IFT 4031/7031,
Machine Learning for Signal Processing
Week 7: Deep Learning Primer

Cem Subakan

Admin

- Homework 1 due soon.
    - Devoir 1 est du bientot.
- The project proposals are due soon also!
    - Les propositions de projets sont dus ce soir!
- VALERIA is available now. How is it going?
    - Comment ca va avec VALERIA?

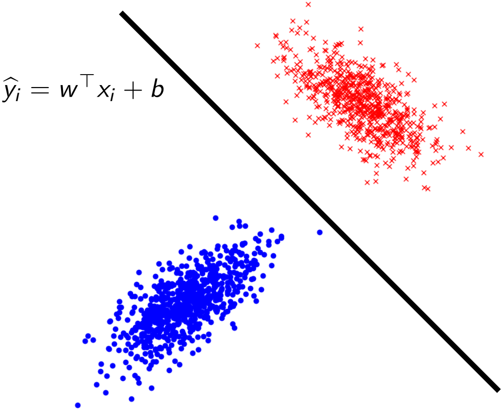
Today

- Neural Networks / Réseaux de Neurones
  - We will talk more about why / On parler davantage sur le ‘pourquoi’.
- Deep Learning
  - We will talk about how / On va parler sur ‘comment’.

Table of Contents

Linear Classifier Again

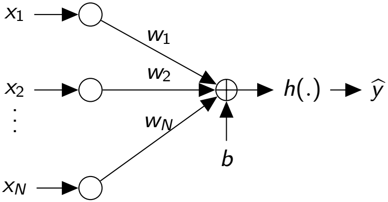
- Linear classifier / Classificateur Linéaire



$\widehat{y}_i = w^\top x_i + b$

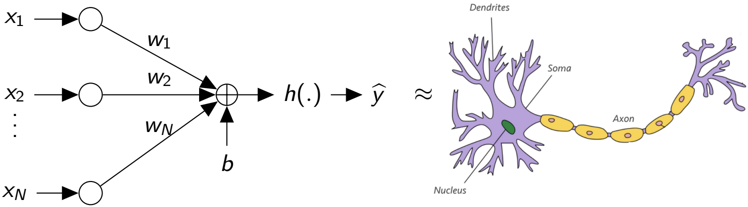
Why ‘Neural’ Network

- $\widehat{y} = w^\top x + b$



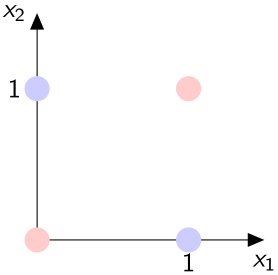
Why 'Neural' Network

- $\widehat{y} = w^\top x + b$



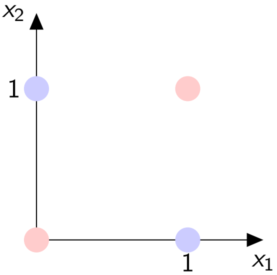
The XOR problem

x_1	x_2	XOR
0	0	0
1	0	1
0	1	1
1	1	0



The XOR problem

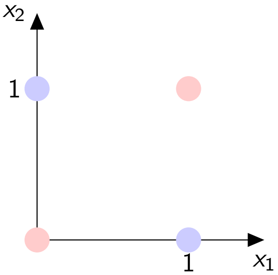
x_1	x_2	XOR
0	0	0
1	0	1
0	1	1
1	1	0



- Can we linearly separate this? / Peut-on séparer ça linéairement?

The XOR problem

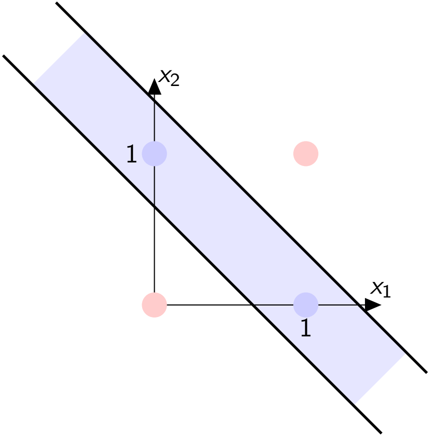
x_1	x_2	XOR
0	0	0
1	0	1
0	1	1
1	1	0



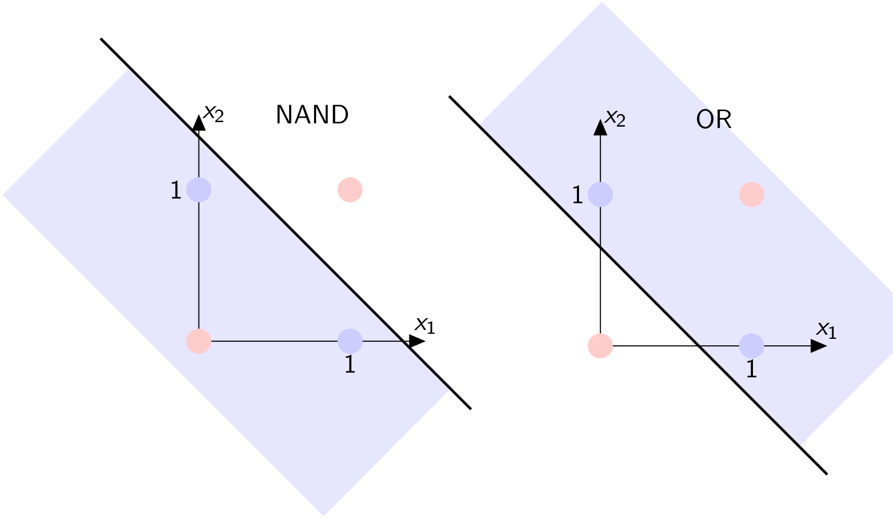
- Can we linearly separate this? / Peut-on séparer ça linéairement?
  - **No!** / **Non!**

An ideal solution

How do we do this though? /
Comment peut-on faire ça?

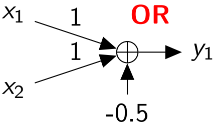


Combine!

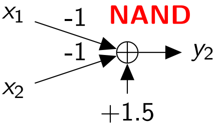


Two parallel perceptrons

x_1	x_2	y_1
0	0	-0.5
1	0	0.5
0	1	0.5
1	1	1.5



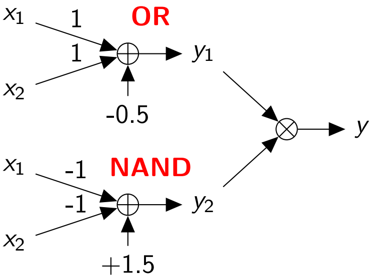
x_1	x_2	y_2
0	0	1.5
1	0	0.5
0	1	0.5
1	1	-0.5



Two parallel perceptrons

x_1	x_2	y_1
0	0	-0.5
1	0	0.5
0	1	0.5
1	1	1.5

x_1	x_2	y_2
0	0	1.5
1	0	0.5
0	1	0.5
1	1	-0.5

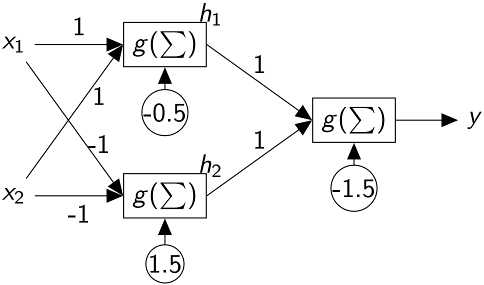


x_1	x_2	y
0	0	-0.75
1	0	0.25
0	1	0.25
1	1	-0.75

Or, everything with a single network

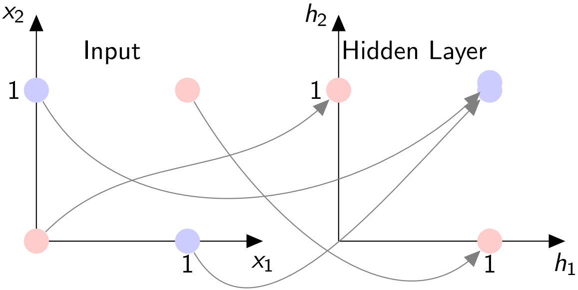
x_1	x_2	h_1
0	0	$u(-0.5) = 0$
1	0	$u(0.5) = 1$
0	1	$u(0.5) = 1$
1	1	$u(1.5) = 1$

x_1	x_2	h_2
0	0	$u(1.5) = 1$
1	0	$u(0.5) = 1$
0	1	$u(0.5) = 1$
1	1	$u(-0.5) = 0$

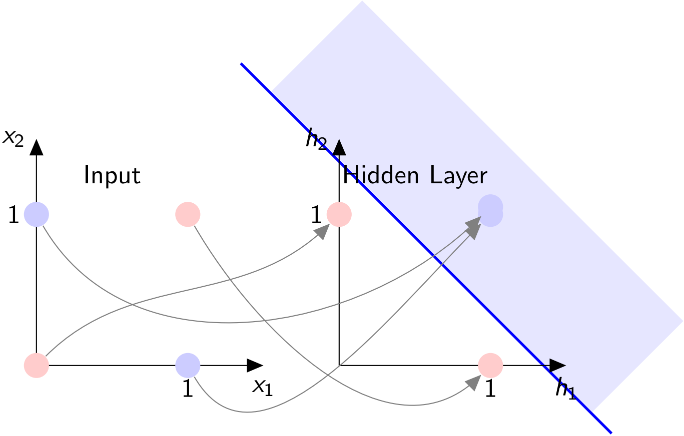


x_1	x_2	h_1	h_2	y
0	0	0	1	$u(-0.5) = 0$
1	0	1	1	$u(0.5) = 1$
0	1	1	1	$u(0.5) = 1$
1	1	1	0	$u(-0.5) = 0$

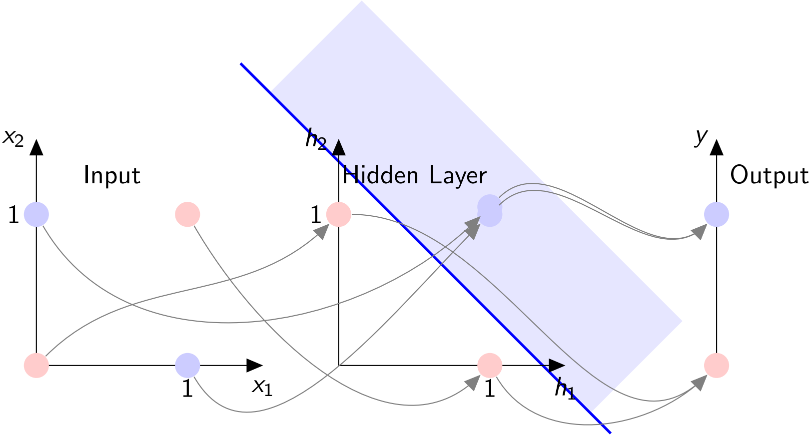
What is each layer doing?



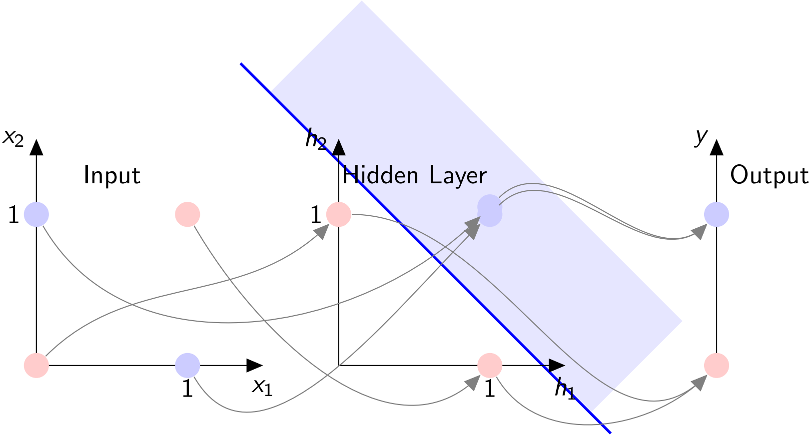
What is each layer doing?



What is each layer doing?

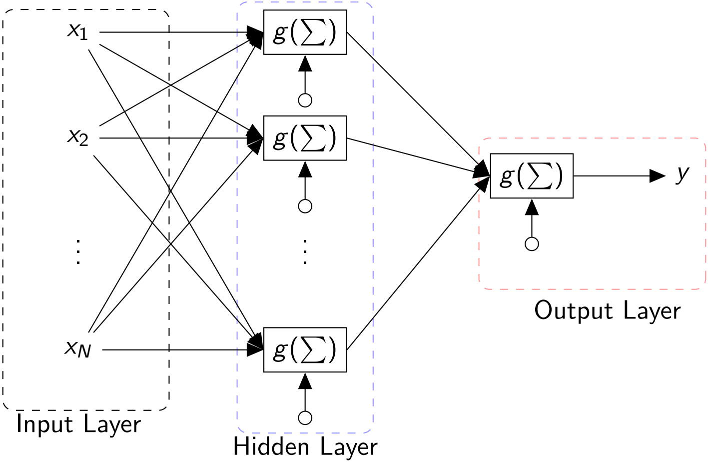


What is each layer doing?



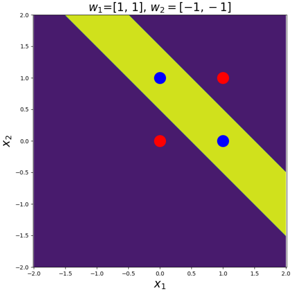
- First layer makes things linearly separable.
    - Premiere couche rend les choses sépérable
- Output layer finishes the job.
    - La couche de sortie finit le job.

Two layer feed-forward neural network

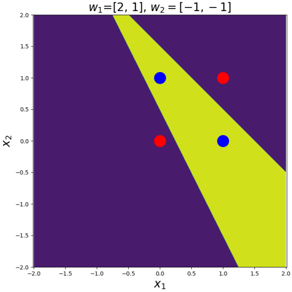


The hidden layer and decision boundary

- The decision boundary implied by network we have talked about
  - La borne de décision impliqué par le réseau dont on a parlé



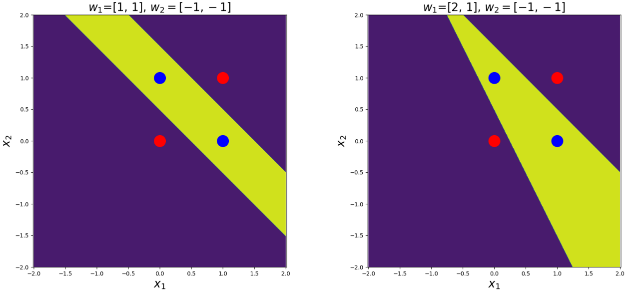
w_1=[1, 1], $w_2 = [-1, -1]$



w_1=[2, 1], $w_2 = [-1, -1]$

The hidden layer and decision boundary

- The decision boundary implied by network we have talked about
  - La borne de décision impliqué par le réseau dont on a parlé



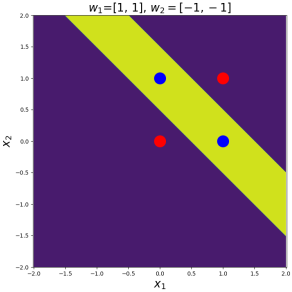
w_1=[1, 1], $w_2 = [-1, -1]$

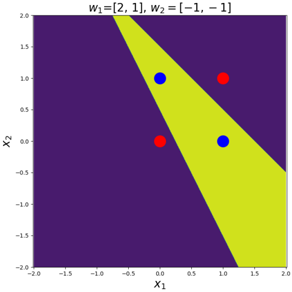
w_1=[2, 1], $w_2 = [-1, -1]$

- So, each hidden unit define decision boundary.
  - Chaque unité caché définit une borne de décision.

The hidden layer and decision boundary

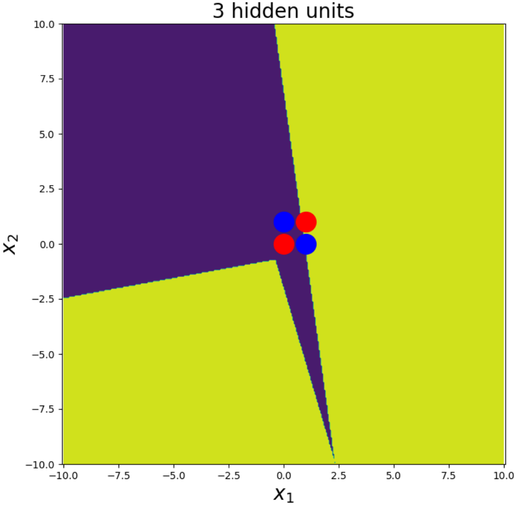
- The decision boundary implied by network we have talked about
  - La borne de décision impliqué par le réseau dont on a parlé





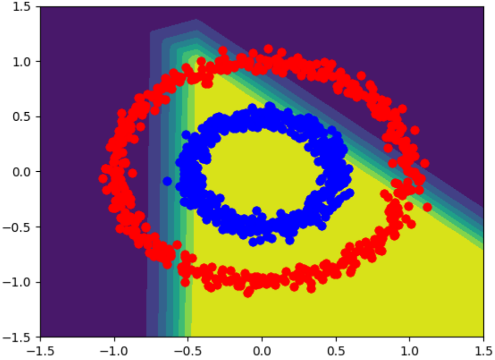
- So, each hidden unit define decision boundary.
  - Chaque unité caché définit une borne de décision.
- What if we have more than one hidden unit?
  - Et si on a plus qu'une seule unité caché?

Three hidden units



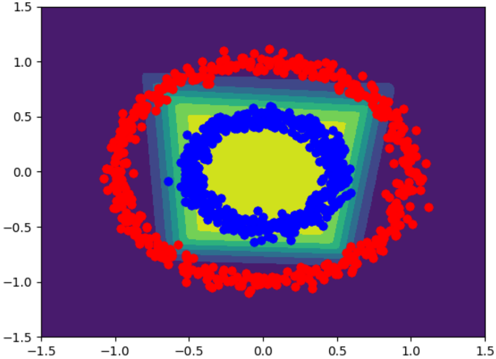
Classifying Circles

- With 2 Hidden Units / Avec 2 Unités Cachées



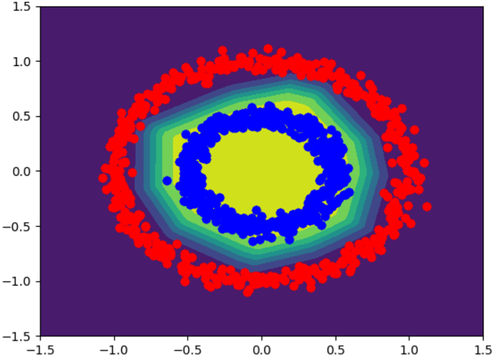
Classifying Circles

- With 3 Hidden Units / Avec 3 Unités Cachées



Classifying Circles

- With 10 Hidden Units / Avec 10 Unités Cachées



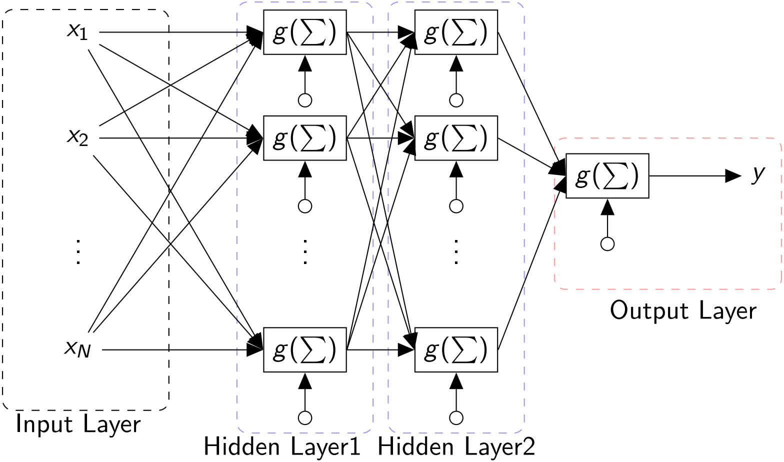
Going Deeper

- As we talked about, the output layer learns a linear separator.
  - Comme on en a parlé, la couche de sortie apprend un sépateur linéaire.

Going Deeper

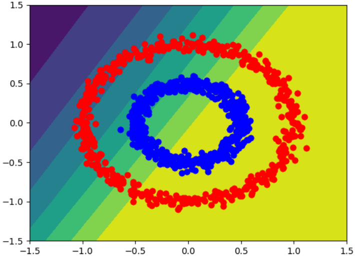
- As we talked about, the output layer learns a linear separator.
  - Comme on en a parlé, la couche de sortie apprend un sépateur linéaire.
- If the output layer is getting not-separable data, we can go deeper!
  - Si la couche de sortie prend du data non-séperable, on peut utiliser un réseau plus profond.

Three layer feed-forward neural network



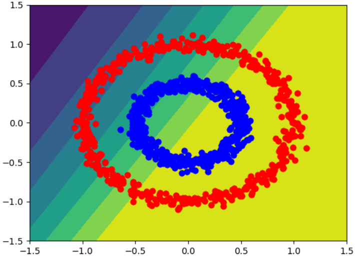
So we can just go arbirarily deep?

- No! Here's what you learn with 7 layers! / Non! voici ce qu'on apprend avec 7 couches:



So we can just go arbirarily deep?

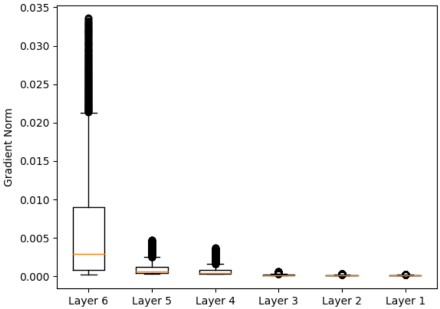
- No! Here's what you learn with 7 layers! / Non! voici ce qu'on apprend avec 7 couches:



- We basically can not learn anything beyond a linear classifier! Why?
  - On peut apprendre qqch au délà d'un classificateur linéaire. Pourquoi?

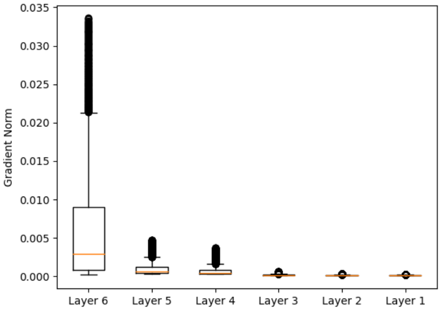
Vanishing gradients

- Here's the gradient distribution with respect to layers / La distribution de gradient par rapport aux couches



Vanishing gradients

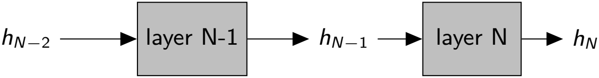
- Here's the gradient distribution with respect to layers / La distribution de gradient par rapport aux couches



- We basically do not update the lower layers! / On ne mets pas les premières couches à jour!

Skip connections

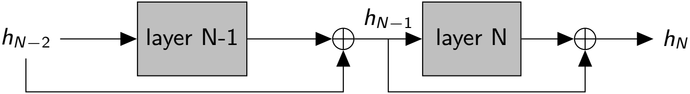
- What if we introduce skip connections?
  - On va introduire des connections qui saute des couches.



- These are known as residual layers also. / On appele ça des 'residual layers' aussi.

Skip connections

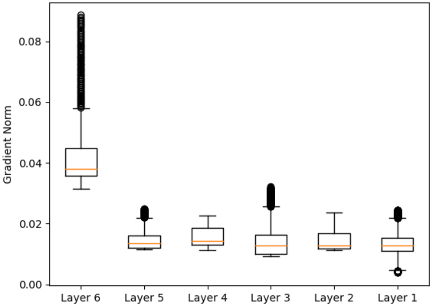
- What if we introduce skip connections?
    - On va introduire des connections qui saute des couches.



- These are known as residual layers also. / On appele ça des 'residual layers' aussi.

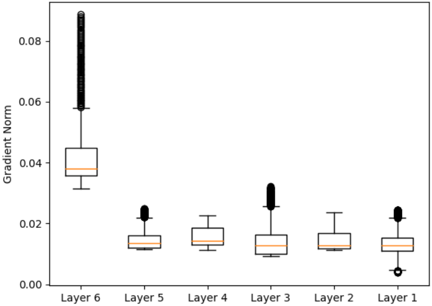
Mitigating vanishing gradients

- Here's the gradient distribution with respect to layers with skip connections / La distribution de gradient par rapport aux couches avec des connections qui sautent



Mitigating vanishing gradients

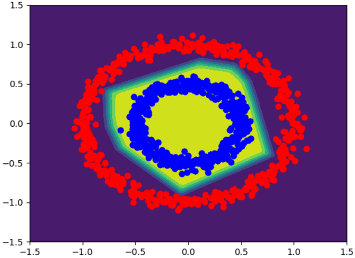
- Here's the gradient distribution with respect to layers with skip connections / La distribution de gradient par rapport aux couches avec des connections qui sautent



- Much better! / Vraiment meilleure!

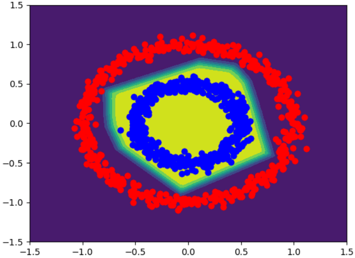
Residual layers in action

- We use a 7 layer residual feed-forward network / On emploie un réseau residual à 7 couches



Residual layers in action

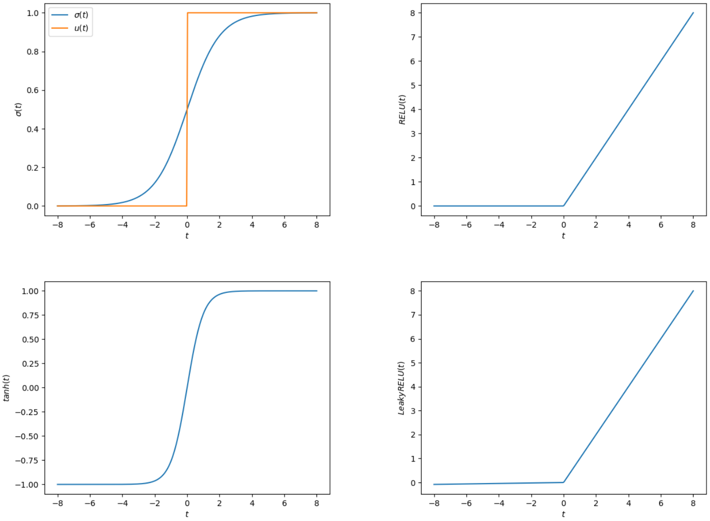
- We use a 7 layer residual feed-forward network / On emploie un réseau residual à 7 couches



- Exact same network as before, but I introduced the skip connections. 6 layers with 3 hidden units, and an output layer with leaky relu activation.
  - Le meme réseau que avant, mais j'ai introduit des connections qui saute. 6 couches avec 3 unités cachés chaqu'un, et on utilise des leaky relu non-linéarité.

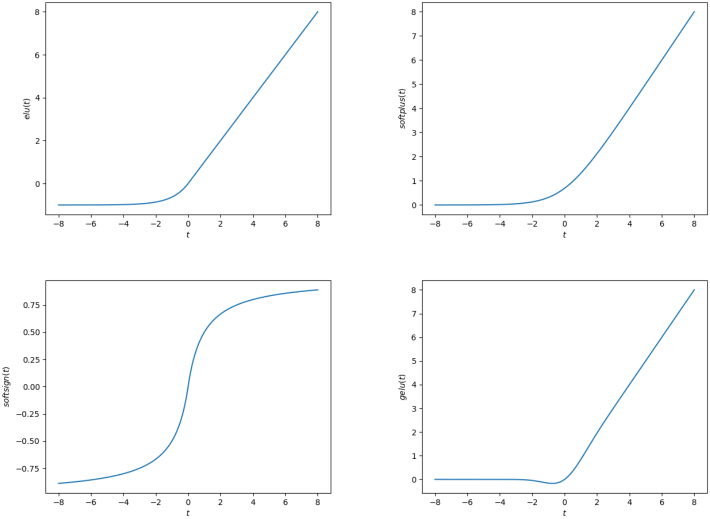
Nonlinearity Functions

- Which one to choose? / Lequel doit-on choisir?

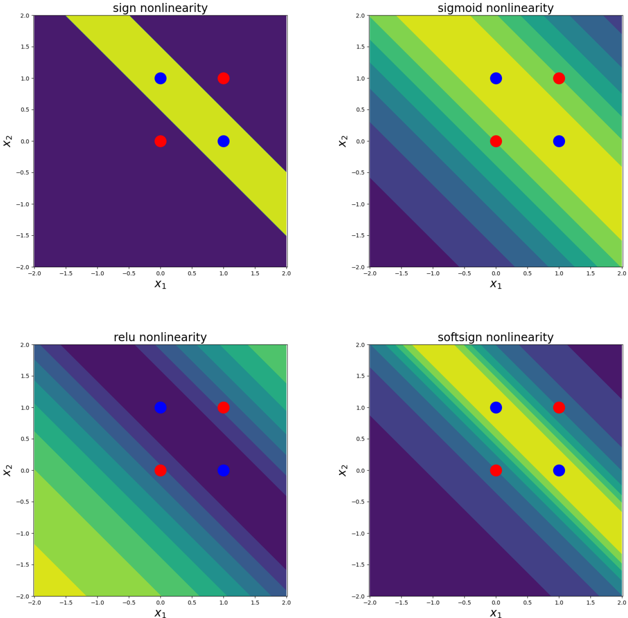


Activation Functions

- Which one to choose? / Lequel doit-on choisir?



Effect on the XOR problem



Which one to choose?

- The effect is mainly on optimization.
  - L'effet est principalement sur l'optimization.

Which one to choose?

- The effect is mainly on optimization.
  - L'effet est principalement sur l'optimization.
- The saturating non-linearities can make it so that you get stuck during optimization!
  - Les non-linéarités qui saturisent fait en sorte qu'on est bouché pendant l'optimization!

Which one to choose?

- The effect is mainly on optimization.
  - L'effet est principalement sur l'optimization.
- The saturating non-linearities can make it so that you get stuck during optimization!
  - Les non-linéarités qui saturisent fait en sorte qu'on est bouché pendant l'optimization!
- Overall there is no definitive answer, except some cases. (e.g. inputting negative numbers to log)

Table of Contents

Do you we can do the same thing here?

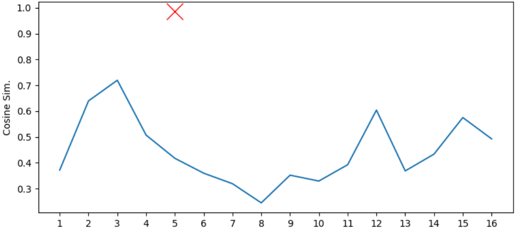
- Do you think dot products would work here? / Pensez-vous que le produit scalaire fonctionnerait içi?


Query


Classes

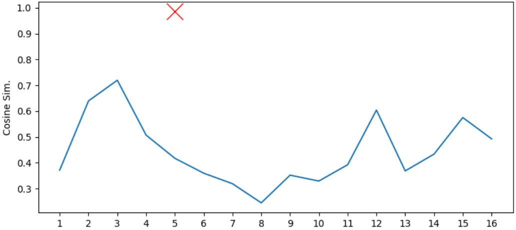
No!

- Inner products / Produits scalaires



No!

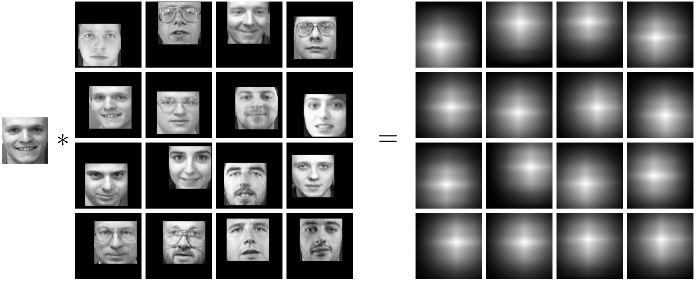
- Inner products / Produits scalaires



- No clear signal, what can we do? / Pas de signal claire, qu'est-ce qu'on peut faire?

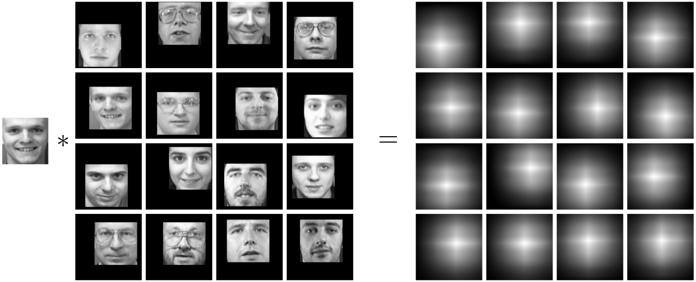
Convolutions! (well technically not)

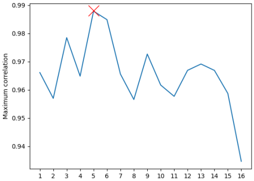
- Correlations!



Convolutions! (well technically not)

- Correlations!





Correlation operation

- The operation / L'opération:

$$y(t) = x * h = \sum_{t'} x(t+t')h(t')$$

Correlation operation

- The operation / L'opération:

$$y(t) = x * h = \sum_{t'} x(t+t')h(t')$$

- Note that this is technically 'correlation' in the signal processing, not convolution. But ML people call it convolution.
  - Notez qu'en traitement de signal, on appelle cette opération 'corrélation'. Mais les gens dans ML appelle ça la convolution.

More than just translation

- Now let's say we want to have a model to detect the same guy in images like this. / Disons qu'on veut détecter la meme personne dans des images variés comme les suivantes.

More than just translation

- Now let's say we want to have a model to detect the same guy in images like this. / Disons qu'on veut détecter la meme personne dans des images variés comme les suivantes.

More than just translation

- Now let's say we want to have a model to detect the same guy in images like this. / Disons qu'on veut détecter la meme personne dans des images variés comme les suivantes.

- Note that we need to be able to do correlations with multiple filters! (to account for rotations, scaling, and more)
  - Notez qu'on a besoin d'effectuer des correlations avec plusieurs filtres afin de prendre en compte les rotations, de plusiers échelles, et plus.

Convolutional Layers to the rescue

- Augment the operation / Augmentons l'opération

$$y(t, c) = x * h(c) = \sum_{t'} x(t + t')h(t', c)$$

Convolutional Layers to the rescue

- Augment the operation / Augmentons l'opération

$$y(t, c) = x * h(c) = \sum_{t'} x(t + t')h(t', c)$$

- The 2D version:

$$y(i, j, c) = x * h(c) = \sum_{i', j'} x(i + i', j + j')h(i', j', c)$$

Convolutional Layers to the rescue

- Augment the operation / Augmentons l'opération

$$y(t,c) = x * h(c) = \sum_{t'} x(t+t')h(t',c)$$

- The 2D version:

$$y(i,j,c) = x * h(c) = \sum_{i',j'} x(i+i',j+j')h(i',j',c)$$

- The Multichannel version

$$y(t,c_2) = \sum_{c_1} x(c_1) * h(c_1,c_2) = \sum_{t',c_1} x(t+t',c_1)h(t',c_1,c_2)$$

Convolutional Layers to the rescue

- Augment the operation / Augmentons l'opération

$$y(t, c) = x * h(c) = \sum_{t'} x(t + t')h(t', c)$$

- The 2D version:

$$y(i, j, c) = x * h(c) = \sum_{i', j'} x(i + i', j + j')h(i', j', c)$$

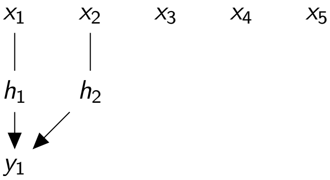
- The Multichannel version

$$y(t, c_2) = \sum_{c_1} x(c_1) * h(c_1, c_2) = \sum_{t', c_1} x(t + t', c_1)h(t', c_1, c_2)$$

- Convolutions can be strided also! Les convolutions peut-etre 'strided' aussi.

Strided convolutions

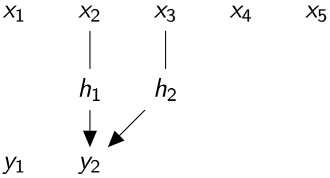
- Stride is the hop size we saw earlier for STFT / Stride est le 'hop size' qu'on a vu avant
- Stride = 1



- This enable to subsample and convolve at the same time. / On sous-echantillone et convolue simultanement.

Strided convolutions

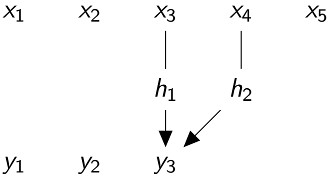
- Stride is the hop size we saw earlier for STFT / Stride est le 'hop size' qu'on a vu avant
- Stride = 1



- This enable to subsample and convolve at the same time. / On sous-echantillone et convolue simultanement.

Strided convolutions

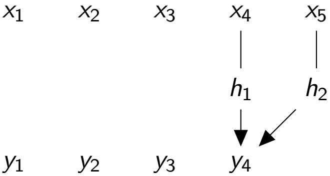
- Stride is the hop size we saw earlier for STFT / Stride est le 'hop size' qu'on a vu avant
- Stride = 1



- This enable to subsample and convolve at the same time. / On sous-echantillone et convolue simultanement.

Strided convolutions

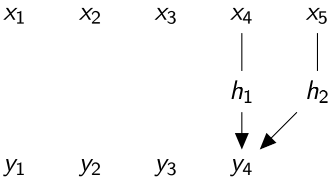
- Stride is the hop size we saw earlier for STFT / Stride est le 'hop size' qu'on a vu avant
- Stride $= 1$



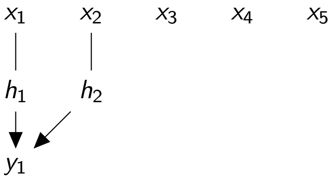
- This enable to subsample and convolve at the same time. / On sous-echantillone et convolue simultanement.

Strided convolutions

- Stride is the hop size we saw earlier for STFT / Stride est le ‘hop size’ qu’on a vu avant
- Stride = 1



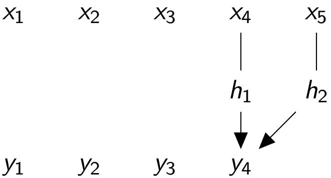
- Stride = 2



- This enable to subsample and convolve at the same time. / On sous-echantillone et convolue simultanement.

Strided convolutions

- Stride is the hop size we saw earlier for STFT / Stride est le ‘hop size’ qu’on a vu avant
- Stride $= 1$

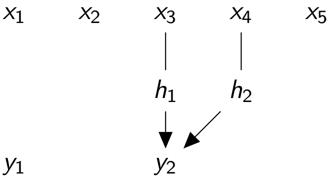


x_1 x_2 x_3 x_4 x_5

h_1 h_2

y_1 y_2 y_3 y_4

- Stride $= 2$



x_1 x_2 x_3 x_4 x_5

h_1 h_2

y_1 y_2

- This enable to subsample and convolve at the same time. / On sous-echantillone et convolue simultanement.

Batch Norm

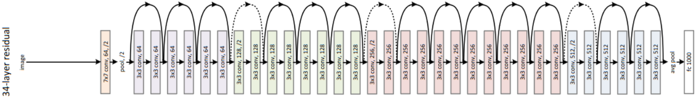
- The operation / L'opération:

$$BN(x) = \frac{x - \mathbb{E}[x]}{\sqrt{\text{var}[x] + \epsilon}}\gamma + \beta$$

- $x, \epsilon, \gamma, \beta \in \mathbb{R}^L$. ϵ, γ, β are learnt on training data / sont appris pendant l'entrainement.
- The statistics $\mathbb{E}[x], \text{var}[x]$ are estimated from the training data/sont estimés pendant l'entrainement.

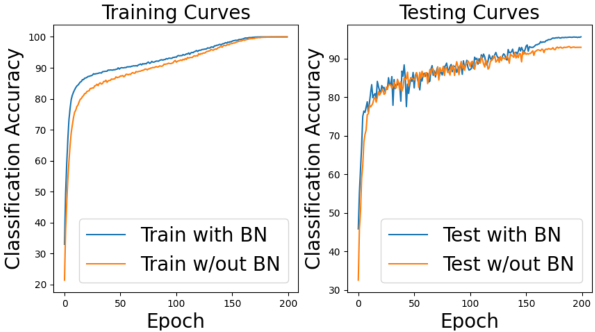
With and without batchnorm

- The architecture (Resnet18)



- We will test the performance with and without batchnorm on CIFAR10. / On va tester la performance avec et sans batchnorm sur CIFAR10.

With and without batchnorm



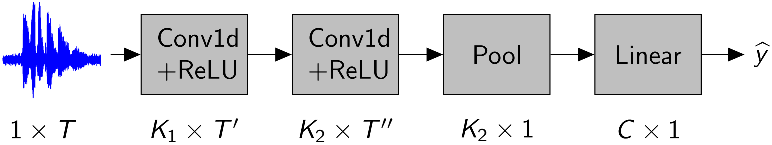
It's slightly better with BN! / C'est un peu mieux avec BN!

Convolutional Nets for Classifying Audio

- How can we use Convolutional Nets for Classifying Audio? / Comment peut-on utiliser des réseaux convolutionnels pour classifier audio?
- Two options: Time domain / Time-Frequency Domain
  - Deux options / Domaine Temporelle / Domaine Fréq-Temporelle
- Let's explore both! / Éxplorons les deux!

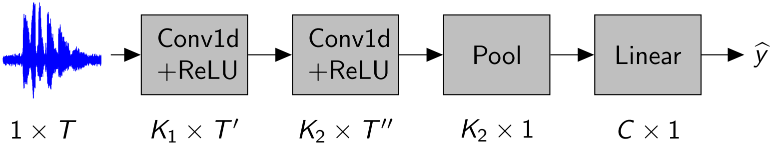
Audio Classification Pipeline

- Time domain classification pipeline / Le pipeline de classification dans le domaine de temps



Audio Classification Pipeline

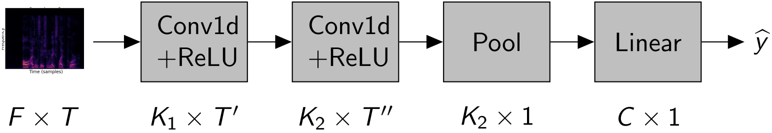
- Time domain classification pipeline / Le pipeline de classification dans le domaine de temps



Conv1d +ReLU → Conv1d +ReLU → Pool → Linear → $\widehat{y}$

$1 \times T$ $K_1 \times T'$ $K_2 \times T''$ $K_2 \times 1$ $C \times 1$

- Time-Frequency domain classification pipeline / Le pipeline de classification dans le domain frequence-et-temps

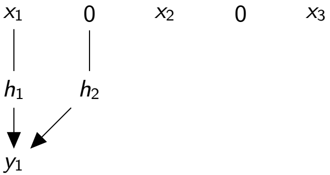


Conv1d +ReLU → Conv1d +ReLU → Pool → Linear → $\widehat{y}$

$F \times T$ $K_1 \times T'$ $K_2 \times T''$ $K_2 \times 1$ $C \times 1$

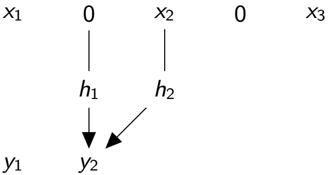
Transposed Convolution

- An upsampling operation / Une opération de superresolution.
- Very similar to the upsampling in DSP: Insert zeros, then filter (convolve with stride=1):
  - Comme le DSP classique on insére des zéros, puis filtrer avec un stride=1.



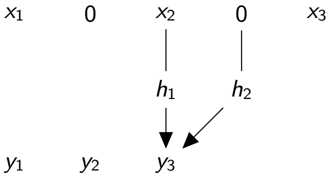
Transposed Convolution

- An upsampling operation / Une opération de superresolution.
- Very similar to the upsampling in DSP: Insert zeros, then filter (convolve with stride=1):
  - Comme le DSP classique on insére des zéros, puis filtrer avec un stride=1.



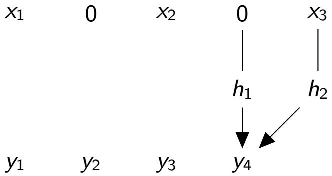
Transposed Convolution

- An upsampling operation / Une opération de superresolution.
- Very similar to the upsampling in DSP: Insert zeros, then filter (convolve with stride=1):
  - Comme le DSP classique on insére des zéros, puis filtrer avec un stride=1.



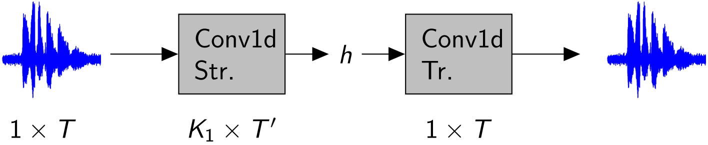
Transposed Convolution

- An upsampling operation / Une opération de superresolution.
- Very similar to the upsampling in DSP: Insert zeros, then filter (convolve with stride=1):
  - Comme le DSP classique on insére des zéros, puis filtrer avec un stride=1.



Transposed Convolution Application: Autoencoder

- We can subsample with strided conv layers, and then upsample with transposed-conv layers. / On peut sousechantillonner avec des couches strided-conv, et puis augmenter la résolution avec des couches transposed-conv.



Convolution Groups

- Grouping decides how to map M input dimensions to N output dimensions using K filters. / Les groupes décident comment mapper les M dimensions d'input à N dimensions d'output en utilisant K filtres.

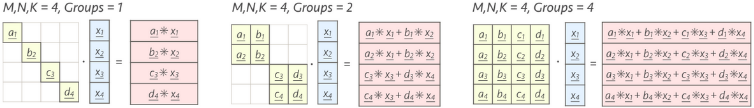


image taken from uiuc mlsp class

- The default value is 1, this keeps each dimension separate. If we want to mix the channels set groups to a larger value.
  - La valeur par défaut est 1. Ça traite les dimensions séparement. Si on veu mélanger les canaux on peut fixer les n. groups à une valeur supérieure.

Table of Contents

Using Convolutional Nets as TDNN

- Let's say we work with bit sequences. / Disons qu'on travaille avec des séquences binaires.

Using Convolutional Nets as TDNN

- Let's say we work with bit sequences. / Disons qu'on travaille avec des séquences binaires.
- XOR again: $y_t = XOR(x_t, x_{t-\tau})$. CNNs will have finite attention span. / Les CNNs regardent un fenetre fixe.
- If τ is large enough CNNs will fail, no matter what! / Si τ est large, les CNNs vont échouer.

Recurrent Neural Nets

- The vanilla RNN Idea:

$$h_t = g(Wh_{t-1} + Ux_t)$$

- $h_t \in \mathbb{R}^K$ defines the 'state'. / définit l'état. $W \in \mathbb{R}^{K \times K}$.
- $x_t \in \mathbb{R}^L$ is in the input / est l'entrée. $U \in \mathbb{R}^{K \times L}$.
- We define the temporal dependencies with a recursion, so we have infinite attention span (in theory) / En théorie on peut modéliser des dépendes à l'infinie.
- There will be a vanishing gradient problem. / On va avoir un probleme de gradients qui se disparaient.

$$\begin{aligned}
h_t =& g(Wg(Wh_{t-2} + Ux_{t-2}) + Ux_t) \\
=& g(Wg(W(g(Wh_{t-3} + Ux_{t-3}) + Ux_{t-2}) + Ux_t) \\
=& g(Wg(Wg(W(\dots g(Wh_0 + Ux_0)\dots) + Ux_{t-3}) + Ux_{t-2}) + Ux_t)
\end{aligned}$$

Recurrent Neural Nets

- The vanilla RNN Idea:

$$h_t = g(Wh_{t-1} + Ux_t)$$

- $h_t \in \mathbb{R}^K$ defines the 'state'. / définit l'état. $W \in \mathbb{R}^{K \times K}$.
- $x_t \in \mathbb{R}^L$ is in the input / est l'entrée. $U \in \mathbb{R}^{K \times L}$.
- We define the temporal dependencies with a recursion, so we have infinite attention span (in theory) / En théorie on peut modéliser des dépendes à l'infinie.
- There will be a vanishing gradient problem. / On va avoir un probleme de gradients qui se disparaient.

$$\begin{aligned} h_t =& g(Wg(Wh_{t-2} + Ux_{t-2}) + Ux_t) \\ =& g(Wg(W(g(Wh_{t-3} + Ux_{t-3}) + Ux_{t-2}) + Ux_t) \\ =& g(Wg(Wg(W(\ldots g(Wh_0 + Ux_0) \ldots) + Ux_{t-3}) + Ux_{t-2}) + Ux_t) \end{aligned}$$

- This will definitely suffer from that for long term dependencies. / Ça va définitivement avoir des problèmes pour des dépendences longue termes.

RNNs with gates

- Minimal gated unit:

$$
\begin{aligned}
f_t =&\sigma(W_f h_{t-1} +_f Ux_t) \\
h_t =&f_t \odot h_{t-1} + (1 - f_t) \odot \tanh(W_h h_{t-1} + U_h x_t)
\end{aligned}
$$

- This allows better gradient flow. / Ça permet mieux débit des gradients.

GRU / LSTM

- GRU:

$$
\begin{aligned}
f_t =& \sigma(W_f h_{t-1} + U_f x_t) \\
r_t =& \sigma(W_r h_{t-1} + U_r x_t) \\
h_t =& f_t \odot h_{t-1} + (1 - f_t) \odot \tanh(W_h(r_t \odot h_{t-1}) + U_h x_t)
\end{aligned}
$$

- LSTM: (More gates!)

$$
\begin{aligned}
f_t =& \sigma(W_f h_{t-1} + U_f x_t) \\
i_t =& \sigma(W_i h_{t-1} + U_i x_t) \\
o_t =& \sigma(W_o h_{t-1} + U_o x_t) \\
c_t =& f_t \odot c_{t-1} + i_t \odot \tanh(W_c h_{t-1} + U_c x_t) \\
h_t =& o_t \odot \tanh(c_t)
\end{aligned}
$$

So CNN or RNN?

- It really depends, usually they give similar results. / Ça depend. En générale les résultats sont similaires.
- CNNs are typically easier to parallelize. / Les CNNs sont typiquement plus facile à paralleliser.

Table of Contents

Self-attention

- Self-attention is the basis of the transformer architecture which obtains state-of-the art results in several domains such as NLP, computer vision, speech recognition. / Self-attention est base de l'architecture transformer qui obtient SOTA dans plusieurs domains.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V$$

$$Q = XW^Q,\ K = XW^K,\ V = XW^V$$

Self-attention

- Self-attention is the basis of the transformer architecture which obtains state-of-the art results in several domains such as NLP, computer vision, speech recognition. / Self-attention est base de l'architecture transformer qui obtient SOTA dans plusieurs domains.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V$$

$$Q = XW^Q,\ K = XW^K,\ V = XW^V$$

$$\begin{bmatrix} \sum_j a_{0j} v_j \\ \sum_j a_{1j} v_j \\ \sum_j a_{2j} v_j \\ \sum_j a_{3j} v_j \end{bmatrix} = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}$$

Self-attention

- Self-attention is the basis of the transformer architecture which obtains state-of-the art results in several domains such as NLP, computer vision, speech recognition. / Self-attention est base de l'architecture transformer qui obtient SOTA dans plusieurs domains.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V$$

$$Q = XW^Q,\ K = XW^K,\ V = XW^V$$

$$\begin{bmatrix} v_3 \\ \frac{v_3}{2} + \frac{v_4}{2} \\ v_2 \\ v_1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}$$

How to apply Self-Attention on Source Separation?

- It is not straightforward how to apply self-attention on long sequences. / C'est pas facile appliquer self-attention directement sur des séquences longues.
- Self-attention has a quadratic complexity with respect to inputs size. / Self-attention a une complexité quadratique par rapport à la taille de l'entrée.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V,$$

$$Q = XW^Q,\ K = XW^K,\ V = XW^V$$

How to apply Self-Attention on Source Separation?

- It is not straightforward how to apply self-attention on long sequences. / C'est pas facile appliquer self-attention directement sur des séquences longues.
- Self-attention has a quadratic complexity with respect to inputs size. / Self-attention a une complexité quadratique par rapport à la taille de l'entrée.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V,$$

$$Q = XW^Q,\ K = XW^K,\ V = XW^V$$

- Can we find a way to efficiently model real world signals? / Peut-on trouver une façon pour modeliser des séquences longues?

How to apply Self-Attention on Source Separation?

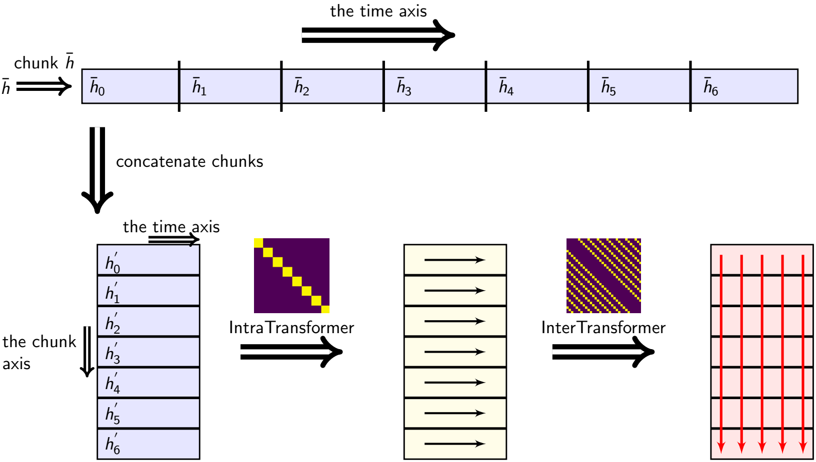
- It is not straightforward how to apply self-attention on long sequences. / C'est pas facile appliquer self-attention directement sur des séquences longues.
- Self-attention has a quadratic complexity with respect to inputs size. / Self-attention a une complexité quadratique par rapport à la taille de l'entrée.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V,$$

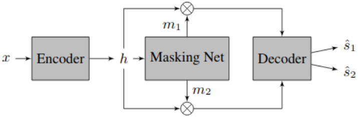
$$Q = XW^Q,\ K = XW^K,\ V = XW^V$$

- Can we find a way to efficiently model real world signals? / Peut-on trouver une façon pour modeliser des séquences longues?
- We propose **SepFormer**! / Nous avong proposé SepFormer qui a obtenu des résultats SOTA dans speech separation.

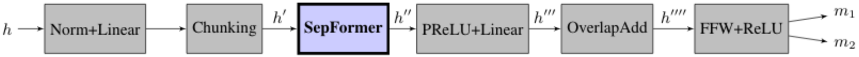
The Dual-Path Transformer Pipeline



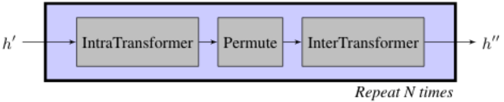
The whole network



Masking Network



SepFormer



Transformer

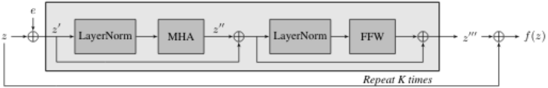


Table of Contents

A note on optimization

- Neural Network Optimization is highly non-convex. / Optimization des réseaux de neurons sont hautement non-convex.
- A convex optimization problem

$$\begin{aligned} &\min_x f(x) \\ &s.t. g_i(x) \leq 0,\ i \in 1, \ldots, m \\ &\quad h_j(x) = 0\ j \in 1, \ldots, p \end{aligned}$$

  - $f(x)$ needs to be convex
  - $g_i(x)$ needs to be convex
  - $h_j(x)$ needs to be affine.

A note on optimization

- Neural Network Optimization is highly non-convex. / Optimization des réseaux de neurons sont hautement non-convex.
- A convex optimization problem

$$\begin{aligned} &\min_x f(x) \\ s.t. g_i(x) &\leq 0,\ i \in 1, \ldots, m \\ h_j(x) &= 0\ j \in 1, \ldots, p \end{aligned}$$

  - $f(x)$ needs to be convex
  - $g_i(x)$ needs to be convex
  - $h_j(x)$ needs to be affine.
- In NNs $f(x)$ is highly non-convex!

A note on optimization

- Neural Network Optimization is highly non-convex. / Optimization des réseaux de neurons sont hautement non-convex.
- A convex optimization problem

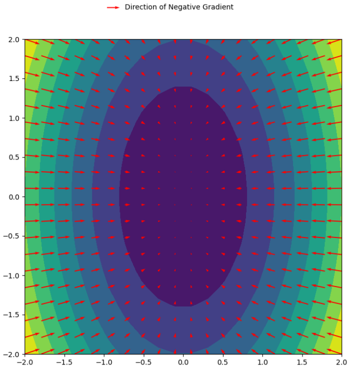
$$\begin{aligned} &\min_{x} f(x) \\ &s.t. g_i(x) \leq 0,\ i \in 1, \ldots, m \\ &\quad h_j(x) = 0\ j \in 1, \ldots, p \end{aligned}$$

  - $f(x)$ needs to be convex
  - $g_i(x)$ needs to be convex
  - $h_j(x)$ needs to be affine.
- In NNs $f(x)$ is highly non-convex!
- Finding globally optimal solution is not guaranteed, harder optimization problem.

Gradient descent

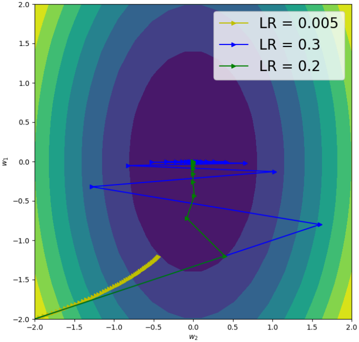
- The basic updates

$$\underbrace{w_t}_{\text{parameters}} = w_{t-1} - \underbrace{\eta}_{\text{learning rate}} \underbrace{\nabla \mathcal{L}(w)}_{thegradient}$$



The effect of learning rate

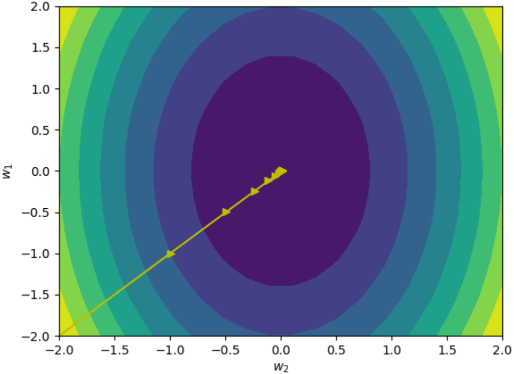
- $\mathcal{L}(w) = 3w_1^2 + w_2^2$



Momentum

- Momentum, low pass filters gradients

$$\begin{aligned} m_t =& \eta \nabla \mathcal{L}(w) + \mu m_{t-1} \\ w_{t+1} =& w_t - m_t \end{aligned}$$

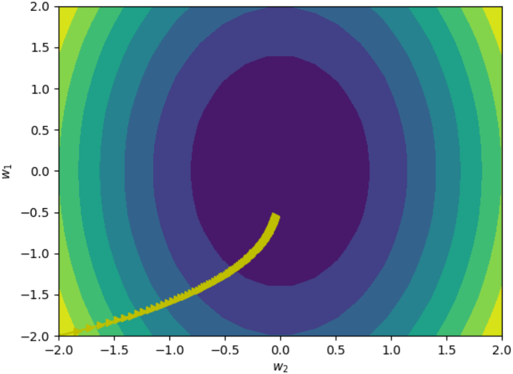


Per Parameter Updates

- Adagrad: The idea is to penalize large updates / L'idée est de pénaliser les mises à jours grands.

$$\begin{aligned} g_{t+1} =& \gamma g_t + (1-\gamma)\nabla\mathcal{L}(w)^2 \\ w_{t+1} =& w_t - \eta\frac{\nabla\mathcal{L}(w)}{\sqrt{\sum_{k=1}^{t+1} g_k^2}} \end{aligned}$$

- Slows down too much! / Ça relantit trop!



More embellished versions

RMSProp

$$g_{t+1} = \gamma g_t + (1-\gamma)\nabla\mathcal{L}(w)^2$$
$$w_{t+1} = w_t - \eta\frac{\nabla\mathcal{L}(w)}{\sqrt{g_{t+1}}}$$

Note the per parameter update, large updates are penalized. / Notez qu'on des mises à jours differents pour chaque parametres afin de pénaliser les grands changements.

Adam

$$m_{t+1} = \beta_1 m_t + (1-\beta_1)\nabla\mathcal{L}(w)$$
$$g_{t+1} = \beta_2 g_t + (1-\beta_2)\nabla\mathcal{L}(w)^2$$
$$w_{t+1} = w_t - \eta\frac{m_{t+1}}{\sqrt{g_{t+1}}}$$

Same as RMSProp but uses momentum smoothing. / La meme chose que RMS Prop mais on utilise momentum smoothing.

Recap

- What changed since the 90s? / Qu'est-ce qui a changé depuis les 90s?

Recap

- What changed since the 90s? / Qu'est-ce qui a changé depuis les 90s?
- Cynical answer: Not much, we just have GPUs now and some bells and whistles. / Réponse cynique: Pas beaucoup.

Recap

- What changed since the 90s? / Qu'est-ce qui a changé depuis les 90s?
- Cynical answer: Not much, we just have GPUs now and some bells and whistles. / Réponse cynique: Pas beaucoup.
- More thoughtful and calm answer: We know how to make things work now, many tricks. (plus compute) / On connait plus des astuces maintenant qui fait les choses fonctionner.

Recap

- What changed since the 90s? / Qu'est-ce qui a changé depuis les 90s?
- Cynical answer: Not much, we just have GPUs now and some bells and whistles. / Réponse cynique: Pas beaucoup.
- More thoughtful and calm answer: We know how to make things work now, many tricks. (plus compute) / On connait plus des astuces maintenant qui fait les choses fonctionner.
- Today we talked about how to make MLPs work, modern conv nets, RNNs, Self-attention. / On a parlé des MLPs, convnets, RNNs, self-attention.

Reading material

- Convolution arithmetics: `https://github.com/vdumoulin/conv_arithmetic`
- Wavenet: `https://arxiv.org/pdf/1609.03499.pdf`
- Transformer: `https://arxiv.org/abs/1706.03762`
- Resnet: `https://arxiv.org/pdf/1512.03385.pdf`

Next week

- Clustering.