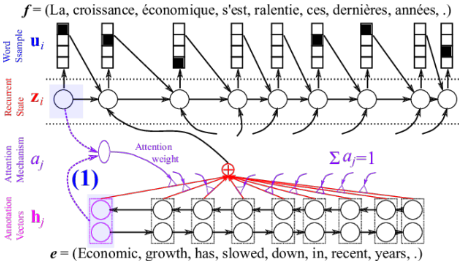
$$a^{(0)} = x \longrightarrow \begin{array}{l} z^{(1)} = \Theta^{(1)} a^{(0)} + \beta^{(1)} \\ g^{(1)}(x) = \tanh(x) \\ a^{(1)} = g^{(1)}(z^{(1)}) \end{array} \longrightarrow \begin{array}{l} z^{(2)} = \Theta^{(2)} a^{(1)} + \beta^{(2)} \\ g^{(2)}(x) = \tanh(x) \\ a^{(2)} = g^{(2)}(z^{(2)}) \end{array} \longrightarrow \begin{array}{l} z^{(3)} = \Theta^{(3)} a^{(2)} + \beta^{(3)} \\ g^{(3)}(x) = \tanh(x) \\ a^{(3)} = g^{(3)}(z^{(3)}) \end{array}$$

Backpropagation and stochastic gradient descent

One iteration:

- For all parameters $\Theta = (\Theta^1, \ldots, \Theta^L)$ create zero-valued helper matrices $\Delta = (\Delta^1, \ldots, \Delta^L)$ of the same size (β omitted for simplicity).
- For m examples in the batch, $i = 1, \ldots, m$:
  - Perform backpropagation for example $(x^{(i)}, y^{(i)})$ and store the gradients $\nabla_\Theta J^{(i)}(\Theta)$
  - $\Delta := \Delta + \frac{1}{m} \nabla_\Theta J^{(i)}(\Theta)$
- Update the weights: $\Theta := \Theta - \alpha\Delta$

More complicated network architectures



- Textbook backprogagation is formulated in terms of layers, weights, biases, activations, weighted inputs, ...
- Actual architectures can contain concatenation of bidirectional RNN states, ...
- What’s the derivation of the "concatenation" operation?

Computing derivatives with reverse-mode autodiff

$$f(x_1, x_2) = \sin(x_1) + x_1 x_2$$

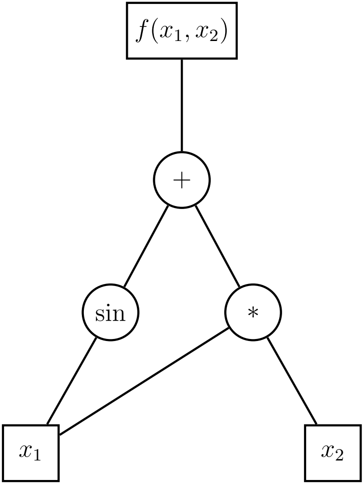
Computing derivatives with reverse-mode autodiff

$$f(x_1, x_2) = \sin(x_1) + x_1 x_2$$

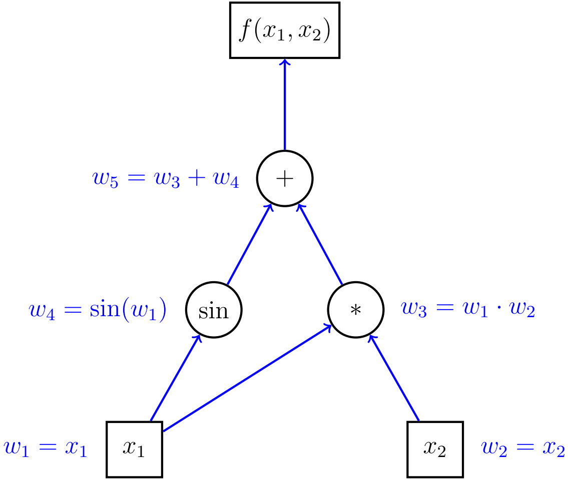
$$\frac{\partial f}{\partial x_1} = ?$$

$$\frac{\partial f}{\partial x_x} = ?$$

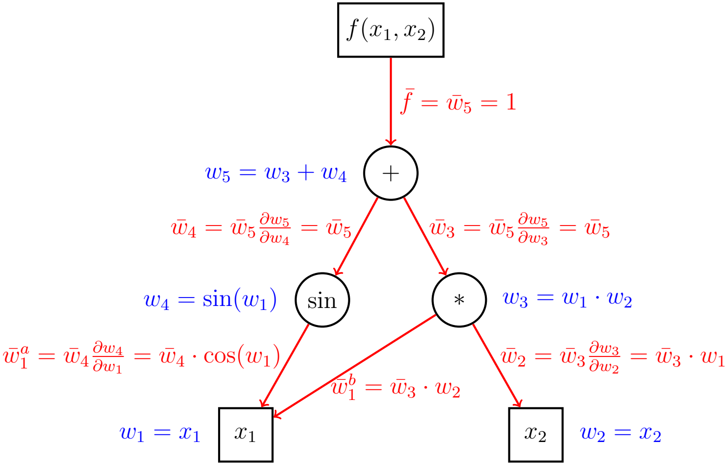
Computation graphs to the rescue



Computation graphs to the rescue



Computation graphs to the rescue

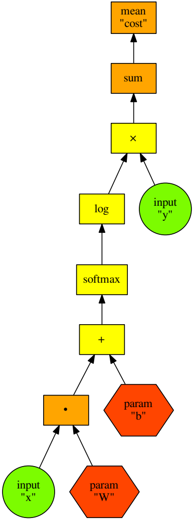


$$\frac{\partial f}{\partial x_1} = \bar{w}_1^a + \bar{w}_1^b = \cos(x_1) + x_2$$

$$\frac{\partial f}{\partial x_2} = \bar{w}_2 = x_1$$

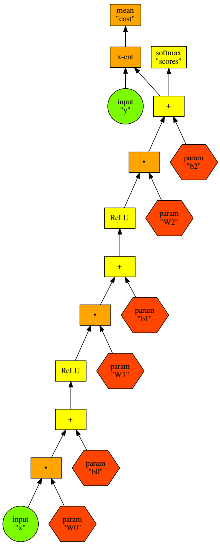
Computation graph for neural networks

$$
\begin{aligned}
a &= \mathrm{softmax}(x \cdot w + b) \\
o &= \mathrm{mean}(\mathrm{sum}(\log(a) \odot y))
\end{aligned}
$$

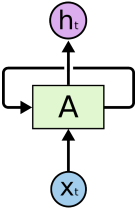


Computation graph for neural networks

$$
\begin{aligned}
a_0 &= x \\
a_1 &= \mathrm{ReLU}(a_0 \cdot w_0 + b_0) \\
a_2 &= \mathrm{ReLU}(a_1 \cdot w_1 + b_1) \\
a_3 &= a_2 \cdot w_2 + b_2 \\
o_1 &= \mathrm{softmax}(a_3) \\
o_2 &= \mathrm{mean}(\mathrm{crossentropy}(a_3, y))
\end{aligned}
$$

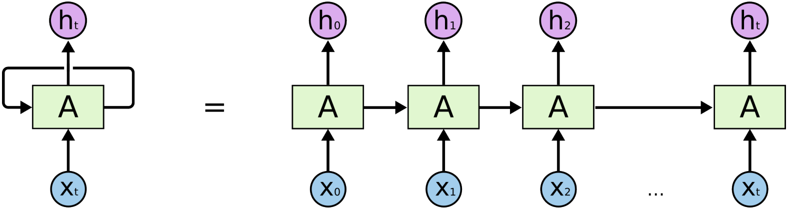


Recurrent neural networks (RNNs)



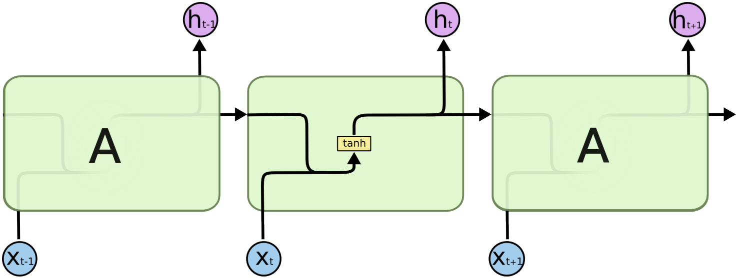
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent neural networks (RNNs)



Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

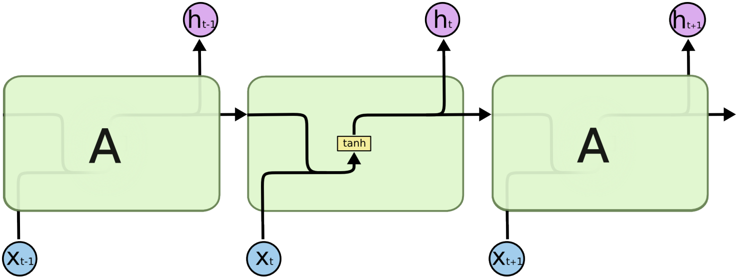
Recurrent neural networks (RNNs)



$$h_t = \tanh(W_h \cdot h_{t-1} + W_x \cdot x_t + b)$$

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs

Recurrent neural networks (RNNs)

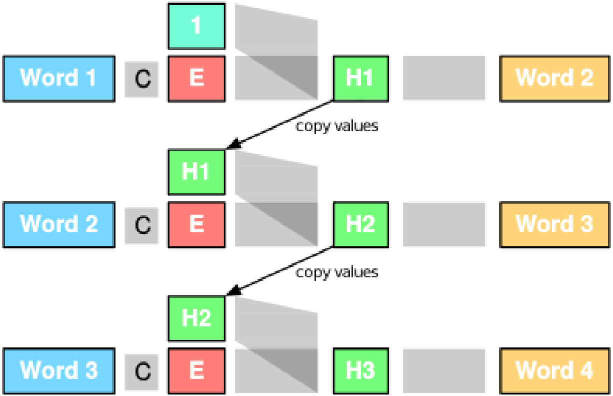


$$h_t = \tanh(W_h \cdot h_{t-1} + W_x \cdot x_t + b)$$

$$h_t = \tanh(W \cdot [h_{t-1}, x_t] + b)$$

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs

Recurrent neural networks language models



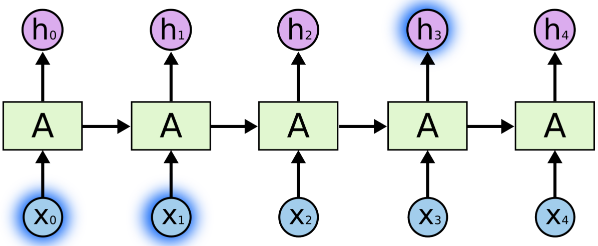
Source: Philipp Koehn, draft chapther on neural machine translation.

Fun with RNNs

Andrej Karpathy: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

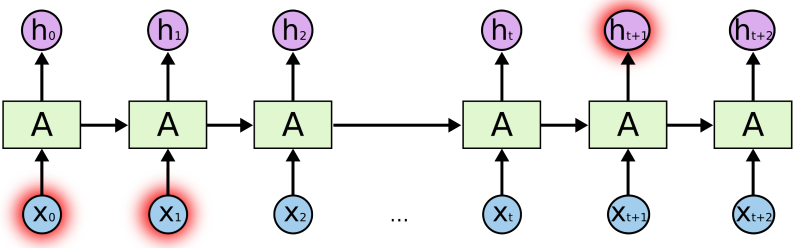
- Character-level language models
- Python code generation
- Poetry generation
- ...

RNNs and long distance dependencies



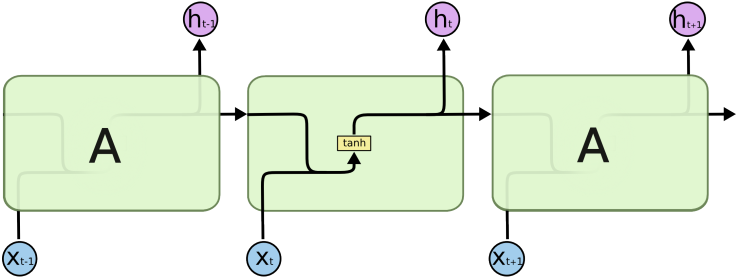
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs

RNNs and long distance dependencies



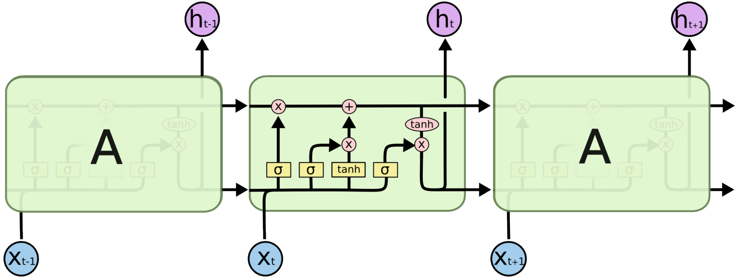
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs

Long Short-Term Memory (LSTM)



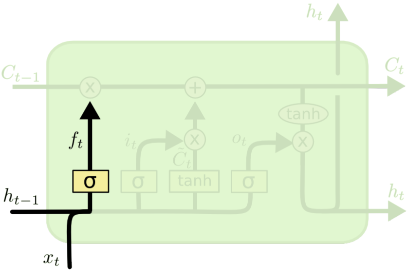
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs

Long Short-Term Memory (LSTM)



Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs

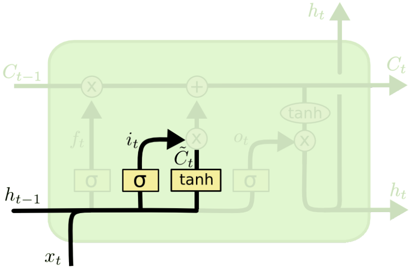
Long Short-Term Memory (LSTM) – Step-by-step



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs

Long Short-Term Memory (LSTM) – Step-by-step

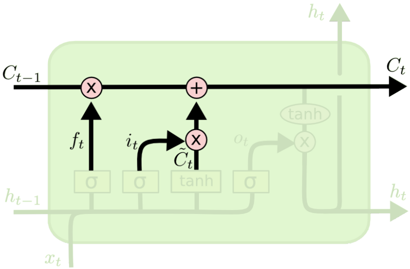


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

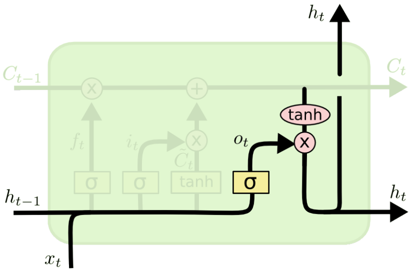
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs

Long Short-Term Memory (LSTM) – Step-by-step



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

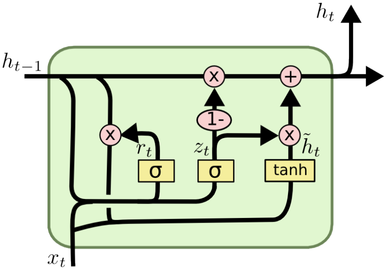
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs

Gated Recurrent Units (GRUs)



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

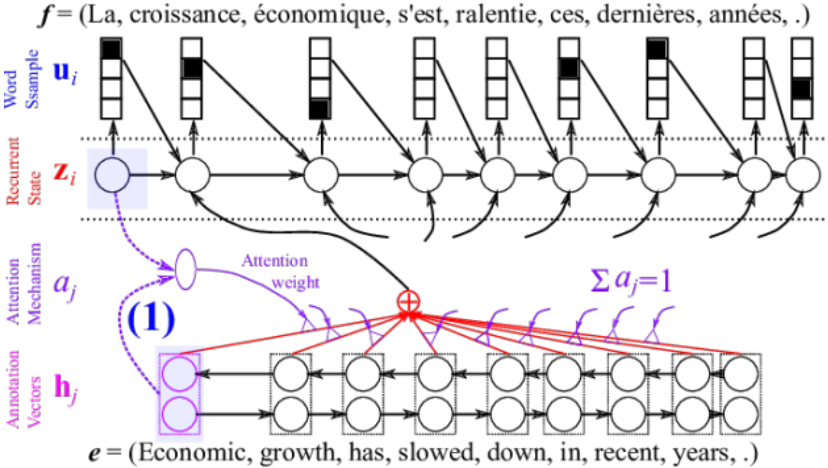
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

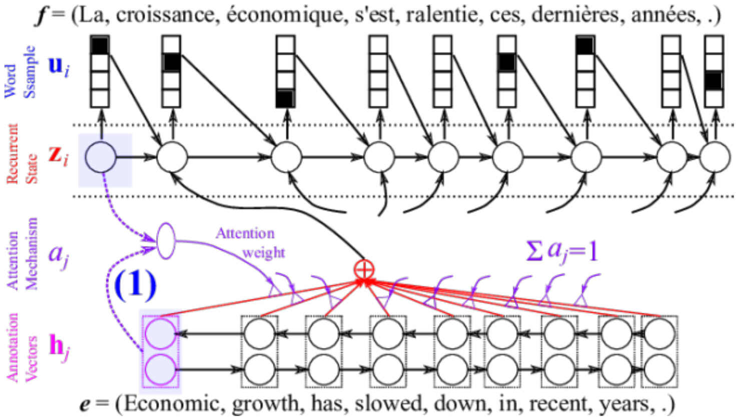
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs

RNNs in Encoder-Decoder architectures



Neural Machine Translation



Kyunghyun Cho
http://devblogs.nvidia.com/parallelforall/introduction-neural-machine-translation-gpus-part-3/

Translation modelling

decomposition of translation problem (for NMT)

- a source sentence S of length m is a sequence $x_1, \dots, x_m$
- a target sentence T of length n is a sequence $y_1, \dots, y_n$

$$
\begin{aligned}
T^* &= \arg\max_t p(T|S) \\
p(T|S) &= p(y_1, \dots, y_n | x_1, \dots, x_m) \\
&= \prod_{i=1}^{n} p(y_i | y_0, \dots, y_{i-1}, x_1, \dots, x_m)
\end{aligned}
$$

Translation modelling

difference from language model

- target-side language model:

$$p(T) = \prod_{i=1}^{n} p(y_i|y_0, \ldots, y_{i-1})$$

- translation model:

$$p(T|S) = \prod_{i=1}^{n} p(y_i|y_0, \ldots, y_{i-1}, x_1, \ldots, x_m)$$

- we could just treat sentence pair as one long sequence, but:
  - we do not care about $p(S)$ (S is given)
  - we may want different vocabulary, network architecture for source text

Translation modelling

difference from language model

- target-side language model:

$$p(T) = \prod_{i=1}^{n} p(y_i|y_0, \ldots, y_{i-1})$$

- translation model:

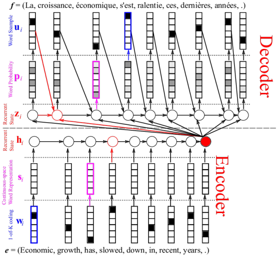
$$p(T|S) = \prod_{i=1}^{n} p(y_i|y_0, \ldots, y_{i-1}, x_1, \ldots, x_m)$$

- we could just treat sentence pair as one long sequence, but:
  - we do not care about $p(S)$ (S is given)
  - we may want different vocabulary, network architecture for source text

Translating with RNNs

Encoder-decoder [Sutskever et al., 2014, Cho et al., 2014]

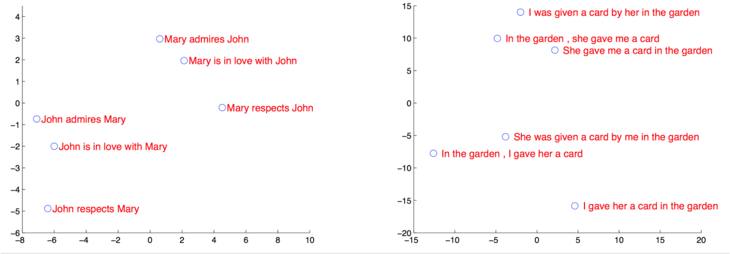
- two RNNs (LSTM or GRU):
  - **encoder** reads input and produces hidden state representations
  - **decoder** produces output, based on last encoder hidden state
- encoder and decoder are learned jointly
  $\rightarrow$ supervision signal from parallel text is backpropagated



Kyunghyun Cho http://devblogs.nvidia.com/parallelforall/introduction-neural-machine-translation-gpus-part-2/

Summary vector

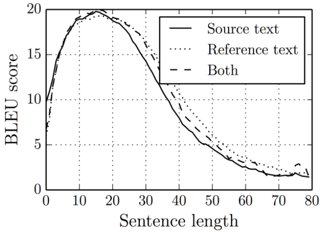
- last encoder hidden-state "summarizes" source sentence
- with multilingual training, we can potentially learn language-independent meaning representation



[Sutskever et al., 2014]

Summary vector as information bottleneck

- can fixed-size vector represent meaning of arbitrarily long sentence?
- empirically, quality decreases for long sentences
- reversing source sentence brings some improvement [Sutskever et al., 2014]

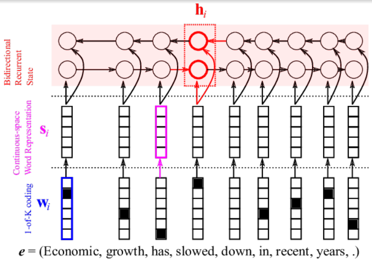


[Sutskever et al., 2014]

Attentional encoder-decoder

encoder

- goal: avoid bottleneck of summary vector
- use bidirectional RNN, and concatenate forward and backward states
  $\rightarrow$ *annotation vector* h_i
- represent source sentence as vector of n annotations
  $\rightarrow$ variable-length representation

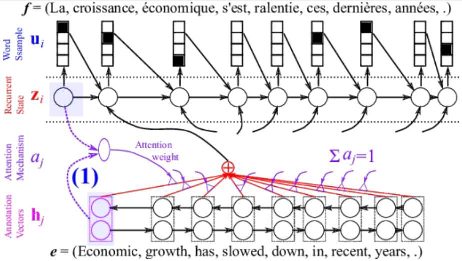


Kyunghyun Cho
http://devblogs.nvidia.com/parallelforall/introduction-neural-machine-translation-gpus-part-3/

Attentional encoder-decoder

attention

- problem: how to incorporate variable-length context into hidden state?
- *attention model* computes *context vector* as weighted average of annotations
- weights are computed by feedforward neural network with softmax activation



Kyunghyun Cho
http://devblogs.nvidia.com/parallelforall/introduction-neural-machine-translation-gpus-part-3/

Attentional encoder-decoder: math

simplifications of model by [Bahdanau et al., 2015] (for illustration)

- plain RNN instead of GRU
- simpler output layer
- we do not show bias terms

notation

- W, U, E, C, V are weight matrices (of different dimensionality)
  - E one-hot to embedding (e.g. $50000 \cdot 512$)
  - W embedding to hidden (e.g. $512 \cdot 1024$)
  - U hidden to hidden (e.g. $1024 \cdot 1024$)
  - C context (2x hidden) to hidden (e.g. $2048 \cdot 1024$)
  - V_o hidden to one-hot (e.g. $1024 \cdot 50000$)
- separate weight matrices for encoder and decoder (e.g. E_x and E_y)
- input X of length T_x; output Y of length T_y

Attentional encoder-decoder: math

encoder

$$\overrightarrow{h}_j = \begin{cases} 0, & \text{, if } j = 0 \\ \tanh(\overrightarrow{W}_x E_x x_j + \overrightarrow{U}_x h_{j-1}) & \text{, if } j > 0 \end{cases}$$

$$\overleftarrow{h}_j = \begin{cases} 0, & \text{, if } j = T_x + 1 \\ \tanh(\overleftarrow{W}_x E_x x_j + \overleftarrow{U}_x h_{j+1}) & \text{, if } j \le T_x \end{cases}$$

$$h_j = (\overrightarrow{h}_j, \overleftarrow{h}_j)$$

Attentional encoder-decoder: math

decoder

$$s_i = \begin{cases} \text{tanh}(W_s \overleftarrow{h}_i), & \text{, if } i = 0 \\ \text{tanh}(W_y E_y y_i + U_y s_{i-1} + C c_i) & \text{, if } i > 0 \end{cases}$$

$$t_i = \text{tanh}(U_o s_{i-1} + W_o E_y y_{i-1} + C_o c_i)$$

$$y_i = \text{softmax}(V_o t_i)$$

attention model

$$e_{ij} = v_a^\top \text{tanh}(W_a s_{i-1} + U_a h_j)$$

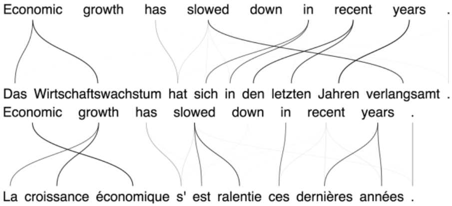
$$\alpha_{ij} = \text{softmax}(e_{ij})$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

Attention model

attention model

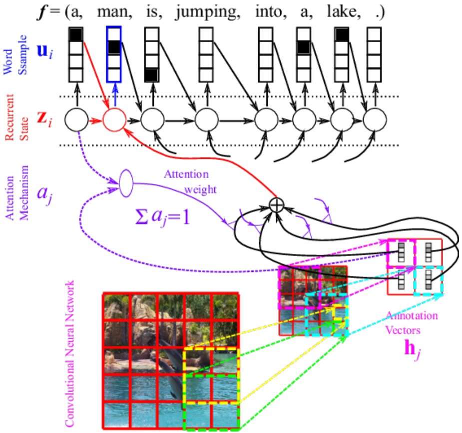
- side effect: we obtain alignment between source and target sentence
- information can also flow along recurrent connections, so there is no guarantee that attention corresponds to alignment
- applications:
  - visualisation
  - replace unknown words with back-off dictionary [Jean et al., 2015]
  - ...



Kyunghyun Cho
http://devblogs.nvidia.com/parallelforall/introduction-neural-machine-translation-gpus-part-3/

Attention model

attention model also works with images:



[Cho et al., 2015]

Attention model

Fig. 5. Examples of the attention-based model attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word) [22]

[Cho et al., 2015]

Applications

score a translation

p(La, croissance, économique, s'est, ralentie, ces, dernières, années, . |
Economic, growth, has, slowed, down, in, recent, year, .) = ?

generate the most probable translation of a source sentence $\rightarrow$ **decoding**

$y^* = \text{argmax}_y \, p(y|$Economic, growth, has, slowed, down, in, recent, year, .$)$

Decoding

exact search

- generate every possible sentence T in target language
- compute score $p(T|S)$ for each
- pick best one

- intractable: $|\text{vocab}|^N$ translations for output length N
  $\rightarrow$ we need approximative search strategy

Decoding

approximative search/1

- at each time step, compute probability distribution $P(y_i|X, y_{<i})$
- select y_i according to some heuristic:
  - sampling: sample from $P(y_i|X, y_{<i})$
  - greedy search: pick $\mathrm{argmax}_y\, p(y_i|X, y_{<i})$
- continue until we generate *<eos>*

- efficient, but suboptimal

Decoding

approximative search/2: **beam search**

- maintain list of K hypotheses (beam)
- at each time step, expand each hypothesis k: $p(y_i^k|X, y_{<i}^k)$
- select K hypotheses with highest total probability:

$$\prod_i p(y_i^k|X, y_{<i}^k)$$

- relatively efficient
- currently default search strategy in neural machine translation
- small beam ($K \approx 10$) offers good speed-quality trade-off

Ensembles

- at each timestep, combine the probability distribution of M different ensemble components.
- combine operator: typically average (log-)probability

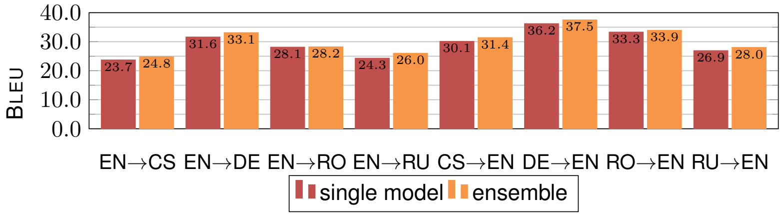
$$\log P(y_i|X, y_{<i}) = \frac{\sum_{m=1}^{M} \log P_m(y_i|X, y_{<i})}{M}$$

- requirements:
  - same output vocabulary
  - same factorization of Y
- internal network architecture may be different
- source representations may be different
  (extreme example: ensemble-like model with different source languages [Junczys-Dowmunt and Grundkiewicz, 2016])

Ensembles

recent ensemble strategies in NMT

- ensemble of 8 independent training runs with different hyperparameters/architectures [Luong et al., 2015a]
- ensemble of 8 independent training runs with different random initializations [Chung et al., 2016]
- ensemble of 4 checkpoints of same training run [Sennrich et al., 2016a]
  $\rightarrow$ probably less effective, but only requires one training run



[Sennrich et al., 2016a]

State of Neural MT

- attentional encoder-decoder networks have become state of the art on various MT tasks
- your mileage may vary depending on
  - language pair and text type
  - amount of training data
  - type of training resources (monolingual?)
  - hyperparameters

Attentional encoder-decoders (NMT) are SOTA

system	BLEU	official rank
uedin-nmt	34.2	1
metamind	32.3	2
uedin-syntax	30.6	3
NYU-UMontreal	30.8	4
online-B	29.4	5-10
KIT/LIMSI	29.1	5-10
cambridge	30.6	5-10
online-A	29.9	5-10
promt-rule	23.4	5-10
KIT	29.0	6-10
jhu-syntax	26.6	11-12
jhu-pbmt	28.3	11-12
uedin-pbmt	28.4	13-14
online-F	19.3	13-15
online-G	23.8	14-15

Table: WMT16 results for EN→DE

system	BLEU	official rank
uedin-nmt	38.6	1
online-B	35.0	2-5
online-A	32.8	2-5
uedin-syntax	34.4	2-5
KIT	33.9	2-6
uedin-pbmt	35.1	5-7
jhu-pbmt	34.5	6-7
online-G	30.1	8
jhu-syntax	31.0	9
online-F	20.2	10

Table: WMT16 results for DE→EN

Attentional encoder-decoders (NMT) are SOTA

system	BLEU	official rank
uedin-nmt	34.2	1
metamind	32.3	2
uedin-syntax	30.6	3
NYU-UMontreal	30.8	4
online-B	29.4	5-10
KIT/LIMSI	29.1	5-10
cambridge	30.6	5-10
online-A	29.9	5-10
promt-rule	23.4	5-10
KIT	29.0	6-10
jhu-syntax	26.6	11-12
jhu-pbmt	28.3	11-12
uedin-pbmt	28.4	13-14
online-F	19.3	13-15
online-G	23.8	14-15

Table: WMT16 results for EN→DE

system	BLEU	official rank
uedin-nmt	38.6	1
online-B	35.0	2-5
online-A	32.8	2-5
uedin-syntax	34.4	2-5
KIT	33.9	2-6
uedin-pbmt	35.1	5-7
jhu-pbmt	34.5	6-7
online-G	30.1	8
jhu-syntax	31.0	9
online-F	20.2	10

Table: WMT16 results for DE→EN

- pure NMT

Attentional encoder-decoders (NMT) are SOTA

system	BLEU	official rank
uedin-nmt	34.2	1
metamind	32.3	2
uedin-syntax	30.6	3
NYU-UMontreal	30.8	4
online-B	29.4	5-10
KIT/LIMSI	29.1	5-10
cambridge	30.6	5-10
online-A	29.9	5-10
promt-rule	23.4	5-10
KIT	29.0	6-10
jhu-syntax	26.6	11-12
jhu-pbmt	28.3	11-12
uedin-pbmt	28.4	13-14
online-F	19.3	13-15
online-G	23.8	14-15

Table: WMT16 results for EN→DE

system	BLEU	official rank
uedin-nmt	38.6	1
online-B	35.0	2-5
online-A	32.8	2-5
uedin-syntax	34.4	2-5
KIT	33.9	2-6
uedin-pbmt	35.1	5-7
jhu-pbmt	34.5	6-7
online-G	30.1	8
jhu-syntax	31.0	9
online-F	20.2	10

Table: WMT16 results for DE→EN

- pure NMT
- NMT component

Attentional encoder-decoders (NMT) are SOTA

uedin-nmt	25.8	1
NYU-UMontreal	23.6	2
jhu-pbmt	23.6	3
cu-chimera	21.0	4-5
cu-tamchyna	20.8	4-5
uedin-cu-syntax	20.9	6-7
online-B	22.7	6-7
online-A	19.5	15
cu-TectoMT	14.7	16
cu-mergedtrees	8.2	18

Table: WMT16 results for EN→CS

uedin-nmt	31.4	1
jhu-pbmt	30.4	2
online-B	28.6	3
PJATK	28.3	8-10
online-A	25.7	11
cu-mergedtrees	13.3	12

Table: WMT16 results for CS→EN

online-B	39.2	1-2
uedin-nmt	33.9	1-2
uedin-pbmt	35.2	3
uedin-syntax	33.6	4-5
online-A	30.8	4-6
jhu-pbmt	32.2	5-7
LIMSI	31.0	6-7

Table: WMT16 results for RO→EN

uedin-nmt	28.1	1-2
QT21-HimL-SysComb	28.9	1-2
KIT	25.8	3-7
uedin-pbmt	26.8	3-7
online-B	25.4	3-7
uedin-lmu-hiero	25.9	3-7
RWTH-SYSCOMB	27.1	3-7
LIMSI	23.9	8-10
lmu-cuni	24.3	8-10
jhu-pbmt	23.5	8-11
usfd-rescoring	23.1	10-12
online-A	19.2	11-12

Table: WMT16 results for EN→RO

Attentional encoder-decoders (NMT) are SOTA

PROMT-rule	22.3	1
amu-uedin	25.3	2-4
online-B	23.8	2-5
uedin-nmt	26.0	2-5
online-G	26.2	3-5
NYU-UMontreal	23.1	6
jhu-pbmt	24.0	7-8
LIMSI	23.6	7-10
online-A	20.2	8-10
AFRL-MITLL-phr	23.5	9-10
AFRL-MITLL-verb	20.9	11
online-F	8.6	12

Table: WMT16 results for EN→RU

uedin-pbmt	23.4	1-4
online-G	20.6	1-4
online-B	23.6	1-4
UH-opus	23.1	1-4
PROMT-SMT	20.3	5
UH-factored	19.3	6-7
uedin-syntax	20.4	6-7
online-A	19.0	8
jhu-pbmt	19.1	9

Table: WMT16 results for FI→EN

amu-uedin	29.1	1-2
online-G	28.7	1-3
NRC	29.1	2-4
online-B	28.1	3-5
uedin-nmt	28.0	4-5
online-A	25.7	6-7
AFRL-MITLL-phr	27.6	6-7
AFRL-MITLL-contrast	27.0	8-9
PROMT-rule	20.4	8-9
online-F	13.5	10

Table: WMT16 results for RU→EN

online-G	15.4	1-3
abumatra-nmt	17.2	1-4
online-B	14.4	1-4
abumatran-combo	17.4	3-5
UH-opus	16.3	4-5
NYU-UMontreal	15.1	6-8
abumatran-pbsmt	14.6	6-8
online-A	13.0	6-8
jhu-pbmt	13.8	9-10
UH-factored	12.8	9-12
aalto	11.6	10-13
jhu-hltcoe	11.9	10-13
UUT	11.6	11-13

Table: WMT16 results for EN→FI

Interlude: why is (machine) translation hard?

ambiguity

words are often polysemous, with different translations for different meanings

system	sentence
source	Dort wurde er von dem Schläger und einer weiteren männlichen Person erneut angegriffen.
reference	There he was attacked again by his original attacker and another male.
uedin-nmt	There he was attacked again by the racket and another male person.
uedin-pbsmt	There, he was at the club and another male person attacked again.

Schläger

Interlude: why is (machine) translation hard?

ambiguity

words are often polysemous, with different translations for different meanings

system	sentence
source	Dort wurde er von dem Schläger und einer weiteren männlichen Person erneut angegriffen.
reference	There he was attacked again by his original attacker and another male.
uedin-nmt	There he was attacked again by the racket and another male person.
uedin-pbsmt	There, he was at the club and another male person attacked again.

Schläger

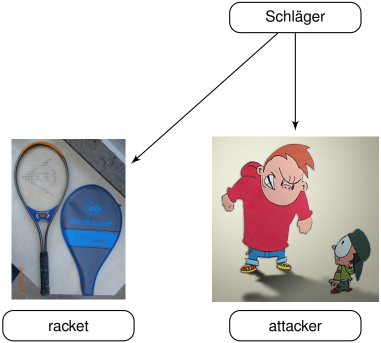
racket

Interlude: why is (machine) translation hard?

ambiguity

words are often polysemous, with different translations for different meanings

system	sentence
source	Dort wurde er von dem Schläger und einer weiteren männlichen Person erneut angegriffen.
reference	There he was attacked again by his original attacker and another male.
uedin-nmt	There he was attacked again by the racket and another male person.
uedin-pbsmt	There, he was at the club and another male person attacked again.

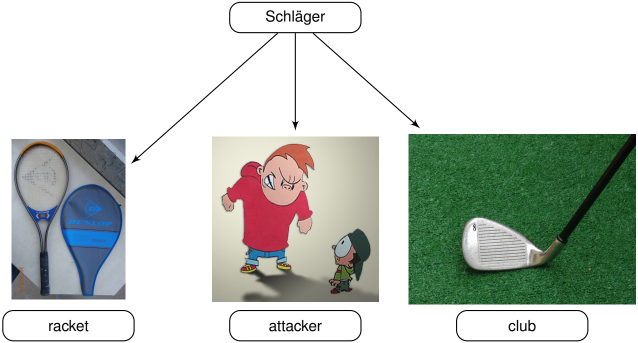


Interlude: why is (machine) translation hard?

ambiguity

words are often polysemous, with different translations for different meanings

system	sentence
source	Dort wurde er von dem Schläger und einer weiteren männlichen Person erneut angegriffen.
reference	There he was attacked again by his original attacker and another male.
uedin-nmt	There he was attacked again by the racket and another male person.
uedin-pbsmt	There, he was at the club and another male person attacked again.



Interlude: why is (machine) translation hard?

word order

there are systematic word order differences between languages. We need to generate words in the correct order.

system	sentence
source	Unsere digitalen Leben haben die Notwendigkeit, stark, lebenslustig und erfolgreich zu erscheinen, verdoppelt [...]
reference	Our digital lives have doubled the need to appear strong, fun-loving and successful [...]
uedin-nmt	Our digital lives have doubled the need to appear strong, lifelike and successful [...]
uedin-pbsmt	Our digital lives are lively, strong, and to be successful, doubled [...]

Interlude: why is (machine) translation hard?

grammatical marking system

grammatical distinctions can be marked in different ways, for instance through word order (English), or inflection (German). The translator needs to produce the appropriate marking.

English	... because the dog chased the man.
German	... weil der Hund den Mann jagte.

Interlude: why is (machine) translation hard?

multiword expressions

the meaning of non-compositional expressions is lost in a word-to-word translation

system	sentence
source	He bends over backwards for the team, ignoring any pain.
reference	Er zerreißt sich für die Mannschaft, geht über Schmerzen drüber. (lit: he tears himself apart for the team)
uedin-nmt	Er beugt sich rückwärts für die Mannschaft, ignoriert jeden Schmerz. (lit: he bends backwards for the team)
uedin-pbsmt	Er macht alles für das Team, den Schmerz zu ignorieren. (lit: he does everything for the team)

Interlude: why is (machine) translation hard?

subcategorization

Words only allow for specific categories of syntactic arguments, that often differ between languages.

English	he remembers his medical appointment.
German	er erinnert sich an seinen Arzttermin.
English	*he remembers himself to his medical appointment.
German	*er erinnert seinen Arzttermin.

agreement

inflected forms may need to agree over long distances to satisfy grammaticality.

English	they can not be found
French	**elles** ne **peuvent** pas être **trouvées**

Interlude: why is (machine) translation hard?

morphological complexity

translator may need to analyze/generate morphologically complex words that were not seen before.

German	Abwasserbehandlungsanlage
English	waste water treatment plant
French	station d'épuration des eaux résiduaires

system	sentence
source	Titelverteidiger ist Drittligaabsteiger SpVgg Unterhaching.
reference	The defending champions are SpVgg Unterhaching, who have been relegated to the third league.
uedin-nmt	Defending champion is third-round pick SpVgg Underhaching.
uedin-pbsmt	Title defender Drittligaabsteiger Week 2.

Interlude: why is (machine) translation hard?

open vocabulary

languages have an open vocabulary, and we need to learn translations for words that we have only seen rarely (or never)

system	sentence
source	Titelverteidiger ist Drittligaabsteiger SpVgg Unterhaching.
reference	The defending champions are SpVgg Unterhaching, who have been relegated to the third league.
uedin-nmt	Defending champion is third-round pick SpVgg Underhaching.
uedin-pbsmt	Title defender Drittligaabsteiger Week 2.

Interlude: why is (machine) translation hard?

discontinuous structures

a word (sequence) can map to a discontinuous structure in another language.

English	I do **not** know
French	Je **ne** sais **pas**

system	sentence
source	Ein Jahr später machten die Fed-Repräsentanten diese Kürzungen rückgängig.
reference	A year later, Fed officials reversed those cuts.
uedin-nmt	A year later, FedEx officials reversed those cuts.
uedin-pbsmt	A year later, the Fed representatives made these cuts.

Interlude: why is (machine) translation hard?

discourse

the translation of referential expressions depends on discourse context, which sentence-level translators have no access to.

English	I made a decision.	Please respect it.
French	J'ai pris une décision.	Respectez-la s'il vous plaît.
French	J'ai fait un choix.	Respectez-le s'il vous plaît.

Interlude: why is (machine) translation hard?

assorted other difficulties

- underspecification
- ellipsis
- lexical gaps
- language change
- language variation (dialects, genres, domains)
- ill-formed input

Comparison between phrase-based and neural MT

human analysis of NMT (reranking) [Neubig et al., 2015]

- NMT is more grammatical
  - word order
  - insertion/deletion of function words
  - morphological agreement
- minor degradation in lexical choice?

Comparison between phrase-based and neural MT

analysis of IWSLT 2015 results [Bentivogli et al., 2016]

- human-targeted translation error rate (HTER) based on automatic translation and human post-edit
- 4 error types: substitution, insertion, deletion, shift

system	HTER (no *shift*)			HTER
	word	lemma	%Δ	(*shift* only)
PBSMT [Ha et al., 2015]	28.3	23.2	-18.0	3.5
NMT [Luong and Manning, 2015]	21.7	18.7	-13.7	1.5

- word-level is closer to lemma-level performance: better at inflection/agreement
- improvement on lemma-level: better lexical choice
- fewer shift errors: better word order

Comparison between phrase-based and neural MT

analysis of IWSLT 2015 results [Bentivogli et al., 2016]

- human-targeted translation error rate (HTER) based on automatic translation and human post-edit
- 4 error types: substitution, insertion, deletion, shift

system	HTER (no *shift*) word	HTER (no *shift*) lemma	HTER (no *shift*) %Δ	HTER (*shift* only)
PBSMT [Ha et al., 2015]	28.3	23.2	-18.0	3.5
NMT [Luong and Manning, 2015]	21.7	18.7	-13.7	1.5

- word-level is closer to lemma-level performance: better at inflection/agreement
- improvement on lemma-level: better lexical choice
- fewer shift errors: better word order

Comparison between phrase-based and neural MT

analysis of IWSLT 2015 results [Bentivogli et al., 2016]

- human-targeted translation error rate (HTER) based on automatic translation and human post-edit
- 4 error types: substitution, insertion, deletion, shift

system	HTER (no *shift*) word	HTER (no *shift*) lemma	HTER (no *shift*) %Δ	HTER (*shift* only)
PBSMT [Ha et al., 2015]	28.3	23.2	-18.0	3.5
NMT [Luong and Manning, 2015]	21.7	18.7	-13.7	1.5

- word-level is closer to lemma-level performance: better at inflection/agreement
- improvement on lemma-level: better lexical choice
- fewer shift errors: better word order

Comparison between phrase-based and neural MT

analysis of IWSLT 2015 results [Bentivogli et al., 2016]

- human-targeted translation error rate (HTER) based on automatic translation and human post-edit
- 4 error types: substitution, insertion, deletion, shift

system	HTER (no *shift*)			HTER
	word	lemma	%Δ	(*shift* only)
PBSMT [Ha et al., 2015]	28.3	23.2	-18.0	3.5
NMT [Luong and Manning, 2015]	21.7	18.7	-13.7	1.5

- word-level is closer to lemma-level performance: better at inflection/agreement
- improvement on lemma-level: better lexical choice
- fewer shift errors: better word order

Fluency

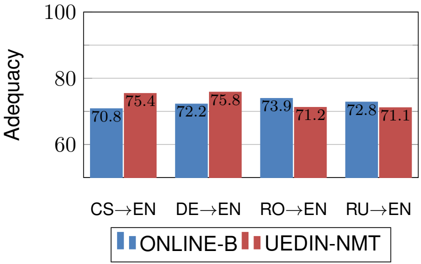


Figure: WMT16 direct assessment results

Fluency

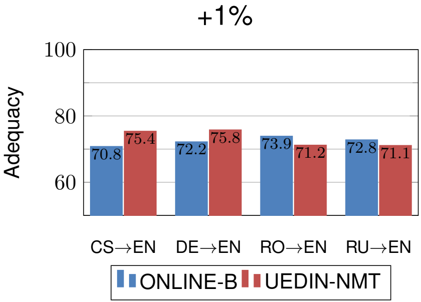


Figure: WMT16 direct assessment results

Fluency

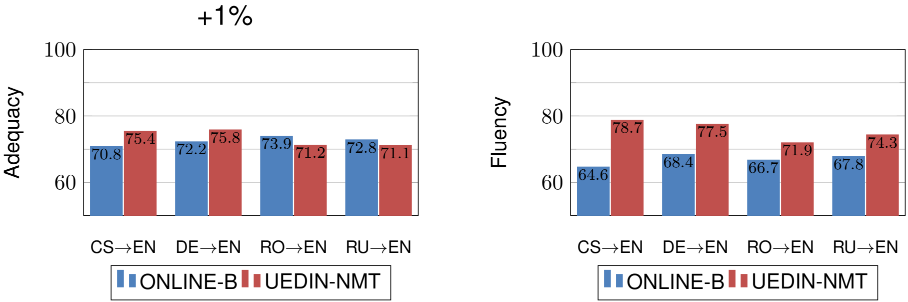


Figure: WMT16 direct assessment results

Fluency

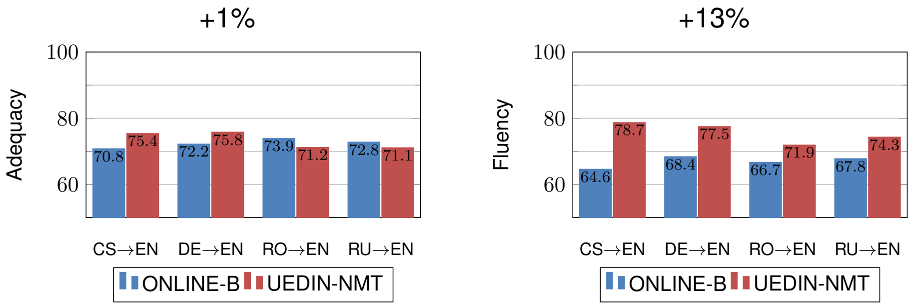


Figure: WMT16 direct assessment results

Why is neural MT output more grammatical?

phrase-based SMT

- log-linear combination of many “weak” features
- data sparsenesss triggers back-off to smaller units
- strong independence assumptions

neural MT

- end-to-end trained model
- generalization via continuous space representation
- output conditioned on full source text and target history

Efficiency

speed bottlenecks

- matrix multiplication
  $\rightarrow$ use of highly parallel hardware (GPUs)
- size of output layer scales with vocabulary size. Solutions:
  - LMs: hierarchical softmax; noise-contrastive estimation; self-normalization
  - NMT: approximate softmax through subset of vocabulary [Jean et al., 2015, Mi et al., 2016, L'Hostis et al., 2016]

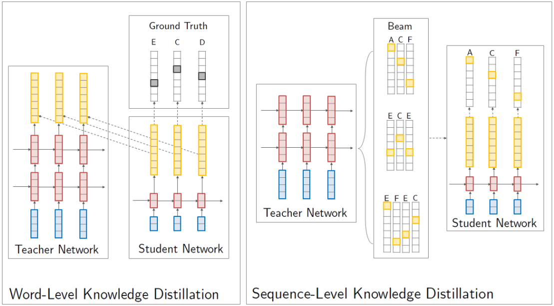
NMT training vs. decoding (on fast GPU)

- training: slow (1-3 weeks)
- decoding: fast (100 000–500 000 sentences / day)[a]

[a] with NVIDIA Titan X and amuNMT (`https://github.com/emjotde/amunmt`)

Efficiency

- aggressive batching during decoding
  - compute all prefixes in beam in single batch
  - compute multiple sentences in single batch
- 8-bit inference [Wu et al., 2016]
- knowledge distillation: student network mimics teacher [Kim and Rush, 2016]



Open-vocabulary translation

Why is vocabulary size a problem?

- size of one-hot input/output vector is linear to vocabulary size
- large vocabularies are space inefficient
- large output vocabularies are time inefficient
- typical network vocabulary size: 30 000–100 000

What about out-of-vocabulary words?

- training set vocabulary typically larger than network vocabulary (1 million words or more)
- at translation time, we regularly encounter novel words:
    - names: *Barack Obama*
    - morph. complex words: *Hand|gepäck|gebühr* ('carry-on bag fee')
    - numbers, URLs etc.

Open-vocabulary translation

Solutions

- copy unknown words, or translate with back-off dictionary [Jean et al., 2015, Luong et al., 2015b, Gulcehre et al., 2016]
  $\rightarrow$ works for names (if alphabet is shared), and 1-to-1 aligned words
- use subword units (characters or others) for input/output vocabulary
  $\rightarrow$ model can learn translation of seen words on subword level
  $\rightarrow$ model can translate unseen words if translation is *transparent*
- active research area [Sennrich et al., 2016c, Luong and Manning, 2016, Chung et al., 2016, Ling et al., 2015, Costa-jussà and Fonollosa, 2016, Zhao and Zhang, 2016, Lee et al., 2016]

Core idea: transparent translations

transparent translations

- some translations are semantically/phonologically transparent
- morphologically complex words (e.g. compounds):
  - solar system (English)
  - Sonnen|system (German)
  - Nap|rendszer (Hungarian)
- named entities:
  - Obama(English; German)
  - Обама (Russian)
  - オバマ (o-ba-ma) (Japanese)
- cognates and loanwords:
  - claustrophobia(English)
  - Klaustrophobie(German)
  - Клаустрофобия (Russian)

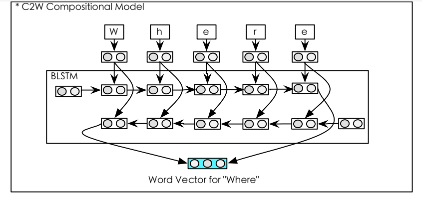
Subword neural machine translation

Flat representation [Sennrich et al., 2016c, Chung et al., 2016]

- sentence is a sequence of subword units

Hierarchical representation [Ling et al., 2015, Luong and Manning, 2016]

- sentence is a sequence of words
- words are a sequence of subword units



open question: should attention be on level of words or subwords?

Subword neural machine translation

Choice of subword unit

- characters: small vocabulary, long sequences
- morphemes (?): hard to control vocabulary size
- hybrid choice: shortlist of words, subwords for rare words
- variable-length character n-grams: byte-pair encoding (BPE)

open research question which subword segmentation is best choice in terms of *efficiency* and *effectiveness*.

Byte pair encoding [Gage, 1994]

algorithm

iteratively replace most frequent byte pair in sequence with unused byte

aaabdaaabac

Byte pair encoding [Gage, 1994]

algorithm

iteratively replace most frequent byte pair in sequence with unused byte

aaabdaaabac
ZabdZabac

Z=aa

Byte pair encoding [Gage, 1994]

algorithm

iteratively replace most frequent byte pair in sequence with unused byte

aaabdaaabac	
ZabdZabac	Z=aa
ZYdZYac	Y=ab

Byte pair encoding [Gage, 1994]

algorithm

iteratively replace most frequent byte pair in sequence with unused byte

aaabdaaabac	
ZabdZabac	Z=aa
ZYdZYac	Y=ab
XdXac	X=ZY

Byte pair encoding for word segmentation

bottom-up character merging

- iteratively replace most frequent pair of symbols ('A','B') with 'AB'
- apply on dictionary, not on full text (for efficiency)
- output vocabulary: character vocabulary + one symbol per merge

word	freq
'l o w </w>'	5
'l o w e r </w>'	2
'n e w e s t </w>'	6
'w i d e s t </w>'	3

freq	symbol pair	new symbol

Byte pair encoding for word segmentation

bottom-up character merging

- iteratively replace most frequent pair of symbols ('A','B') with 'AB'
- apply on dictionary, not on full text (for efficiency)
- output vocabulary: character vocabulary + one symbol per merge

word	freq
'l o w </w>'	5
'l o w e r </w>'	2
'n e w es t </w>'	6
'w i d es t </w>'	3

freq	symbol pair		new symbol
9	('e', 's')	$\rightarrow$	'es'

Byte pair encoding for word segmentation

bottom-up character merging

- iteratively replace most frequent pair of symbols ('A','B') with 'AB'
- apply on dictionary, not on full text (for efficiency)
- output vocabulary: character vocabulary + one symbol per merge

word	freq
'l o w </w>'	5
'l o w e r </w>'	2
'n e w est </w>'	6
'w i d est </w>'	3

freq	symbol pair		new symbol
9	('e', 's')	→	'es'
9	('es', 't')	→	'est'

Byte pair encoding for word segmentation

bottom-up character merging

- iteratively replace most frequent pair of symbols ('A','B') with 'AB'
- apply on dictionary, not on full text (for efficiency)
- output vocabulary: character vocabulary + one symbol per merge

word	freq
'l o w </w>'	5
'l o w e r </w>'	2
'n e w est</w>'	6
'w i d est</w>'	3

freq	symbol pair		new symbol
9	('e', 's')	$\rightarrow$	'es'
9	('es', 't')	$\rightarrow$	'est'
9	('est', '</w>')	$\rightarrow$	'est</w>'

Byte pair encoding for word segmentation

bottom-up character merging

- iteratively replace most frequent pair of symbols ('A','B') with 'AB'
- apply on dictionary, not on full text (for efficiency)
- output vocabulary: character vocabulary + one symbol per merge

word	freq
'lo w </w>'	5
'lo w e r </w>'	2
'n e w est</w>'	6
'w i d est</w>'	3

freq	symbol pair		new symbol
9	('e', 's')	$\rightarrow$	'es'
9	('es', 't')	$\rightarrow$	'est'
9	('est', '</w>')	$\rightarrow$	'est</w>'
7	('l', 'o')	$\rightarrow$	'lo'

Byte pair encoding for word segmentation

bottom-up character merging

- iteratively replace most frequent pair of symbols ('A','B') with 'AB'
- apply on dictionary, not on full text (for efficiency)
- output vocabulary: character vocabulary + one symbol per merge

word	freq
'low </w>'	5
'low e r </w>'	2
'n e w est</w>'	6
'w i d est</w>'	3

freq	symbol pair		new symbol
9	('e', 's')	$\rightarrow$	'es'
9	('es', 't')	$\rightarrow$	'est'
9	('est', '</w>')	$\rightarrow$	'est</w>'
7	('l', 'o')	$\rightarrow$	'lo'
7	('lo', 'w')	$\rightarrow$	'low'
...			

Byte pair encoding for word segmentation

why BPE?

- don't waste time on frequent character sequences
  → trade-off between text length and vocabulary sizes
- open-vocabulary:
  learned operations can be applied to unknown words
- alternative view: character-level model on compressed text

'l o w e s t </w>'

('e', 's')	→	'es'
('es', 't')	→	'est'
('est', '</w>')	→	'est</w>'
('l', 'o')	→	'lo'
('lo', 'w')	→	'low'

Byte pair encoding for word segmentation

why BPE?

- don't waste time on frequent character sequences
  → trade-off between text length and vocabulary sizes
- open-vocabulary:
  learned operations can be applied to unknown words
- alternative view: character-level model on compressed text

'l o w es t </w>'

('e', 's')	→	'es'
('es', 't')	→	'est'
('est', '</w>')	→	'est</w>'
('l', 'o')	→	'lo'
('lo', 'w')	→	'low'

Byte pair encoding for word segmentation

why BPE?

- don't waste time on frequent character sequences
  → trade-off between text length and vocabulary sizes
- open-vocabulary:
  learned operations can be applied to unknown words
- alternative view: character-level model on compressed text

'l o w est </w>'

('e', 's')	→	'es'
('es', 't')	→	'est'
('est', '</w>')	→	'est</w>'
('l', 'o')	→	'lo'
('lo', 'w')	→	'low'

Byte pair encoding for word segmentation

why BPE?

- don't waste time on frequent character sequences
  $\rightarrow$ trade-off between text length and vocabulary sizes
- open-vocabulary:
  learned operations can be applied to unknown words
- alternative view: character-level model on compressed text

'l o w est</w>'

('e', 's')	$\rightarrow$	'es'
('es', 't')	$\rightarrow$	'est'
('est', '</w>')	$\rightarrow$	'est</w>'
('l', 'o')	$\rightarrow$	'lo'
('lo', 'w')	$\rightarrow$	'low'

Byte pair encoding for word segmentation

why BPE?

- don't waste time on frequent character sequences
  → trade-off between text length and vocabulary sizes
- open-vocabulary:
  learned operations can be applied to unknown words
- alternative view: character-level model on compressed text

'lo w est</w>'

('e', 's')	→	'es'
('es', 't')	→	'est'
('est', '</w>')	→	'est</w>'
('l', 'o')	→	'lo'
('lo', 'w')	→	'low'

Byte pair encoding for word segmentation

why BPE?

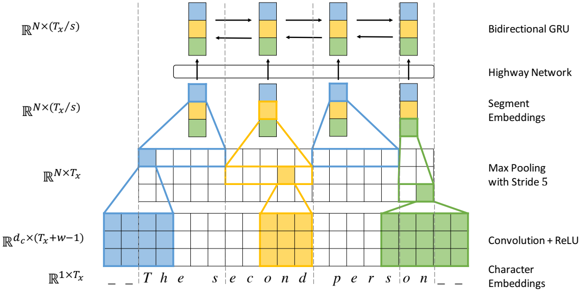
- don't waste time on frequent character sequences
  → trade-off between text length and vocabulary sizes
- open-vocabulary:
  learned operations can be applied to unknown words
- alternative view: character-level model on compressed text

'low est</w>'

('e', 's')	→	'es'
('es', 't')	→	'est'
('est', '</w>')	→	'est</w>'
('l', 'o')	→	'lo'
('lo', 'w')	→	'low'

Fully Character-level NMT [Lee et al., 2016]

- character-to-character model requires no language-specific segmentation
- drawback: RNN over characters is slow (especially attention!)
- (shorter) segment sequences are obtained from characters via convolution and max-pooling layers



Architecture variants

an incomplete selection

- different encoder architectures:
  - convolution network [Kalchbrenner and Blunsom, 2013, Kalchbrenner et al., 2016]
  - TreeLSTM [Eriguchi et al., 2016]
- modifications to attention mechanism [Luong et al., 2015a, Feng et al., 2016, Zhang et al., 2016]
- deeper networks [Zhou et al., 2016, Wu et al., 2016]
- coverage model [Mi et al., 2016, Tu et al., 2016b, Tu et al., 2016a]

Sequence-level training

- problem: at training time, target-side history is reliable;
  at test time, it is not.
- solution: instead of using gold context, sample from the model to obtain target context [Shen et al., 2016, Ranzato et al., 2016, Bengio et al., 2015, Wiseman and Rush, 2016]
- more efficient cross entropy training remains in use to initialize weights

Trading-off target and source context

system	sentence
source	Ein Jahr später machten die Fed-Repräsentanten diese Kürzungen rückgängig.
reference	A year later, Fed officials reversed those cuts.
uedin-nmt	A year later, FedEx officials reversed those cuts.
uedin-pbsmt	A year later, the Fed representatives made these cuts.

problem

- RNN is locally normalized at each time step
- given *Fed:* as previous word, *Ex* is very likely in training data: $p(\text{Ex}|\text{Fed:}) = 0.55$
- *label bias problem*: locally-normalized models may ignore input in low-entropy state

potential solutions (speculative)

- sampling at training time
- bidirectional decoder [Liu et al., 2016, Sennrich et al., 2016a]
- context gates to trade-off source and target context [Tu et al., 2016]

Monolingual Training Data

why monolingual data for phrase-based SMT?

- more training data ✓
- more appropriate training data (domain adaptation) ✓
- relax independence assumptions ✓

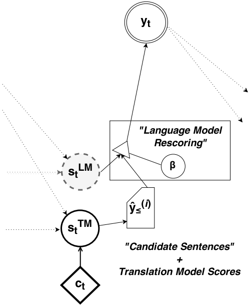
why monolingual data for neural MT?

- more training data ✓
- more appropriate training data (domain adaptation) ✓
- relax independence assumptions ✗

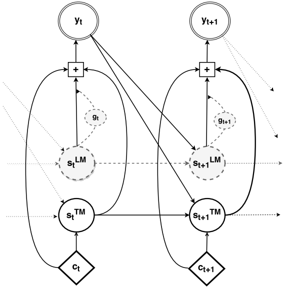
Training data: monolingual

Solutions/1

- shallow fusion: rescore beam with language model [Gülçehre et al., 2015]
- deep fusion: extra, LM-specific hidden layer [Gülçehre et al., 2015]



(a) Shallow Fusion (Sec. 4.1)



(b) Deep Fusion (Sec. 4.2)

[Gülçehre et al., 2015]

Training data: monolingual

Solutions/2

- decoder is already a language model. Train encoder-decoder with added monolingual data [Sennrich et al., 2016b]

$$t_i = \tanh(U_o s_{i-1} + V_o E_y y_{i-1} + C_o c_i)$$
$$y_i = \text{softmax}(W_o t_i)$$

- how do we get approximation of context vector c_i?
  - dummy source context (moderately effective)
  - automatically back-translate monolingual data into source language

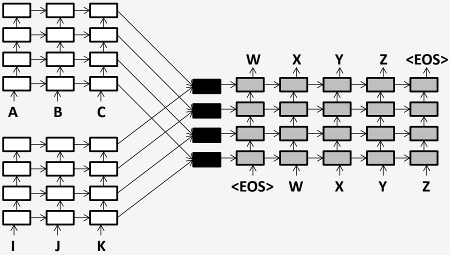
name	2014	2015
PBSMT [Haddow et al., 2015]	28.8	29.3
NMT [Gülçehre et al., 2015]	23.6	-
shallow fusion [Gülçehre et al., 2015]	23.7	-
deep fusion [Gülçehre et al., 2015]	24.0	-
NMT baseline	25.9	26.7
+back-translated monolingual data	**29.5**	**30.4**

Table: DE→EN translation performance (BLEU) on WMT training/test sets.

Training data: multilingual

Multi-source translation [Zoph and Knight, 2016]

we can condition on multiple input sentences

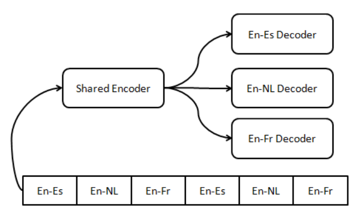


- benefits:
  - one source text may contain information that is undespecified in other $\rightarrow$ possible quality gains
- drawbacks:
  - we need multiple source sentences at training and decoding time

Training data: multilingual

Multilingual models [Dong et al., 2015, Firat et al., 2016a]

we can share layers (encoder/decoder/attention) of the model across language pairs

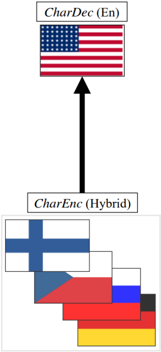


- benefits:
  - transfer learning from one language pair to the other
  - scalability: no need for $N^2 - N$ independent models for N languages
- drawbacks:
  - no successful generalization to language pairs with no training data (but: synthetic training data works: [Firat et al., 2016b])

Training data: multilingual

Multilingual models [Lee et al., 2016]

- single, character level encoder trained on multiple languages
  - more compact model
  - occasional quality improvements over single language pairs
  - robust towards (synthetic) code-switched input

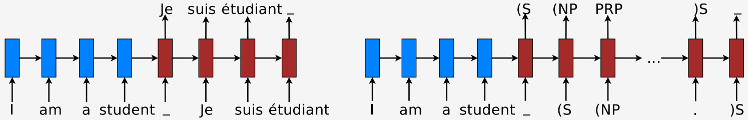


Firat and Cho: https://ufal.mff.cuni.cz/mtm16/files/12-recent-advances-and-future-of-neural-mt-orhat-firat.pdf

Training data: other tasks

Multi-task models [Luong et al., 2016]

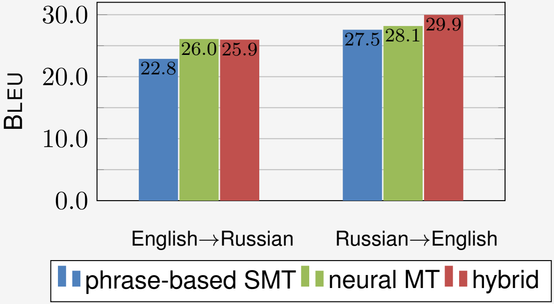
- other tasks can be modelled with sequence-to-sequence models
- we can share layers between translation and other tasks



NMT as a component in log-linear models

Log-linear models

- model ensembling is well-established
- reranking output of phrase-based/syntax-based with NMT [Neubig et al., 2015]
- incorporating NMT as a feature function into PBSMT [Junczys-Dowmunt et al., 2016]
  $\rightarrow$ results depend on relative performance of PBSMT and NMT



Linguistic Features [Sennrich and Haddow, 2016] a.k.a. Factored Neural Machine Translation

motivation: disambiguate words by POS

English	German
$\text{close}_{\text{verb}}$	schließen
$\text{close}_{\text{adj}}$	nah
$\text{close}_{\text{noun}}$	Ende

source	*We thought a win like this might be* $\textit{close}_{\textit{adj}}$.
reference	*Wir dachten, dass ein solcher Sieg nah sein könnte.*
baseline NMT	**Wir dachten, ein Sieg wie dieser könnte schließen.*

Linguistic Features: Architecture

use separate embeddings for each feature, then concatenate

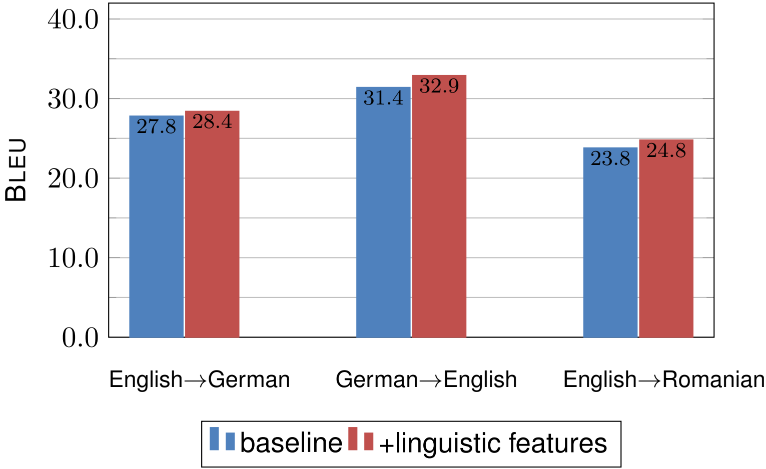
baseline: only word feature

$$E(close) = \begin{bmatrix} 0.5 \\ 0.2 \\ 0.3 \\ 0.1 \end{bmatrix}$$

$|F|$ input features

$$E_1(close) = \begin{bmatrix} 0.4 \\ 0.1 \\ 0.2 \end{bmatrix} \quad E_2(adj) = \begin{bmatrix} 0.1 \end{bmatrix} \quad E_1(close) \parallel E_2(adj) = \begin{bmatrix} 0.4 \\ 0.1 \\ 0.2 \\ 0.1 \end{bmatrix}$$

Linguistic Features: Results



Further Reading

secondary literature

- lecture notes by Kyunghyun Cho: [Cho, 2015]
- chapter on *Neural Network Models* in “Statistical Machine Translation” by Philipp Koehn `http://mt-class.org/jhu/assets/papers/neural-network-models.pdf`

(A small selection of) Resources

NMT tools

- dl4mt-tutorial (theano) https://github.com/nyu-dl/dl4mt-tutorial
  (our branch: nematus https://github.com/rsennrich/nematus)
- nmt.matlab https://github.com/lmthang/nmt.matlab
- seq2seq (tensorflow) https://www.tensorflow.org/versions/r0.8/tutorials/seq2seq/index.html
- neural monkey (tensorflow) https://github.com/ufal/neuralmonkey
- seq2seq-attn (torch) https://github.com/harvardnlp/seq2seq-attn

Do it yourself

- sample files and instructions for training NMT model
  `https://github.com/rsennrich/wmt16-scripts`
- pre-trained models to test decoding (and for further experiments)
  `http://statmt.org/rsennrich/wmt16_systems/`
- lab on installing/using Nematus:
  `http://ufal.mff.cuni.cz/mtm16/files/13-nematus-lab-rico-sennrich.pdf`

Bibliography I

Allen, R. B. (1987).
Several studies on natural language and back-propagation.
In In Proceedings of the IEEE First International Conference on Neural Networks, pages 335–341, San Diego, CA. IEEE.

Bahdanau, D., Cho, K., and Bengio, Y. (2015).
Neural Machine Translation by Jointly Learning to Align and Translate.
In Proceedings of the International Conference on Learning Representations (ICLR).

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. (2015).
Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks.
CoRR, abs/1506.03099.

Bentivogli, L., Bisazza, A., Cettolo, M., and Federico, M. (2016).
Neural versus Phrase-Based Machine Translation Quality: a Case Study.
In EMNLP 2016.

Cho, K. (2015).
Natural Language Understanding with Distributed Representation.
CoRR, abs/1511.07916.

Cho, K., Courville, A., and Bengio, Y. (2015).
Describing Multimedia Content using Attention-based Encoder-Decoder Networks.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014).
Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation.
In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Bibliography II

Chung, J., Cho, K., and Bengio, Y. (2016).
A Character-level Decoder without Explicit Segmentation for Neural Machine Translation.
CoRR, abs/1603.06147.

Costa-jussà, R. M. and Fonollosa, R. J. A. (2016).
Character-based Neural Machine Translation.
In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 357–361. Association for Computational Linguistics.

Dong, D., Wu, H., He, W., Yu, D., and Wang, H. (2015).
Multi-Task Learning for Multiple Language Translation.
In
Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference c
pages 1723–1732, Beijing, China. Association for Computational Linguistics.

Eriguchi, A., Hashimoto, K., and Tsuruoka, Y. (2016).
Tree-to-Sequence Attentional Neural Machine Translation.
In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 823–833. Association for Computational Linguistics.

Feng, S., Liu, S., Li, M., and Zhou, M. (2016).
Implicit Distortion and Fertility Models for Attention-based Encoder-Decoder NMT Model.
CoRR, abs/1601.03317.

Firat, O., Cho, K., and Bengio, Y. (2016a).
Multi-Way, Multilingual Neural Machine Translation with a Shared Attention Mechanism.
In
Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Langua
pages 866–875. Association for Computational Linguistics.

Bibliography III

Firat, O., Sankaran, B., Al-Onaizan, Y., Yarman-Vural, F. T., and Cho, K. (2016b).
Zero-Resource Translation with Multi-Lingual Neural Machine Translation.
CoRR, abs/1606.04164.

Gage, P. (1994).
A New Algorithm for Data Compression.
C Users J., 12(2):23–38.

Gülçehre, c., Firat, O., Xu, K., Cho, K., Barrault, L., Lin, H., Bougares, F., Schwenk, H., and Bengio, Y. (2015).
On Using Monolingual Corpora in Neural Machine Translation.
CoRR, abs/1503.03535.

Gulcehre, C., Ahn, S., Nallapati, R., Zhou, B., and Bengio, Y. (2016).
Pointing the Unknown Words.
In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 140–149. Association for Computational Linguistics.

Ha, T.-L., Niehues, J., Cho, E., Mediani, M., and Waibel, A. (2015).
The KIT translation systems for IWSLT 2015.
In Proceedings of the International Workshop on Spoken Language Translation (IWSLT), pages 62–69.

Haddow, B., Huck, M., Birch, A., Bogoychev, N., and Koehn, P. (2015).
The Edinburgh/JHU Phrase-based Machine Translation Systems for WMT 2015.
In Proceedings of the Tenth Workshop on Statistical Machine Translation, pages 126–133, Lisbon, Portugal. Association for Computational Linguistics.

Bibliography IV

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2015).
On Using Very Large Target Vocabulary for Neural Machine Translation.
In
Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference o
pages 1–10, Beijing, China. Association for Computational Linguistics.

Junczys-Dowmunt, M., Dwojak, T., and Sennrich, R. (2016).
The AMU-UEDIN Submission to the WMT16 News Translation Task: Attention-based NMT Models as Feature Functions in Phrase-based SMT.
In Proceedings of the First Conference on Machine Translation, Volume 2: Shared Task Papers, pages 316–322, Berlin, Germany. Association for Computational Linguistics.

Junczys-Dowmunt, M. and Grundkiewicz, R. (2016).
Log-linear Combinations of Monolingual and Bilingual Neural Machine Translation Models for Automatic Post-Editing.
In Proceedings of the First Conference on Machine Translation, pages 751–758, Berlin, Germany. Association for Computational Linguistics.

Kalchbrenner, N. and Blunsom, P. (2013).
Recurrent Continuous Translation Models.
In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle. Association for Computational Linguistics.

Kalchbrenner, N., Espeholt, L., Simonyan, K., van den Oord, A., Graves, A., and Kavukcuoglu, K. (2016).
Neural Machine Translation in Linear Time.
ArXiv e-prints.

Kim, Y. and Rush, A. M. (2016).
Sequence-Level Knowledge Distillation.
CoRR, abs/1606.07947.

Bibliography V

Lee, J., Cho, K., and Hofmann, T. (2016).
Fully Character-Level Neural Machine Translation without Explicit Segmentation.
ArXiv e-prints.

L'Hostis, G., Grangier, D., and Auli, M. (2016).
Vocabulary Selection Strategies for Neural Machine Translation.
ArXiv e-prints.

Ling, W., Trancoso, I., Dyer, C., and Black, A. W. (2015).
Character-based Neural Machine Translation.
ArXiv e-prints.

Liu, L., Utiyama, M., Finch, A., and Sumita, E. (2016).
Agreement on Target-bidirectional Neural Machine Translation .
In NAACL HLT 16, San Diego, CA.

Luong, M., Le, Q. V., Sutskever, I., Vinyals, O., and Kaiser, L. (2016).
Multi-task Sequence to Sequence Learning.
In ICLR 2016.

Luong, M.-T. and Manning, C. D. (2015).
Stanford Neural Machine Translation Systems for Spoken Language Domains.
In Proceedings of the International Workshop on Spoken Language Translation 2015, Da Nang, Vietnam.

Luong, M.-T. and Manning, D. C. (2016).
Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models.
In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1054–1063. Association for Computational Linguistics.

Bibliography VI

Luong, T., Pham, H., and Manning, C. D. (2015a).
Effective Approaches to Attention-based Neural Machine Translation.
In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Luong, T., Sutskever, I., Le, Q., Vinyals, O., and Zaremba, W. (2015b).
Addressing the Rare Word Problem in Neural Machine Translation.
In
Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference o
pages 11–19, Beijing, China. Association for Computational Linguistics.

Mi, H., Sankaran, B., Wang, Z., and Ittycheriah, A. (2016).
A Coverage Embedding Model for Neural Machine Translation.
ArXiv e-prints.

Mi, H., Wang, Z., and Ittycheriah, A. (2016).
Vocabulary Manipulation for Neural Machine Translation.
CoRR, abs/1605.03209.

Neubig, G., Morishita, M., and Nakamura, S. (2015).
Neural Reranking Improves Subjective Quality of Machine Translation: NAIST at WAT2015.
In Proceedings of the 2nd Workshop on Asian Translation (WAT2015), pages 35–41, Kyoto, Japan.

Pym, A., Grin, F., and Sfreddo, C. (2012).
The status of the translation profession in the European Union, volume 7.
European Commission, Luxemburg.

Bibliography VII

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2016).
Sequence Level Training with Recurrent Neural Networks.
In ICLR 2016.

Sennrich, R. and Haddow, B. (2016).
Linguistic Input Features Improve Neural Machine Translation.
In Proceedings of the First Conference on Machine Translation, Volume 1: Research Papers, pages 83–91, Berlin, Germany. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2016a).
Edinburgh Neural Machine Translation Systems for WMT 16.
In Proceedings of the First Conference on Machine Translation, Volume 2: Shared Task Papers, pages 368–373, Berlin, Germany. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2016b).
Improving Neural Machine Translation Models with Monolingual Data.
In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2016c).
Neural Machine Translation of Rare Words with Subword Units.
In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Shen, S., Cheng, Y., He, Z., He, W., Wu, H., Sun, M., and Liu, Y. (2016).
Minimum Risk Training for Neural Machine Translation.
In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Berlin, Germany.

Bibliography VIII

Sutskever, I., Vinyals, O., and Le, Q. V. (2014).
Sequence to Sequence Learning with Neural Networks.
In Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, pages 3104–3112, Montreal, Quebec, Canada.

Tu, Z., Liu, Y., Lu, Z., Liu, X., and Li, H. (2016).
Context Gates for Neural Machine Translation.
ArXiv e-prints.

Tu, Z., Lu, Z., Liu, Y., Liu, X., and Li, H. (2016a).
Coverage-based Neural Machine Translation.
CoRR, abs/1601.04811.

Tu, Z., Lu, Z., Liu, Y., Liu, X., and Li, H. (2016b).
Modeling Coverage for Neural Machine Translation.
In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 76–85. Association for Computational Linguistics.

Turovsky, B. (2016).
Ten years of Google Translate.
`https://googleblog.blogspot.co.uk/2016/04/ten-years-of-google-translate.html`.

Wiseman, S. and Rush, A. M. (2016).
Sequence-to-Sequence Learning as Beam-Search Optimization.
CoRR, abs/1606.02960.

Bibliography IX

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016).
Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation.
ArXiv e-prints.

Zhang, B., Xiong, D., and Su, J. (2016).
Recurrent Neural Machine Translation.
CoRR, abs/1607.08725.

Zhao, S. and Zhang, Z. (2016).
An Efficient Character-Level Neural Machine Translation.
ArXiv e-prints.

Zhou, J., Cao, Y., Wang, X., Li, P., and Xu, W. (2016).
Deep Recurrent Models with Fast-Forward Connections for Neural Machine Translation.
Transactions of the Association of Computational Linguistics – Volume 4, Issue 1, pages 371–383.

Zoph, B. and Knight, K. (2016).
Multi-Source Neural Translation.
In NAACL HLT 2016.