NEURONS

GD / SGD For Linear Neuron (Linear activation)

GD	SGD
$(\boldsymbol{X}, \boldsymbol{d})$	$(\boldsymbol{x}_p, d_p)$
$J = \frac{1}{2}\sum_{p=1}^{P}(d_p - y_p)^2$	$J = \frac{1}{2}(d_p - y_p)^2$
$\boldsymbol{y} = \boldsymbol{u} = \boldsymbol{X}\boldsymbol{w} + b\boldsymbol{1}_P$	$y_p = u_p = \boldsymbol{x}_p{}^T\boldsymbol{w} + b$
$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha\boldsymbol{X}^T(\boldsymbol{d} - \boldsymbol{y})$	$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha(d_p - y_p)\boldsymbol{x}_p$
$b \leftarrow b + \alpha\boldsymbol{1}_P{}^T(\boldsymbol{d} - \boldsymbol{y})$	$b \leftarrow b + \alpha(d_p - y_p)$

GD / SGD For Perceptron (Sigmoid activation)

GD	SGD
$(\boldsymbol{X}, \boldsymbol{d})$	$(\boldsymbol{x}_p, d_p)$
$J = \frac{1}{2}\sum_{p=1}^{P}(d_p - y_p)^2$	$J = \frac{1}{2}(d_p - y_p)^2$
$\boldsymbol{u} = \boldsymbol{X}\boldsymbol{w} + b\boldsymbol{1}_P$	$u_p = \boldsymbol{x}_p{}^T\boldsymbol{w} + b$
$\boldsymbol{y} = f(\boldsymbol{u})$	$y_p = f(u_p)$
$\boldsymbol{w} = \boldsymbol{w} + \alpha\boldsymbol{X}^T(\boldsymbol{d} - \boldsymbol{y}) \cdot f'(\boldsymbol{u})$	$\boldsymbol{w} = \boldsymbol{w} + \alpha(d_p - y_p)f'(u_p)\boldsymbol{x}_p$
$b = b + \alpha\boldsymbol{1}_P{}^T(\boldsymbol{d} - \boldsymbol{y}) \cdot f'(\boldsymbol{u})$	$b = b + \alpha(d_p - y_p)f'(u_p)$

GD / SGD For Logistic Regression Neuron (Sigmoid Activation)

GD	SGD
$(\boldsymbol{X}, \boldsymbol{d})$	$(\boldsymbol{x}_p, d_p)$
$J = -\sum_{p=1}^{P} d_p log\left(f(u_p)\right) + (1 - d_p)log\left(1 - f(u_p)\right)$	$J_p = -d_p log\left(f(u_p)\right) - (1 - d_p)log\left(1 - f(u_p)\right)$
$\boldsymbol{u} = \boldsymbol{X}\boldsymbol{w} + b\boldsymbol{1}_P$	$u_p = \boldsymbol{w}^T\boldsymbol{x}_p + b$
$f(\boldsymbol{u}) = \frac{1}{1 + e^{-\boldsymbol{u}}}$	$f(u_p) = \frac{1}{1 + e^{-u_p}}$
$\boldsymbol{y} = 1(f(\boldsymbol{u}) > 0.5)$	$y_p = 1(f(u_p) > 0.5)$
$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha\boldsymbol{X}^T(\boldsymbol{d} - f(\boldsymbol{u}))$	$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha\left(d_p - f(u_p)\right)\boldsymbol{x}_p$
$b \leftarrow b + \alpha\boldsymbol{1}_P{}^T(\boldsymbol{d} - f(\boldsymbol{u}))$	$b \leftarrow b + \alpha\left(d_p - f(u_p)\right)$

LAYERS

SGD / GD For a Single layer of neurons

Learning a layer of neurons	
SGD	$\boldsymbol{W} = \boldsymbol{W} - \alpha\boldsymbol{x}(\nabla_{\boldsymbol{u}}J)^T$ $\boldsymbol{b} = \boldsymbol{b} - \alpha\nabla_{\boldsymbol{u}}J$
GD	$\boldsymbol{W} = \boldsymbol{W} - \alpha\boldsymbol{X}^T\ \nabla_{\boldsymbol{U}}J$ $\boldsymbol{b} = \boldsymbol{b} - \alpha(\nabla_{\boldsymbol{U}}J)^T\boldsymbol{1}_P$

GD / SGD For a perceptron (sigmoid activation) layer - compare Perceptron!

GD	SGD
$(\boldsymbol{X}, \boldsymbol{D})$	$(\boldsymbol{x}, \boldsymbol{d})$
$\boldsymbol{U} = \boldsymbol{X}\boldsymbol{W} + \boldsymbol{B}$	$\boldsymbol{u} = \boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b}$
$\boldsymbol{Y} = f(\boldsymbol{U})$	$\boldsymbol{y} = f(\boldsymbol{u})$
$\nabla_{\boldsymbol{U}}J = -(\boldsymbol{D} - \boldsymbol{Y}) \cdot f'(\boldsymbol{U})$	$\nabla_{\boldsymbol{u}}J = -(\boldsymbol{d} - \boldsymbol{y}) \cdot f'(\boldsymbol{u})$
$\boldsymbol{W} = \boldsymbol{W} - \alpha\boldsymbol{X}^T\ \nabla_{\boldsymbol{U}}J$	$\boldsymbol{W} = \boldsymbol{W} - \alpha\boldsymbol{x}(\nabla_{\boldsymbol{u}}J)^T$
$\boldsymbol{b} = \boldsymbol{b} - \alpha(\nabla_{\boldsymbol{U}}J)^T\boldsymbol{1}_P$	$\boldsymbol{b} = \boldsymbol{b} - \alpha\nabla_{\boldsymbol{u}}J$

GD / SGD For Softmax Layer - Compare Logistic Regression (discrete output at end)

GD	SGD
$(\boldsymbol{X}, \boldsymbol{D})$	$(\boldsymbol{x}, d)$
$\boldsymbol{U} = \boldsymbol{X}\boldsymbol{W} + \boldsymbol{B}$	$\boldsymbol{u} = \boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b}$
$f(\boldsymbol{U}) = \frac{e^{\boldsymbol{U}}}{\sum_{k'=1}^{K} e^{\boldsymbol{U}_{k'}}}$	$f(\boldsymbol{u}) = \frac{e^{u_k}}{\sum_{k'=1}^{K} e^{u_{k'}}}$
$\boldsymbol{y} = \underset{k}{\text{argmax}}\, f(\boldsymbol{U})$	$y = \underset{k}{\text{argmax}}\, f(\boldsymbol{u})$
$\nabla_{\boldsymbol{U}}J = -(\boldsymbol{K} - f(\boldsymbol{U}))$	$\nabla_{\boldsymbol{u}}J = -(1(\boldsymbol{k} = d) - f(\boldsymbol{u}))$
$\boldsymbol{W} = \boldsymbol{W} - \alpha\boldsymbol{X}^T\ \nabla_{\boldsymbol{U}}J$	$\boldsymbol{W} = \boldsymbol{W} - \alpha\boldsymbol{x}(\nabla_{\boldsymbol{u}}J)^T$
$\boldsymbol{b} = \boldsymbol{b} - \alpha(\nabla_{\boldsymbol{U}}J)^T\boldsymbol{1}_P$	$\boldsymbol{b} = \boldsymbol{b} - \alpha\nabla_{\boldsymbol{u}}J$

$\frac{d}{du} sigmoid(u) = y(1 - y).\ \ y = \frac{1}{1+e^{-u}}$

$\frac{d}{du} tanh(u) = 1 - y^2,\ \ y = \frac{e^{u}-e^{-u}}{e^{u}+e^{-u}}$

$w_i \sim U(-a, a),\ \ a = gain \times \sqrt{\frac{6}{n_{in}+n_{out}}}$

$w_i \sim N(0, \frac{gain}{n}),\ \ n = num\ weights\ in\ neuron$

Gain scale depending on the neuron activation function

activation	linear	sigmoid	Tanh	ReLU	Leaky ReLU
gain	1	1	5/3	$\sqrt{2}$	$\sqrt{\frac{1}{1+slope^2}}$

GD / SGD in Two Layer FFN (two layers, V weights and c bias for second layer)

GD	SGD
$(\boldsymbol{X}, \boldsymbol{D})$	$(\boldsymbol{x}, \boldsymbol{d})$
$\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{W} + \boldsymbol{B}$	$\boldsymbol{z} = \boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b}$
$\boldsymbol{H} = g(\boldsymbol{Z})$	$\boldsymbol{h} = g(\boldsymbol{z})$
$\boldsymbol{U} = \boldsymbol{H}\boldsymbol{V} + \boldsymbol{C}$	$\boldsymbol{u} = \boldsymbol{V}^T\boldsymbol{h} + \boldsymbol{c}$
$\boldsymbol{Y} = f(\boldsymbol{U})$	$\boldsymbol{y} = f(\boldsymbol{u})$
$\nabla_{\boldsymbol{U}} J = \begin{cases} -(\boldsymbol{D} - \boldsymbol{Y}) \\ -(\boldsymbol{K} - f(\boldsymbol{U})) \end{cases}$	$\nabla_{\boldsymbol{u}} J = \begin{cases} -(\boldsymbol{d} - \boldsymbol{y}) \\ (1(\boldsymbol{k} = d) - f(\boldsymbol{u})) \end{cases}$
$\nabla_{\boldsymbol{Z}} J = (\nabla_{\boldsymbol{U}} J)\boldsymbol{V}^T \cdot g'(\boldsymbol{Z})$	$\nabla_{\boldsymbol{z}} J = \boldsymbol{V}\nabla_{\boldsymbol{u}} J \cdot g'(\boldsymbol{z})$
$\boldsymbol{W} \leftarrow \boldsymbol{W} - \alpha\boldsymbol{X}^T\ \nabla_{\boldsymbol{Z}} J$	$\boldsymbol{W} \leftarrow \boldsymbol{W} - \alpha\boldsymbol{x}(\nabla_{\boldsymbol{z}} J)^T$
$\boldsymbol{b} \leftarrow \boldsymbol{b} - \alpha(\nabla_{\boldsymbol{Z}} J)^T\boldsymbol{1}_P$	$\boldsymbol{b} \leftarrow \boldsymbol{b} - \alpha\nabla_{\boldsymbol{z}} J$
$\boldsymbol{V} \leftarrow \boldsymbol{V} - \alpha\boldsymbol{H}^T\ \nabla_{\boldsymbol{U}} J$	$\boldsymbol{V} \leftarrow \boldsymbol{V} - \alpha\boldsymbol{h}(\nabla_{\boldsymbol{u}} J)^T$
$\boldsymbol{c} \leftarrow \boldsymbol{c} - \alpha(\nabla_{\boldsymbol{U}} J)^T\boldsymbol{1}_P$	$\boldsymbol{c} \leftarrow \boldsymbol{c} - \alpha\nabla_{\boldsymbol{u}} J$

Adam - Combines RMSprop and moment methods

Suggested defaults alpha=0.001, rho1=0.9, rho2=0.999, eps=10^-8

Momentum term: $\boldsymbol{s} \leftarrow \rho_1\boldsymbol{s} + (1-\rho_1)\nabla_{\boldsymbol{W}} J$

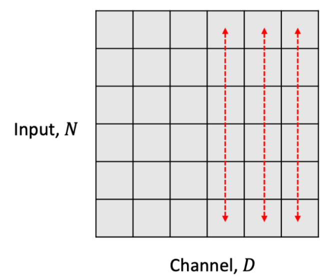
Learning rate term: $\boldsymbol{r} \leftarrow \rho_2\boldsymbol{r} + (1-\rho_2)(\nabla_{\boldsymbol{W}} J)^2$

$$\boldsymbol{s} \leftarrow \frac{\boldsymbol{s}}{1-\rho_1}$$

$$\boldsymbol{r} \leftarrow \frac{\boldsymbol{r}}{1-\rho_2}$$

$$\boldsymbol{W} \leftarrow \boldsymbol{W} - \frac{\alpha}{\varepsilon + \sqrt{\boldsymbol{r}}} \cdot \boldsymbol{s}$$

Batch Normalization



$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$ Per-channel mean, shape is $1\times D$

$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N} (x_{i,j} - \mu_j)^2$ Per-channel variance, shape is $1\times D$

$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$ Normalize the values, shape is $N\times D$

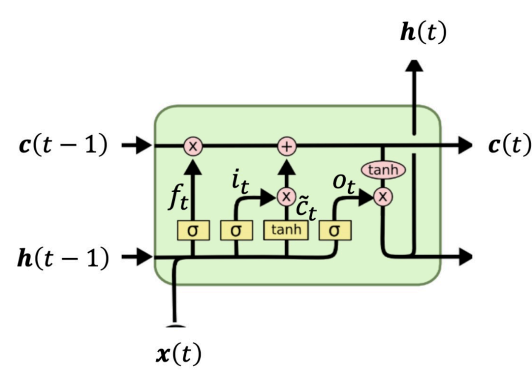
$y_{i,j} = \gamma_j\hat{x}_{i,j} + \beta_j$ **Scale and shift** the **normalized** values to add flexibility, output shape is $N\times D$

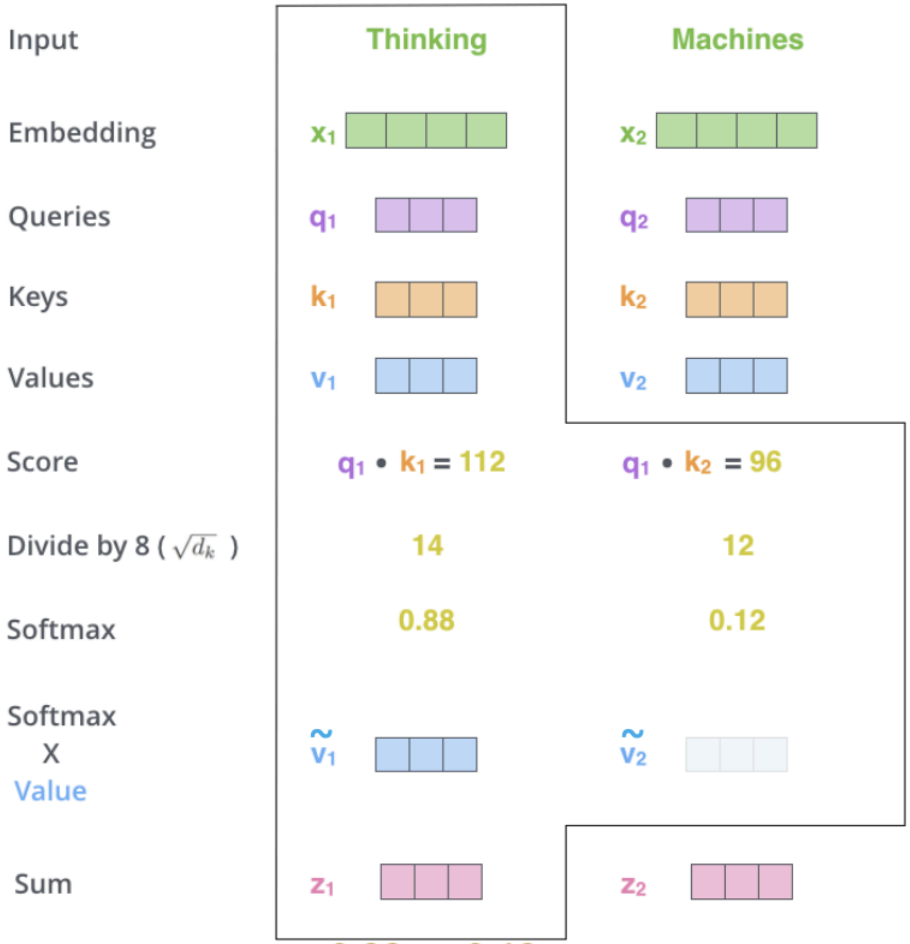
Learnable parameters: γ and β, shape is $1\times D$

LSTM Cell

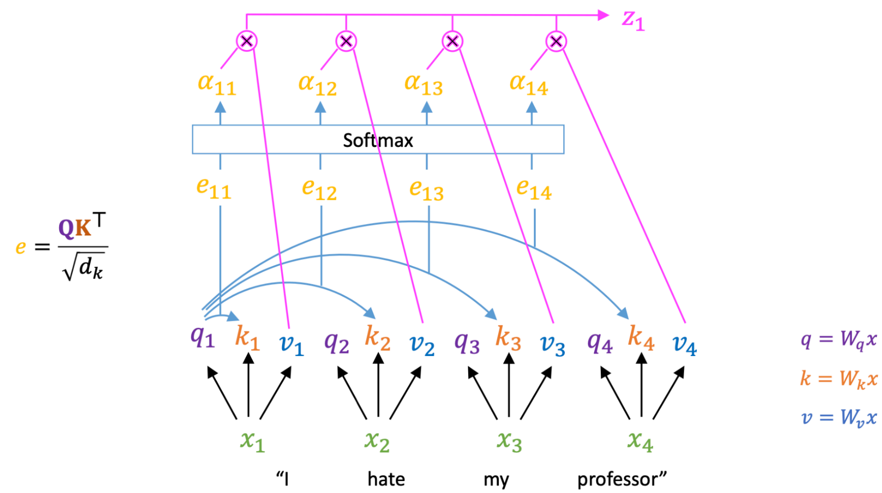
Key of LSTM is "state"

- A persistent module called the cell-state
- "State" is a representation of past history
- It comprises a common thread through time





$z_1 = 0.88v_1 + 0.12v_2$

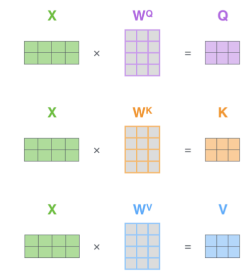


First Step

Calculate the Query, Key, and Value matrices.

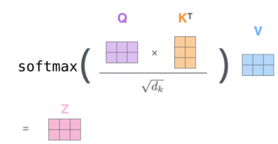
Pack our embeddings into a matrix X, and multiplying it by the weight matrices we've trained (W^Q, W^K, W^V)

Every row in the X matrix corresponds to a word in the input sentence.



Second Step

Calculate the outputs of the self-attention layer. SoftMax is row-wise



$$\mathrm{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{SoftMax}\left(\frac{\mathbf{QK}^{\mathsf{T}}}{\sqrt{d_k}}\right)\mathbf{V}$$