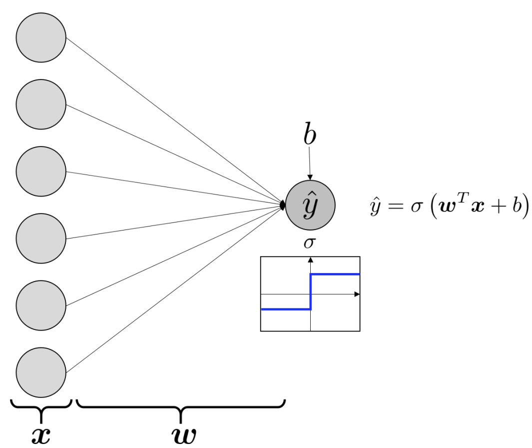
Introduction to Machine Learning
Lecture 9 - Feed Forward Networks

1 Introduction to (Feed-forward) Neural Nets

1.1 Introduction

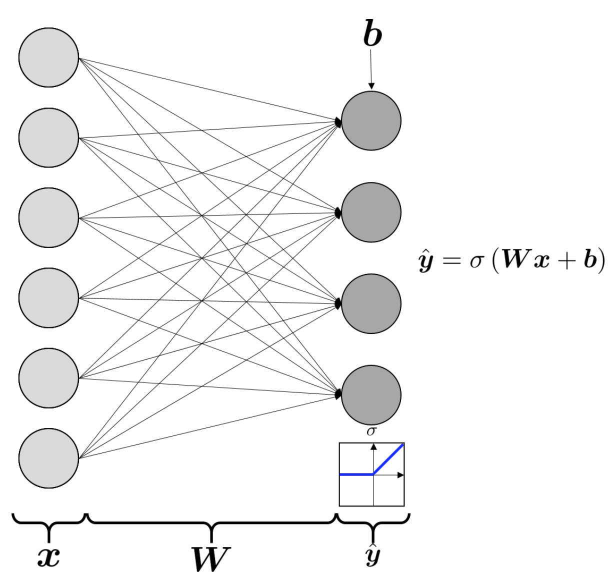
Reminder: Single perceptron



where σ is a non-linear activation function.
(For binary classification we set $\sigma(\cdot) = \text{sign}(\cdot)$).

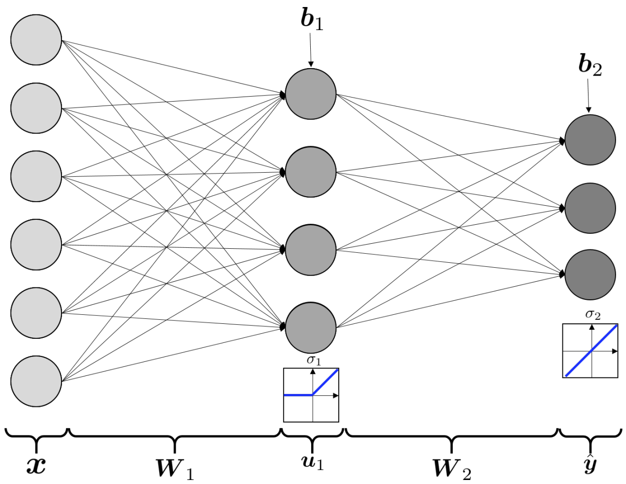
A single perceptron is a function from $\mathbb{R}^d \longrightarrow \mathbb{R}$ (where $\boldsymbol{x} \in \mathbb{R}^d$).

Multiple perceptron: The fully connected perceptron layer is a function from $\mathbb{R}^d \longrightarrow \mathbb{R}^{d_2}$ (where $\hat{\boldsymbol{y}} \in \mathbb{R}^{d_2}$):



Note that in this figure, σ is the ReLU activation $(\text{ReLU}(x) = \max\{0, x\})$

1 hidden layer (Fully connected) To obtain a (complex) non linear function we add additional (hidden) layer:



$$\begin{aligned}\hat{\boldsymbol{y}} &= \boldsymbol{W}_2\boldsymbol{u}_1 + \boldsymbol{b}_2\\ &= \boldsymbol{W}_2\sigma\left(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1\right) + \boldsymbol{b}_2\end{aligned}$$

Remarks:

1. By setting different values to $\{\boldsymbol{W}_i\}$ and $\{\boldsymbol{b}_i\}$ we can represent different functions.
2. There are several common non-linear activations: tanh, sigmoid, ReLU and more...
3. Notice that usually the activation in the output layer is dependent on the network task.
   For example a linear activation (the identity) for regression, or softmax activation for classification.
4. Deep feed-forward neural nets can have more then 1 hidden layer (much more).
5. There are more advanced layers: convolution, LSTM and more...

1.2 Representation example

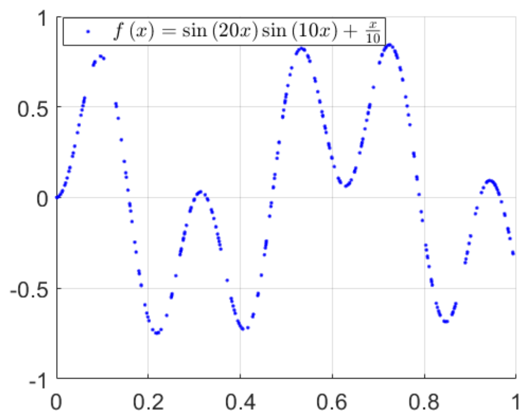
The following example shows the ability of a single hidden layer network to represent some continuous function. Consider the following function:

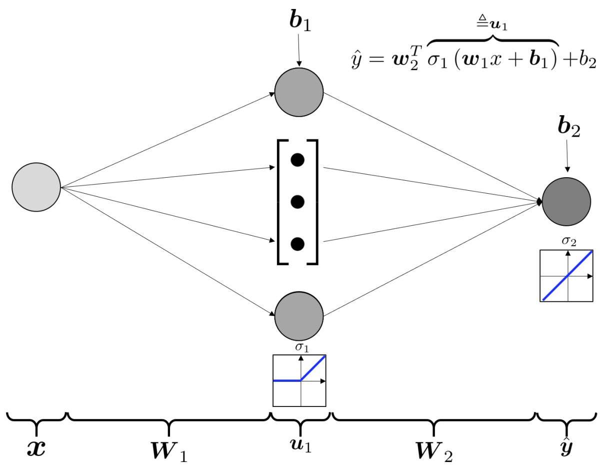
$$f(x) = \sin(20x)\sin(10x) + \frac{x}{10}$$

We generated a training set ($N = 300$):

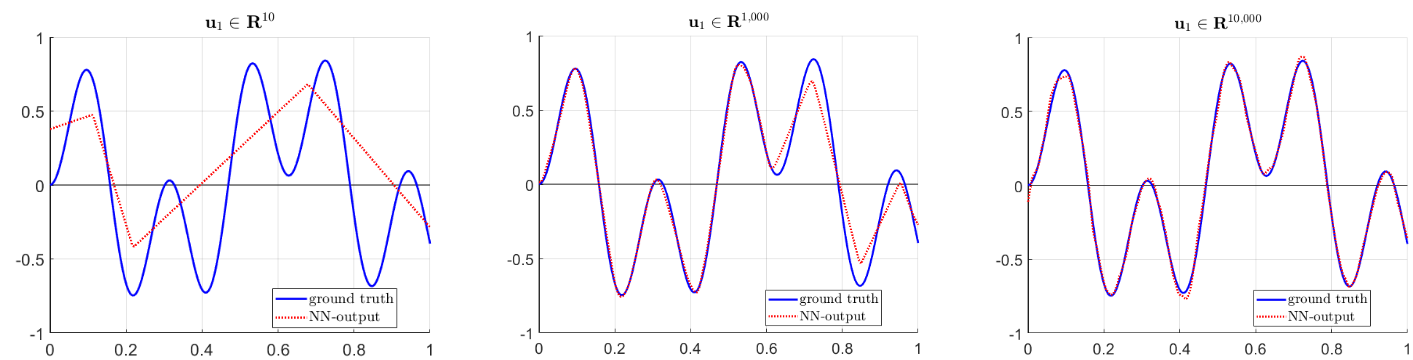
$$y_i = f(x_i), \qquad i \in \{1, 2, \ldots, N\}$$

Below are the training set (left) and a simple architecture $\hat{y} = \hat{f}(x)$ with one hidden layer (right).





In each experiment, we changed the size of the hidden layer:
(1) $\boldsymbol{u}_1 \in \mathbb{R}^{10}$, (2) $\boldsymbol{u}_1 \in \mathbb{R}^{1{,}000}$ and (3) $\boldsymbol{u}_1 \in \mathbb{R}^{10{,}000}$.
We approximate f with $\hat{f}$ by setting (training) the values of $\boldsymbol{w}_1, \boldsymbol{w}_2, b_1$ and b_2.



As one can see, the approximation ability is improving as the number of neurons increases.

Universal approximation theorem

Let σ be a non-constant, bounded, and monotonically-increasing continuous function.
Let $f : \mathbb{R}^d \to \mathbb{R}$ be a continuous on the d-dimensional unit hypercube $[0,1]^d$.
Then, given any $\varepsilon > 0$, there exist an integer D, constants vectors $\boldsymbol{b}, \boldsymbol{w}_2 \in \mathbb{R}^D$ and a matrix $\boldsymbol{W}_1 \in \mathbb{R}^{D \times d}$ such that we may define:

$$\hat{f}(\boldsymbol{x}) = \boldsymbol{w}_2^T \sigma(\boldsymbol{W}_1 \boldsymbol{x} + \boldsymbol{b})$$

as an approximate realization of the function f where f is independent of σ; that is,

$$\left| \hat{f}(\boldsymbol{x}) - f(\boldsymbol{x}) \right| < \varepsilon, \qquad \forall \boldsymbol{x} \in [0,1]^d$$

In other words, functions of the form $F(\boldsymbol{x})$ are dense in $C\left([0,1]^d\right)$.
This still holds when replacing $[0,1]^d$ with any compact subset of $\mathbb{R}^d$.

2 Training the network

We can train the network, that is, finding the weights $\{\boldsymbol{W}_i\}$ and biases $\{b_i\}$,
by using a training set $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$ of size N.
This can be done both for regression and classification tasks.

2.1 Loss functions

2.1.1 Regression

- L_2 loss (MSE):

$$L_2 = \frac{1}{2N} \sum_{i=1}^{N} \|\hat{\boldsymbol{y}}_i - \boldsymbol{y}\|_2^2$$

- L_1 loss:

$$L_1 = \frac{1}{N} \sum_{i=1}^{N} \|\hat{\boldsymbol{y}}_i - \boldsymbol{y}\|_1$$

In regression, the MSE loss function is a common choice.
However, other types of losses can be used (such as the L_1 loss: Mean Absolute Error (MAE)).

2.1.2 Classification

- One-hot encoding.
  In classification tasks, it is common to set the target vector $\boldsymbol{y}$ as a delta function (one-hot encoding):

$$\boldsymbol{y}_i = \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{bmatrix}^T \in \mathbb{R}^{|\mathcal{Y}|}$$

  where $\mathcal{Y}$ is the set of all classes, and the value 1 location is associate with the class number.
  For example, if $\boldsymbol{x}_i$ belongs to the second class, then:

$$\boldsymbol{y}_i = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \end{bmatrix}^T$$

- Softmax layer:

$$\boldsymbol{z} = \phi_{\text{softmax}}(\hat{\boldsymbol{y}}) = \frac{\exp(\hat{\boldsymbol{y}})}{\boldsymbol{1}^T \exp(\hat{\boldsymbol{y}})}$$

  The output vector $\boldsymbol{z}$, of the softmax function $\phi_{\text{softmax}} : \mathbb{R}^d \to \mathbb{R}^d$ is a probability vector.
  Namely, $\boldsymbol{z}[i] \geq 0$ and $\sum_i \boldsymbol{z}[i] = 1$.
  For example:

$$\hat{\boldsymbol{y}} = \begin{bmatrix} 3 \\ 1 \\ -1 \end{bmatrix} \Longrightarrow \boldsymbol{z} = \phi_{\text{softmax}}(\hat{\boldsymbol{y}}) = \begin{bmatrix} 0.87 \\ 0.12 \\ 0.01 \end{bmatrix}$$

- Cross entropy loss.
  In classification tasks, a common loss function is the cross-entropy (together with a softmax layer):

$$L = -\frac{1}{N} \sum_{i=1}^{N} \boldsymbol{y}_i^T \log(\boldsymbol{z}_i)$$

2.2 Training

We train the network by minimizing the loss function L:

$$\min_{\{\boldsymbol{W}_i\},\{b_i\}} L$$

This is done by optimization algorithms, such as the gradient descent (or some more advanced algorithms).
In most cases, the number of samples N is extremely large so stochastic optimization methods are used.

3 The Gradient and the Chain Rule

Consider a network with two hidden layers and activation functions ϕ_i:

$$\hat{\boldsymbol{y}} = \boldsymbol{W}_3\phi_2\left(\boldsymbol{W}_2\phi_1\left(\boldsymbol{W}_1\boldsymbol{x}+\boldsymbol{b}_1\right)+\boldsymbol{b}_2\right)+\boldsymbol{b}_3$$

we denote:

$$\begin{aligned}\boldsymbol{v}_1 &\triangleq \boldsymbol{W}_1\boldsymbol{x}+\boldsymbol{b}_1, \qquad \boldsymbol{u}_1 \triangleq \phi_1\left(\boldsymbol{v}_1\right)\\ \boldsymbol{v}_2 &\triangleq \boldsymbol{W}_2\boldsymbol{u}_1+\boldsymbol{b}_2, \qquad \boldsymbol{u}_2 \triangleq \phi_2\left(\boldsymbol{v}_2\right)\\ &\Rightarrow \hat{\boldsymbol{y}} = \boldsymbol{W}_3\boldsymbol{u}_2+\boldsymbol{b}_3\end{aligned}$$

Consider the MSE loss function (given some training set):

$$L = \frac{1}{2N}\sum_{i=1}^{N}\left\|\hat{\boldsymbol{y}}_i-\boldsymbol{y}\right\|_2^2$$

To apply gradient descent and update the weights $\{\boldsymbol{W}_i\}$ and $\{\boldsymbol{b}_i\}$,
we need to compute the gradient of L with respect to $\{\boldsymbol{W}_i\}$ and $\{b_i\}$.
For simplicity we will focus only on a single example ($N=1$):

$$L = \frac{1}{2}\left\|\hat{\boldsymbol{y}}-\boldsymbol{y}\right\|_2^2$$

3.1 Warm-up

Exercise 1 Find the gradient (with respect to $\boldsymbol{x}$) of:

$$f\left(\boldsymbol{x}\right) = \frac{1}{2}\left\|\boldsymbol{x}-\boldsymbol{a}\right\|_2^2$$
$$\nabla_x f = ?$$

Solution:

$$f = \frac{1}{2}\left(\boldsymbol{x}-\boldsymbol{a}\right)^T\left(\boldsymbol{x}-\boldsymbol{a}\right)$$

$$\Rightarrow \mathrm{d}f = \frac{1}{2}\left(\mathrm{d}\boldsymbol{x}^T\left(\boldsymbol{x}-\boldsymbol{a}\right)+\left(\boldsymbol{x}-\boldsymbol{a}\right)^T\mathrm{d}\boldsymbol{x}\right) = \frac{1}{2}\left(\left(\boldsymbol{x}-\boldsymbol{a}\right)^T\mathrm{d}\boldsymbol{x}+\left(\boldsymbol{x}-\boldsymbol{a}\right)^T\mathrm{d}\boldsymbol{x}\right) = \left(\boldsymbol{x}-\boldsymbol{a}\right)^T\mathrm{d}\boldsymbol{x}$$

$$\Rightarrow \boxed{\nabla_x f = \boldsymbol{x}-\boldsymbol{a}}$$

Exercise 2 Find the gradient of:

$$f\left(\boldsymbol{x}\right) = \begin{bmatrix}\phi\left(\boldsymbol{x}\left[1\right]\right)\\ \phi\left(\boldsymbol{x}\left[2\right]\right)\\ \vdots\\ \phi\left(\boldsymbol{x}\left[d\right]\right)\end{bmatrix}, \qquad \boldsymbol{x}\in\mathbb{R}^d$$

for some differential scalar function $\phi:\mathbb{R}\to\mathbb{R}$.
Solution:

$$\begin{aligned}f\left(\boldsymbol{x}\right) &= \begin{bmatrix}\phi\left(\boldsymbol{x}\left[1\right]\right)\\ \vdots\\ \phi\left(\boldsymbol{x}\left[d\right]\right)\end{bmatrix}\\ \Rightarrow \mathrm{d}f &= \begin{bmatrix}\mathrm{d}\phi\left(\boldsymbol{x}\left[1\right]\right)\\ \vdots\\ \mathrm{d}\phi\left(\boldsymbol{x}\left[d\right]\right)\end{bmatrix} = \begin{bmatrix}\phi'\left(\boldsymbol{x}\left[1\right]\right)\mathrm{d}\boldsymbol{x}\left[1\right]\\ \vdots\\ \phi'\left(\boldsymbol{x}\left[d\right]\right)\mathrm{d}\boldsymbol{x}\left[d\right]\end{bmatrix} = \underbrace{\begin{bmatrix}\phi'\left(\boldsymbol{x}\left[1\right]\right) & 0 & 0\\ 0 & \ddots & 0\\ 0 & 0 & \phi'\left(\boldsymbol{x}\left[d\right]\right)\end{bmatrix}}_{\triangleq\boldsymbol{\Phi}'\left(\boldsymbol{x}\right)}\mathrm{d}\boldsymbol{x}\end{aligned}$$

$$\Rightarrow \boxed{\nabla f = \left(\boldsymbol{\Phi}'\left(\boldsymbol{x}\right)\right)^T = \boldsymbol{\Phi}'\left(\boldsymbol{x}\right)}$$

3.2 Back-propagation (the chain rule)

The architecture is given by:

$$\hat{\boldsymbol{y}} = \boldsymbol{W}_3\phi_2\left(\boldsymbol{W}_2\phi_1\left(\boldsymbol{W}_1\boldsymbol{x}+\boldsymbol{b}_1\right)+\boldsymbol{b}_2\right)+\boldsymbol{b}_3$$

The MSE loss is given by:

$$L = \frac{1}{2}\left\|\hat{\boldsymbol{y}}-\boldsymbol{y}\right\|_2^2$$

3.2.1 Step I:

Note that:

$$\Rightarrow \boxed{\nabla_{\hat{y}}L = \hat{\boldsymbol{y}}-\boldsymbol{y}}$$

3.2.2 Step II:

Let us calculate $\nabla_{b_3}L$.
Using the chain rule we have:

$$\begin{aligned}\Rightarrow \mathrm{d}_{b_3}L &= \nabla_{\hat{y}}^T L\cdot \mathrm{d}_{b_3}\hat{\boldsymbol{y}}\\ &= \left(\hat{\boldsymbol{y}}-\boldsymbol{y}\right)^T\cdot \mathrm{d}_{b_3}\hat{\boldsymbol{y}}, \qquad \hat{\boldsymbol{y}} = \boldsymbol{W}_3\boldsymbol{u}_2+\boldsymbol{b}_3\\ &= \left(\hat{\boldsymbol{y}}-\boldsymbol{y}\right)^T \mathrm{d}\boldsymbol{b}_3\end{aligned}$$

$$\Rightarrow \boxed{\nabla_{b_3}L = \hat{\boldsymbol{y}}-\boldsymbol{y}}$$

3.2.3 Step III:

Let us calculate $\nabla_{b_2}L$.
Using the previous result, we have:

$$\begin{aligned}\Rightarrow \mathrm{d}_{b_2}L &= \nabla_{b_3}^T L\cdot \mathrm{d}_{b_2}\hat{\boldsymbol{y}}, \qquad \hat{\boldsymbol{y}} = \boldsymbol{W}_3\boldsymbol{u}_2+\boldsymbol{b}_3\\ &= \nabla_{b_3}^T L\cdot \boldsymbol{W}_3\mathrm{d}_{b_2}\boldsymbol{u}_2, \qquad \boldsymbol{u}_2 = \phi_2\left(\boldsymbol{v}_2\right)\\ &= \nabla_{b_3}^T L\cdot \boldsymbol{W}_3\boldsymbol{\Phi}_2'\left(\boldsymbol{u}_2\right)\mathrm{d}_{b_2}\boldsymbol{v}_2, \qquad \boldsymbol{v}_2 = \boldsymbol{W}_2\boldsymbol{u}_1+\boldsymbol{b}_2\\ &= \nabla_{b_3}^T L\cdot \boldsymbol{W}_3\boldsymbol{\Phi}_2'\left(\boldsymbol{u}_2\right)\mathrm{d}\boldsymbol{b}_2\end{aligned}$$

$$\Rightarrow \boxed{\nabla_{b_2}L = \boldsymbol{\Phi}_2'\left(\boldsymbol{u}_2\right)\boldsymbol{W}_3^T\nabla_{b_3}L} = \boldsymbol{\Phi}_2'\left(\boldsymbol{u}_2\right)\boldsymbol{W}_3^T\left(\hat{\boldsymbol{y}}-\boldsymbol{y}\right)$$

3.2.4 Step IV:

Let us calculate $\nabla_{b_1}L$.
Using the previous result, we have:

$$\begin{aligned}\Rightarrow \mathrm{d}_{b_1}L &= \nabla_{b_2}^T L\cdot \mathrm{d}_{b_1}\boldsymbol{v}_2, \qquad \boldsymbol{v}_2 = \boldsymbol{W}_2\boldsymbol{u}_1+\boldsymbol{b}_2\\ &= \nabla_{b_2}^T L\cdot \boldsymbol{W}_2\mathrm{d}_{b_1}\boldsymbol{u}_1, \qquad \boldsymbol{u}_1 = \phi_1\left(\boldsymbol{v}_1\right)\\ &= \nabla_{b_2}^T L\cdot \boldsymbol{W}_2\Phi_1'\left(\boldsymbol{u}_1\right)\mathrm{d}_{b_1}\boldsymbol{v}_1, \qquad \boldsymbol{v}_1 = \boldsymbol{v}_1 = \boldsymbol{W}_1\boldsymbol{x}+\boldsymbol{b}_1\\ &= \nabla_{b_2}^T L\cdot \boldsymbol{W}_2\Phi_1'\left(\boldsymbol{u}_1\right)\mathrm{d}_{b_1}\boldsymbol{b}_1\end{aligned}$$

$$\Rightarrow \boxed{\nabla_{b_1}L = \boldsymbol{\Phi}_1'\left(\boldsymbol{u}_1\right)\boldsymbol{W}_2^T\nabla_{b_2}L} = \boldsymbol{\Phi}_1'\left(\boldsymbol{u}_1\right)\boldsymbol{W}_2^T\boldsymbol{\Phi}_2'\left(\boldsymbol{u}_2\right)\boldsymbol{W}_3^T\left(\hat{\boldsymbol{y}}-\boldsymbol{y}\right)$$

Summary (derivatives of $\{\boldsymbol{b}_i\}$):

Derivative	Chain Rule
$\boldsymbol{g}_3 \triangleq \nabla_{b_3} L$	$\boldsymbol{g}_3 = \hat{\boldsymbol{y}} - \boldsymbol{y}$
$\boldsymbol{g}_2 \triangleq \nabla_{b_2} L$	$\boldsymbol{g}_2 = \boldsymbol{\Phi}_2' (\boldsymbol{u}_2) \boldsymbol{W}_3^T \boldsymbol{g}_3$
$\boldsymbol{g}_1 \triangleq \nabla_{b_1} L$	$\boldsymbol{g}_1 = \boldsymbol{\Phi}_1' (\boldsymbol{u}_1) \boldsymbol{W}_2^T \boldsymbol{g}_2$

For a general network with L layers,
we can compute the gradients $\boldsymbol{g}_i \triangleq \nabla_{b_i} L$ using the back-propagation rule by:

$$\begin{cases} \boldsymbol{g}_L = \hat{\boldsymbol{y}} - \boldsymbol{y} & i = L \\ \boldsymbol{g}_i = \boldsymbol{\Phi}_i' (\boldsymbol{u}_i) \boldsymbol{W}_{i+1}^T \boldsymbol{g}_{i+1} & i < L \end{cases}$$

In a vary similar way (see appendix) we obtain the derivatives of $\{\boldsymbol{W}_i\}$:

Derivative	Chain Rule	Notations
$\nabla_{W_3} L$	$\nabla_{W_3} L = \boldsymbol{g}_3 \boldsymbol{u}_2^T$	$\boldsymbol{g}_3 \triangleq \nabla_{b_3} L$
$\nabla_{W_2} L$	$\nabla_{W_2} L = \boldsymbol{g}_2 \boldsymbol{u}_1^T$	$\boldsymbol{g}_2 \triangleq \nabla_{b_2} L$
$\nabla_{W_1} L$	$\nabla_{W_1} L = \boldsymbol{g}_1 \boldsymbol{x}^T$	$\boldsymbol{g}_1 \triangleq \nabla_{b_1} L$

This process of calculating the gradient of each layer using the later layer's gradient is called back-propagation.

The gradient of the ReLU activation is simply:

$$\mathrm{ReLU}(x) = \begin{cases} x & x \geq 0 \\ 0 & \text{else} \end{cases}$$

$$\Rightarrow \frac{\mathrm{d}}{\mathrm{d}x} \mathrm{ReLU}(x) = \begin{cases} 1 & x \geq 0 \\ 0 & \text{else} \end{cases}$$

For other activation functions, one just needs to compute their respective derivative.

4 Softmax and Cross-entropy Loss:

- The softmax layer:

$$\boldsymbol{z} \triangleq \frac{\exp(\hat{\boldsymbol{y}})}{\mathbf{1}^T \exp(\hat{\boldsymbol{y}})}$$

- Cross entropy loss:

$$\begin{aligned} L &= -\boldsymbol{y}^T \log(\boldsymbol{z}) \\ &= -\boldsymbol{y}^T \log\left(\frac{\exp(\hat{\boldsymbol{y}})}{\mathbf{1}^T \exp(\hat{\boldsymbol{y}})}\right) \\ &= -\boldsymbol{y}^T \left(\hat{\boldsymbol{y}} - \mathbf{1} \cdot \log\left(\mathbf{1}^T \exp(\hat{\boldsymbol{y}})\right)\right) \\ &= -\boldsymbol{y}^T \hat{\boldsymbol{y}} + \boldsymbol{y}^T \mathbf{1} \log\left(\mathbf{1}^T \exp(\hat{\boldsymbol{y}})\right) \end{aligned}$$

(Note that for classification tasks we can assume: $\boldsymbol{y}^T \mathbf{1} = 1$):

$$\Rightarrow L = -\boldsymbol{y}^T \hat{\boldsymbol{y}} + \log\left(\mathbf{1}^T \exp(\hat{\boldsymbol{y}})\right)$$

- Gradient with respect to $\hat{\boldsymbol{y}}$:

$$\begin{aligned} \mathrm{d}_{\hat{\boldsymbol{y}}} L &= -\boldsymbol{y}^T \mathrm{d}\hat{\boldsymbol{y}} + \frac{1}{\left(\mathbf{1}^T \exp(\hat{\boldsymbol{y}})\right)} \exp\left(\hat{\boldsymbol{y}}^T\right) \mathrm{d}\hat{\boldsymbol{y}} \\ &= \left(\frac{\exp\left(\hat{\boldsymbol{y}}^T\right)}{\left(\mathbf{1}^T \exp(\hat{\boldsymbol{y}})\right)} - \boldsymbol{y}^T\right) \mathrm{d}\hat{\boldsymbol{y}} \\ &= \left(\boldsymbol{z}^T - \boldsymbol{y}^T\right) \mathrm{d}\hat{\boldsymbol{y}} \end{aligned}$$

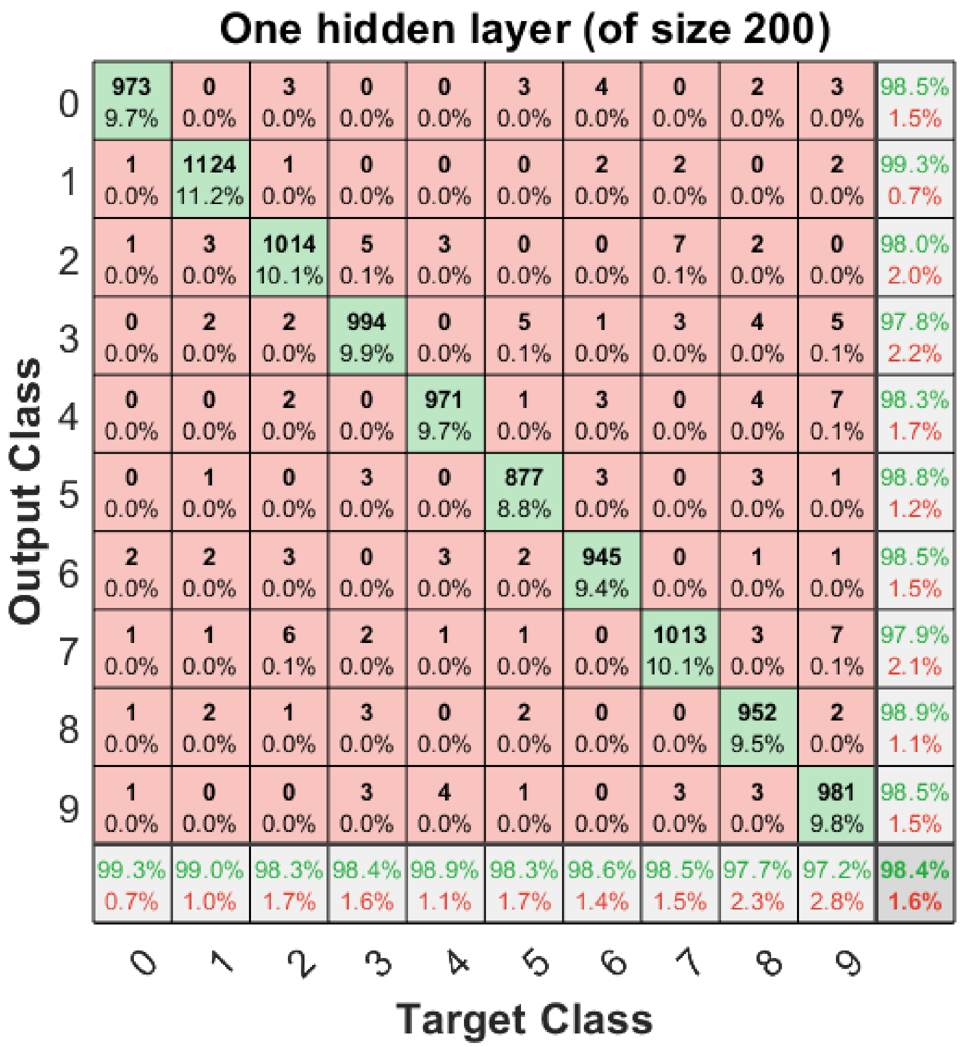
$$\Rightarrow \boxed{\nabla_{\hat{\boldsymbol{y}}} L = \boldsymbol{z} - \boldsymbol{y}}$$

In the general case where $\boldsymbol{y}^T \mathbf{1} \neq 1$ we have:

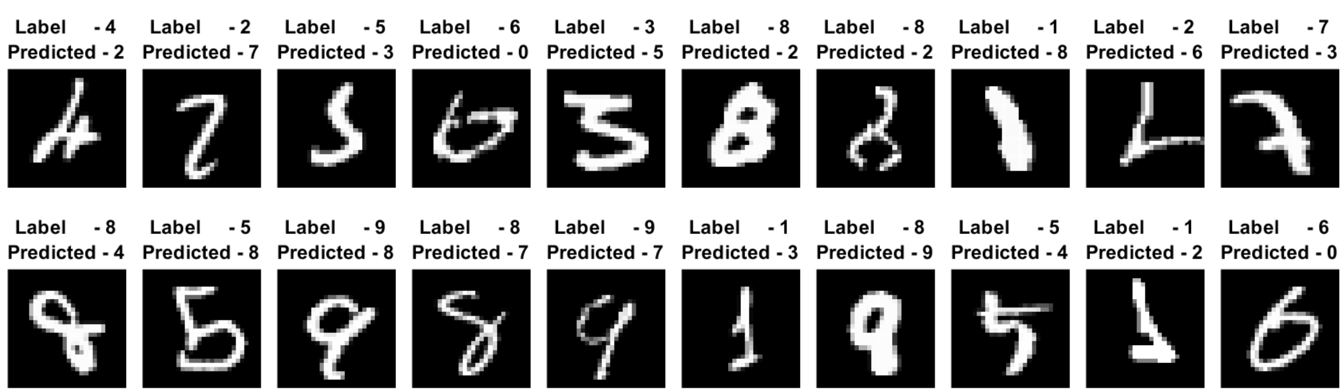
$$\nabla_{\hat{\boldsymbol{y}}} L = \left(\boldsymbol{y}^T \mathbf{1}\right) \boldsymbol{z} - \boldsymbol{y}$$

5 MNIST Example

We trained a NN with one hidden layer architecture ($\boldsymbol{u}_1 \in \mathbb{R}^{200}$) on the MNIST training set.
This is the classification result on the test set:

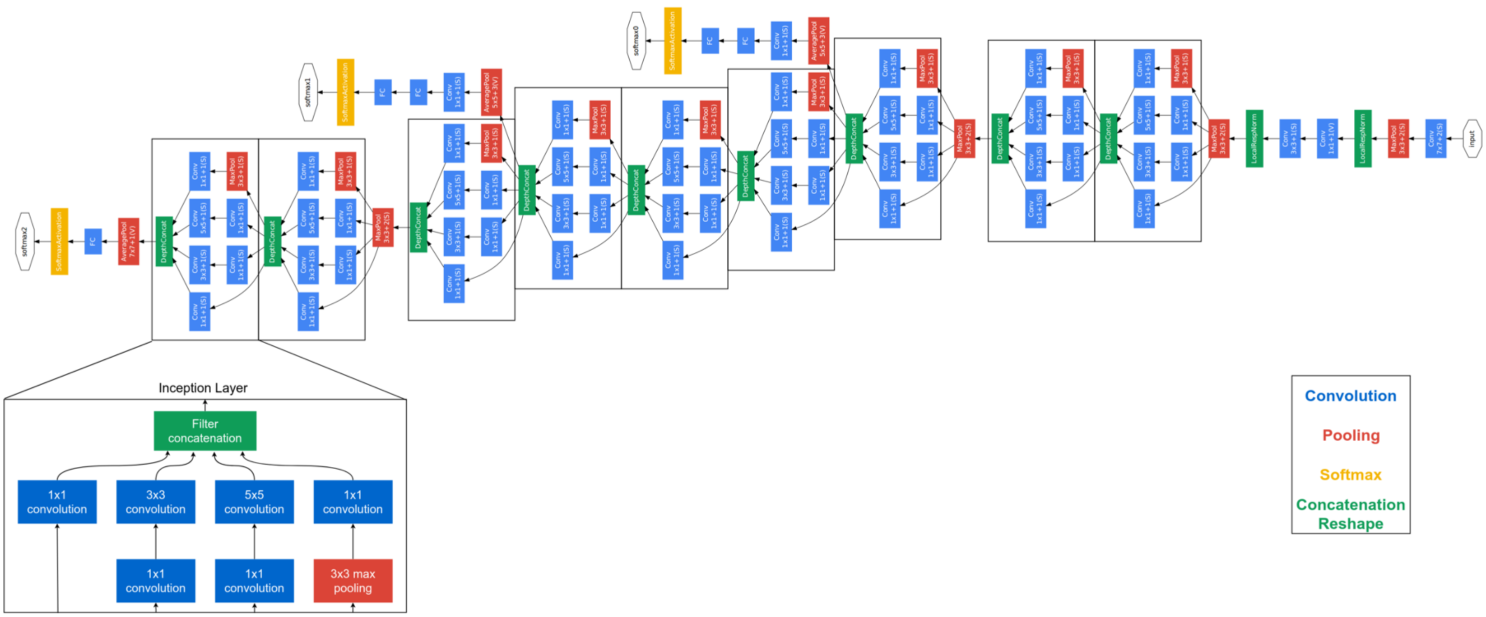


Some errors:



6 Deep Networks

In deep networks, there are usually millions of parameters to optimize.
For example, the GoogLeNet network:



Tasks Deep learning can be useful in many practical task such as:

- Face detection + recognition

- Gray to Color

Colorado National Park, 1941 Textile Mill, June 1937 Berry Field, June 1909 Hamilton, 1936

- Text translation
- Language recognition
- Autonomous vehicles
- Deep-learning robots
- and much much more

7 Appendix

7.1 The gradient with respect to $\boldsymbol{W}_i$

$$\Rightarrow \hat{\boldsymbol{y}} = \phi_3\left(\boldsymbol{W}_3\left(\phi_2\left(\boldsymbol{W}_2\left(\phi_1\left(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1\right)\right) + \boldsymbol{b}_2\right)\right) + \boldsymbol{b}_3\right)$$

$$L = \frac{1}{2}\left\|\hat{\boldsymbol{y}} - \boldsymbol{y}\right\|_2^2$$

$$\begin{aligned}
\mathrm{d}\hat{\boldsymbol{y}} = \mathrm{d}\boldsymbol{u_3} \qquad &= \mathrm{d}\phi_3\left(\boldsymbol{v}_3\right) \\
&= \boldsymbol{\Phi}_3'\left(\boldsymbol{v}_3\right)\mathrm{d}\boldsymbol{v}_3 \\
&= \boldsymbol{\Phi}_3'\left(\boldsymbol{v}_3\right)\boldsymbol{W}_3\mathrm{d}\boldsymbol{u}_2 \\
&= \boldsymbol{\Phi}_3'\left(\boldsymbol{v}_3\right)\boldsymbol{W}_3\boldsymbol{\Phi}_2'\left(\boldsymbol{v}_2\right)\mathrm{d}\boldsymbol{v}_2 \\
&= \boldsymbol{\Phi}_3'\left(\boldsymbol{v}_3\right)\boldsymbol{W}_3\boldsymbol{\Phi}_2'\left(\boldsymbol{v}_2\right)\boldsymbol{W}_2\mathrm{d}\boldsymbol{u}_2 \\
&= \boldsymbol{\Phi}_3'\left(\boldsymbol{v}_3\right)\boldsymbol{W}_3\boldsymbol{\Phi}_2'\left(\boldsymbol{v}_2\right)\boldsymbol{W}_2\boldsymbol{\Phi}_1'\left(\boldsymbol{v}_1\right)\mathrm{d}\boldsymbol{v}_1 \\
&= \boldsymbol{\Phi}_3'\left(\boldsymbol{v}_3\right)\boldsymbol{W}_3\boldsymbol{\Phi}_2'\left(\boldsymbol{v}_2\right)\boldsymbol{W}_2\boldsymbol{\Phi}_1'\left(\boldsymbol{v}_1\right)\mathrm{d}\boldsymbol{W}_1\boldsymbol{x}
\end{aligned}$$

$$\begin{aligned}
\mathrm{d}_{W_3}L &= \left(\hat{\boldsymbol{y}} - \boldsymbol{y}\right)^T\mathrm{d}_{W_3}\hat{\boldsymbol{y}} \\
&= \left(\hat{\boldsymbol{y}} - \boldsymbol{y}\right)^T\boldsymbol{\Phi}_3'\left(\boldsymbol{v}_3\right)\mathrm{d}\boldsymbol{W}_3\boldsymbol{u}_2 \\
&= \mathrm{Tr}\left\{\left(\hat{\boldsymbol{y}} - \boldsymbol{y}\right)^T\boldsymbol{\Phi}_3'\left(\boldsymbol{v}_3\right)\mathrm{d}\boldsymbol{W}_3\boldsymbol{u}_2\right\} \\
&= \mathrm{Tr}\left\{\boldsymbol{u}_2\left(\hat{\boldsymbol{y}} - \boldsymbol{y}\right)^T\boldsymbol{\Phi}_3'\left(\boldsymbol{v}_3\right)\mathrm{d}\boldsymbol{W}_3\right\}
\end{aligned}$$

$$\begin{aligned}
\mathrm{d}_{W_2}L &= \left(\hat{\boldsymbol{y}} - \boldsymbol{y}\right)^T\mathrm{d}_{W_2}\hat{\boldsymbol{y}} \\
&= \left(\hat{\boldsymbol{y}} - \boldsymbol{y}\right)^T\boldsymbol{\Phi}_3'\left(\boldsymbol{v}_3\right)\boldsymbol{W}_3\boldsymbol{\Phi}_2'\left(\boldsymbol{v}_2\right)\mathrm{d}\boldsymbol{W}_2\boldsymbol{u}_1 \\
&= \mathrm{Tr}\left\{\left(\hat{\boldsymbol{y}} - \boldsymbol{y}\right)^T\boldsymbol{\Phi}_3'\left(\boldsymbol{v}_3\right)\boldsymbol{W}_3\boldsymbol{\Phi}_2'\left(\boldsymbol{v}_2\right)\mathrm{d}\boldsymbol{W}_2\boldsymbol{u}_1\right\} \\
&= \mathrm{Tr}\left\{\boldsymbol{u}_1\left(\hat{\boldsymbol{y}} - \boldsymbol{y}\right)^T\boldsymbol{\Phi}_3'\left(\boldsymbol{v}_3\right)\boldsymbol{W}_3\boldsymbol{\Phi}_2'\left(\boldsymbol{v}_2\right)\mathrm{d}\boldsymbol{W}_2\right\}
\end{aligned}$$

$$\boxed{\nabla_{b_3}L = \boldsymbol{\Phi}_3'\left(\boldsymbol{v}_3\right)\left(\hat{\boldsymbol{y}} - \boldsymbol{y}\right)} = \boldsymbol{g}_3$$

$$\boxed{\nabla_{b_2}L = \boldsymbol{\Phi}_2'\left(\boldsymbol{v}_2\right)\boldsymbol{W}_3^T\boldsymbol{\Phi}_3'\left(\boldsymbol{v}_3\right)\left(\hat{\boldsymbol{y}} - \boldsymbol{y}\right)} = \boldsymbol{\Phi}_2'\left(\boldsymbol{v}_2\right)\boldsymbol{W}_3^T\boldsymbol{g}_3 = \boldsymbol{g}_2$$

$$\boxed{\nabla_{b_1}L = \boldsymbol{\Phi}_1'\left(\boldsymbol{v}_1\right)\boldsymbol{W}_2^T\boldsymbol{\Phi}_2'\left(\boldsymbol{v}_2\right)\boldsymbol{W}_3^T\boldsymbol{\Phi}_3'\left(\boldsymbol{v}_3\right)\left(\hat{\boldsymbol{y}} - \boldsymbol{y}\right)} = \boldsymbol{\Phi}_1'\left(\boldsymbol{v}_1\right)\boldsymbol{W}_2^T\boldsymbol{g}_2 = \boldsymbol{g}_1$$

$$\Rightarrow \boxed{\nabla_{W_3}L = \boldsymbol{\Phi}_3'\left(\boldsymbol{v}_3\right)\left(\hat{\boldsymbol{y}} - \boldsymbol{y}\right)\boldsymbol{u}_2^T} = \boldsymbol{g}_3\boldsymbol{u}_2^T$$

$$\Rightarrow \boxed{\nabla_{W_2}L = \boldsymbol{\Phi}_2'\left(\boldsymbol{v}_2\right)\boldsymbol{W}_3^T\boldsymbol{\Phi}_3'\left(\boldsymbol{v}_3\right)\left(\hat{\boldsymbol{y}} - \boldsymbol{y}\right)\boldsymbol{u}_1^T} = \boldsymbol{\Phi}_2'\left(\boldsymbol{v}_2\right)\boldsymbol{W}_3^T\boldsymbol{g}_3\boldsymbol{u}_1^T = \boldsymbol{g}_2\boldsymbol{u}_1^T$$

$$\Rightarrow \boxed{\nabla_{W_1}L = \boldsymbol{\Phi}_1'\left(\boldsymbol{v}_1\right)\boldsymbol{W}_2^T\boldsymbol{\Phi}_2'\left(\boldsymbol{v}_2\right)\boldsymbol{W}_3^T\boldsymbol{\Phi}_3'\left(\boldsymbol{v}_3\right)\left(\hat{\boldsymbol{y}} - \boldsymbol{y}\right)\boldsymbol{x}^T} = \boldsymbol{\Phi}_1'\left(\boldsymbol{v}_1\right)\boldsymbol{W}_2^T\boldsymbol{g}_2\boldsymbol{x}^T = \boldsymbol{g}_1\boldsymbol{x}^T$$