Convolutional Neural Networks

in Keras

(CNN)

lesson 03

Convolution

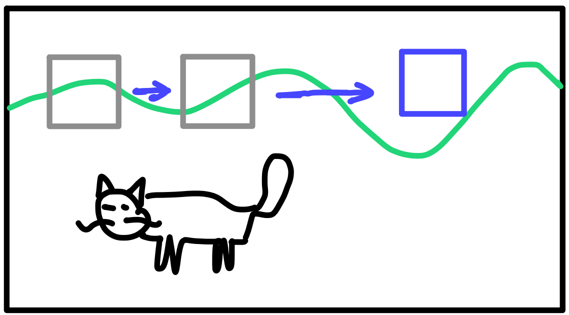
function $f(x)$

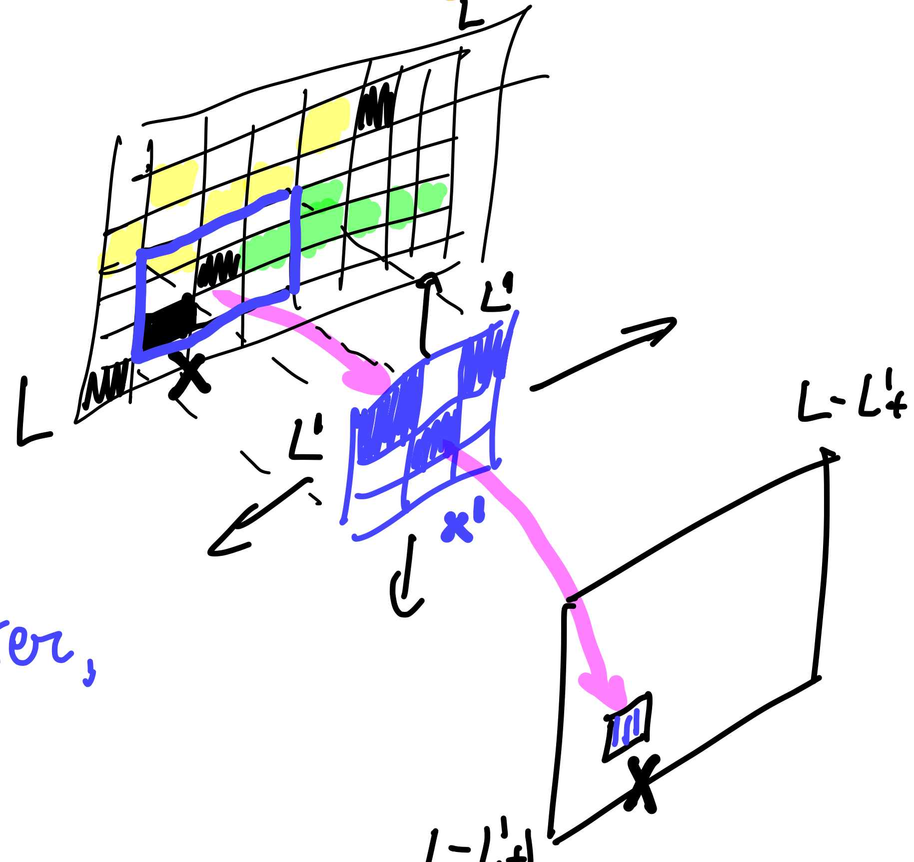
filter function $g(x)$

$$C(\mathbf{x}) = \int_a^b f(\mathbf{x} - x')\, g(x')\, dx$$

Convolutional Neural Networks

in Keras

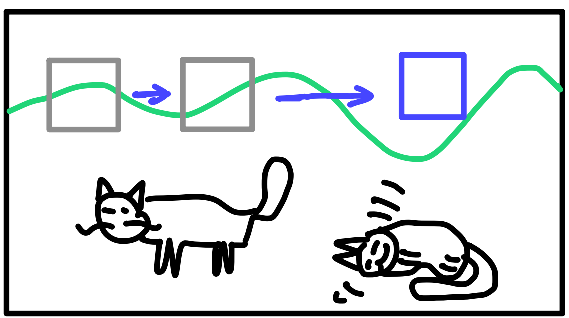




 mask, filter, kernel ...

Convolutional Neural Networks

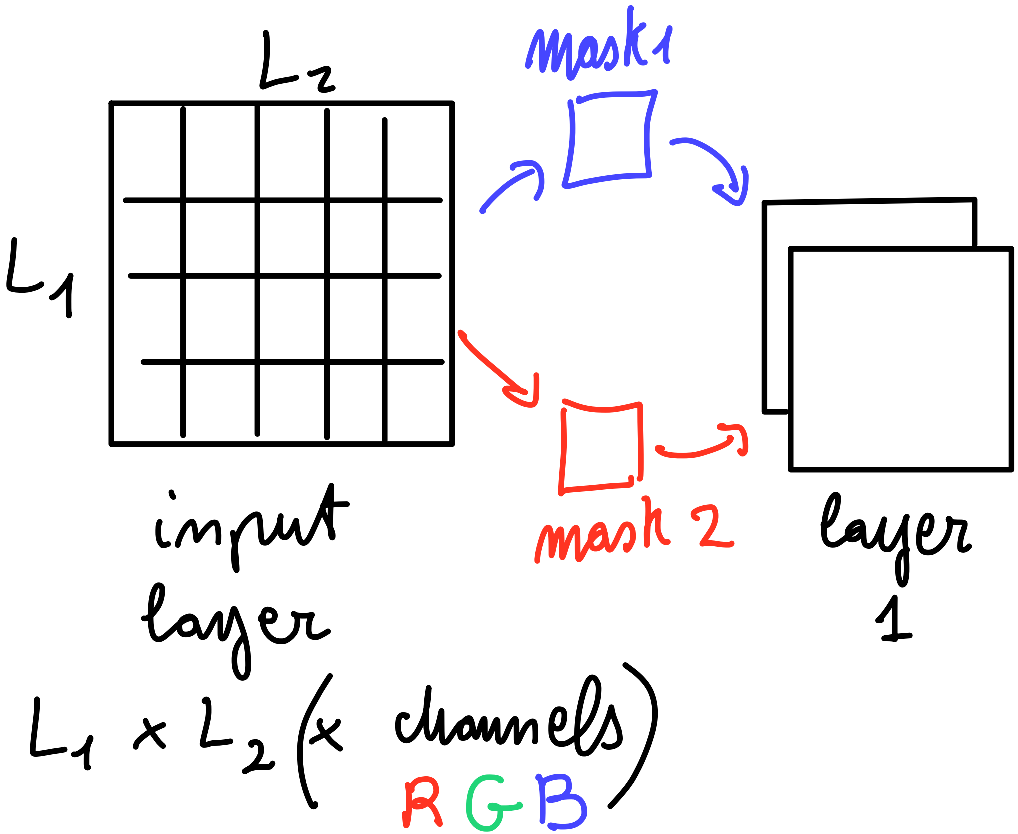
in Keras



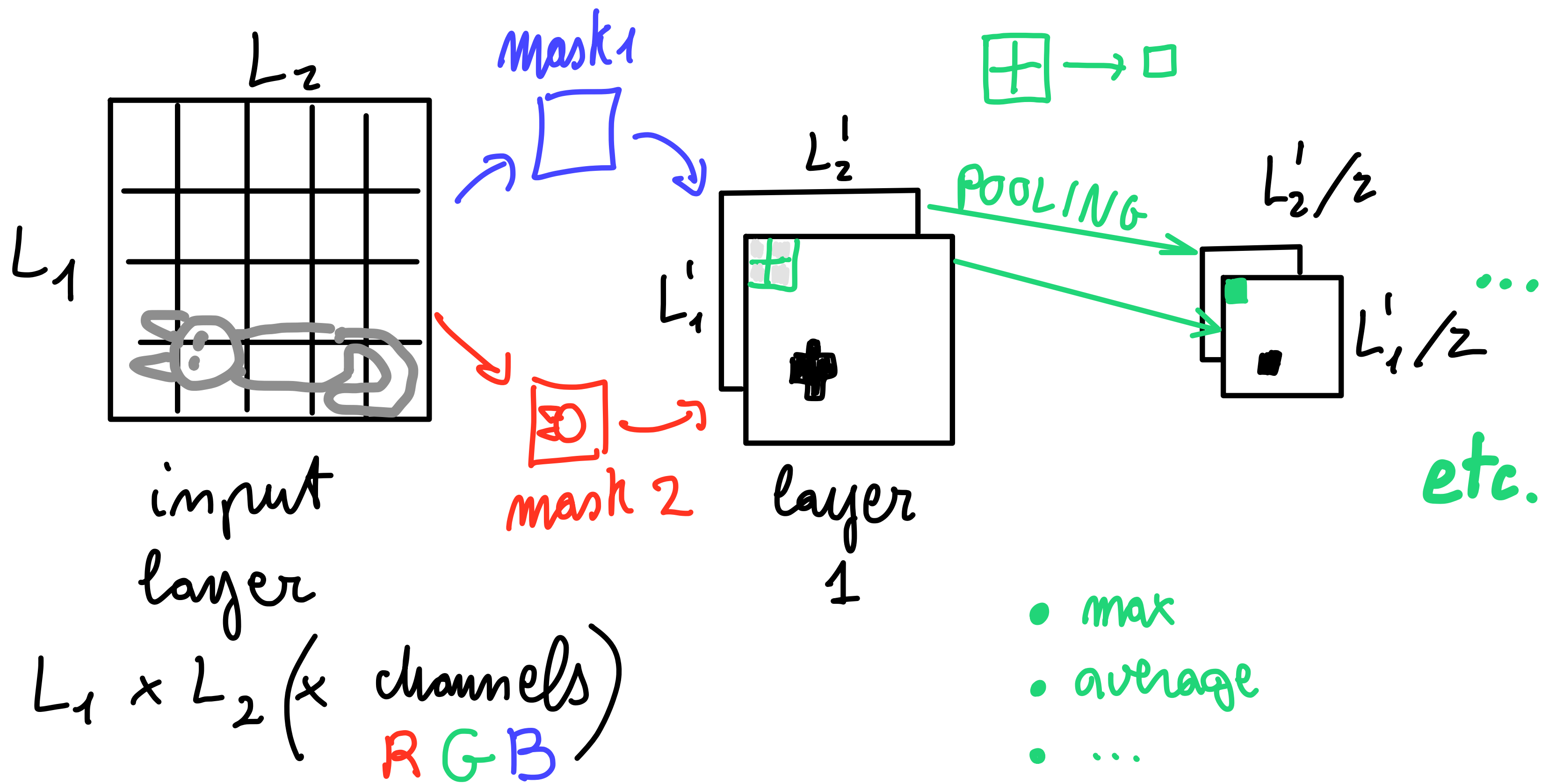
- automatic definition of filters by CNN

 mask 1

 mask 2



L_2
L_1
mask1
mask 2
input layer
layer 1
$L_1 \times L_2$ (x channels
RGB)

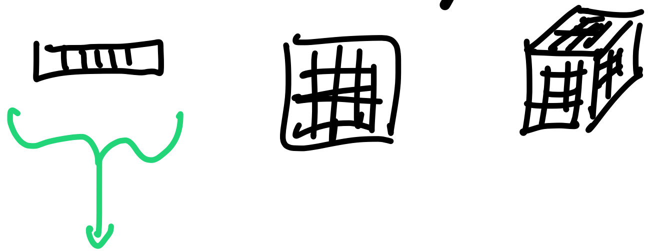

mask 1
L_2
L_1
input layer
$L_1 \times L_2$ (x channels RGB)
mask 2
L_2'
L_1'
layer 1
POOLING
$L_2'/2$
$L_1'/2$
...
etc.
• max
• average
• ...

Why CNN ?

- learning fewer parameters than normal NN
  - more efficient
  - reduce overfitting ?
- logic choice if we look for local structures in each data sample
  - spatial
  - temporal

(translational invariance)

CNN: 1D, 2D, 3D ...

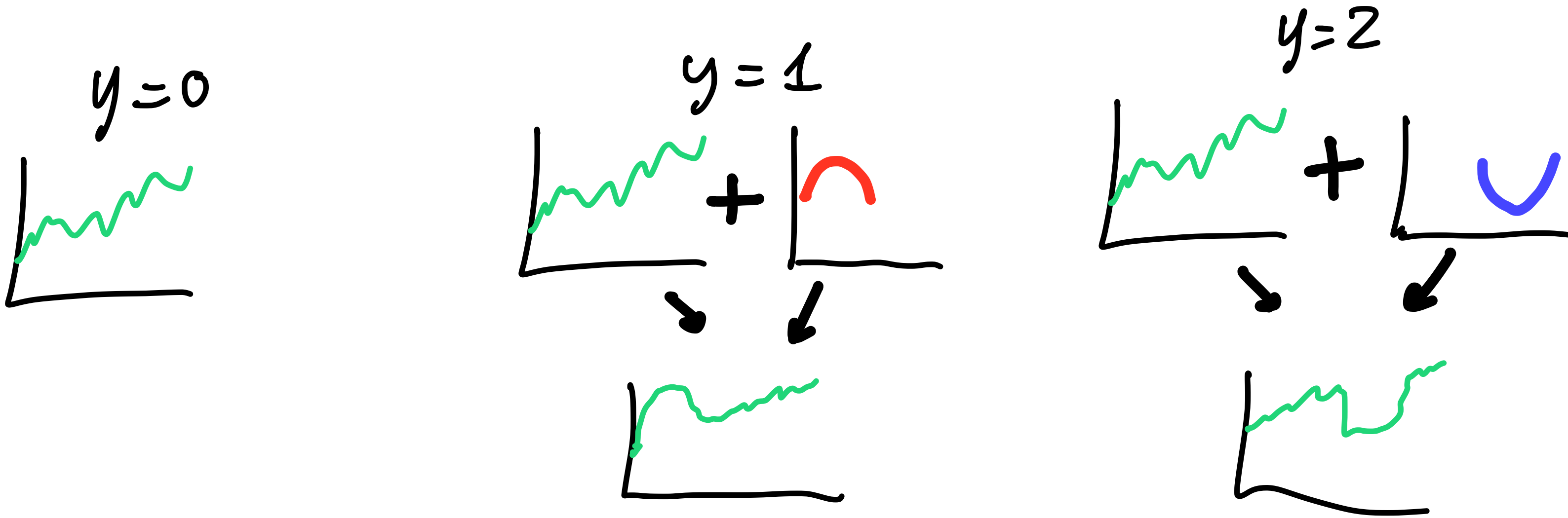


$$x_n = [x_n(0), \dots, x_n(L-1)]$$

exercise 3

- stochastic process : Markov chain $X(t) \xrightarrow{P} X(t+1)$

- samples in 3 categories



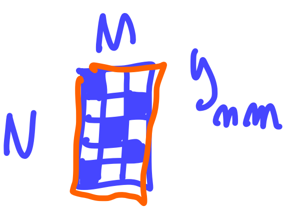
Categories:

$M=3$ categories $\qquad y=(0,1,2)$

$(1,0,0) \quad (0,1,0) \quad (0,0,1)$

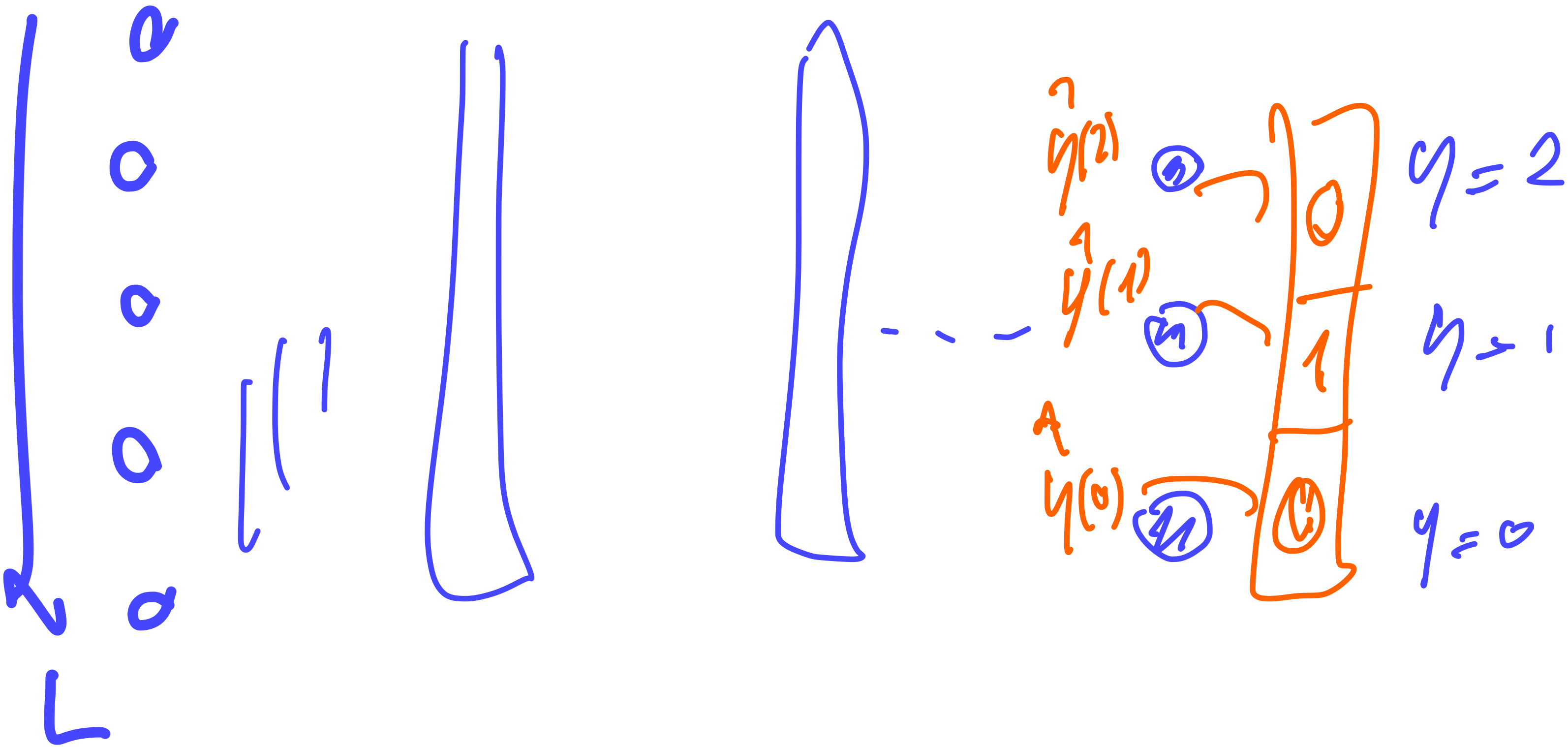
cost function: softmax (eq (81) review)

(probability of each category) p



categorical cross entropy

$$C_\theta = -\sum_{n=1}^{N}\sum_{m=0}^{M-1}\left[y_{nm}\log p(y_{nm}|x_n) + (1-y_{nm})\log\left(1-p(y_{nm}|x_n)\right)\right]$$


L
y(2)
y(1)
y(0)
0
1
y = 2
y = 1
y = 0

Categories:

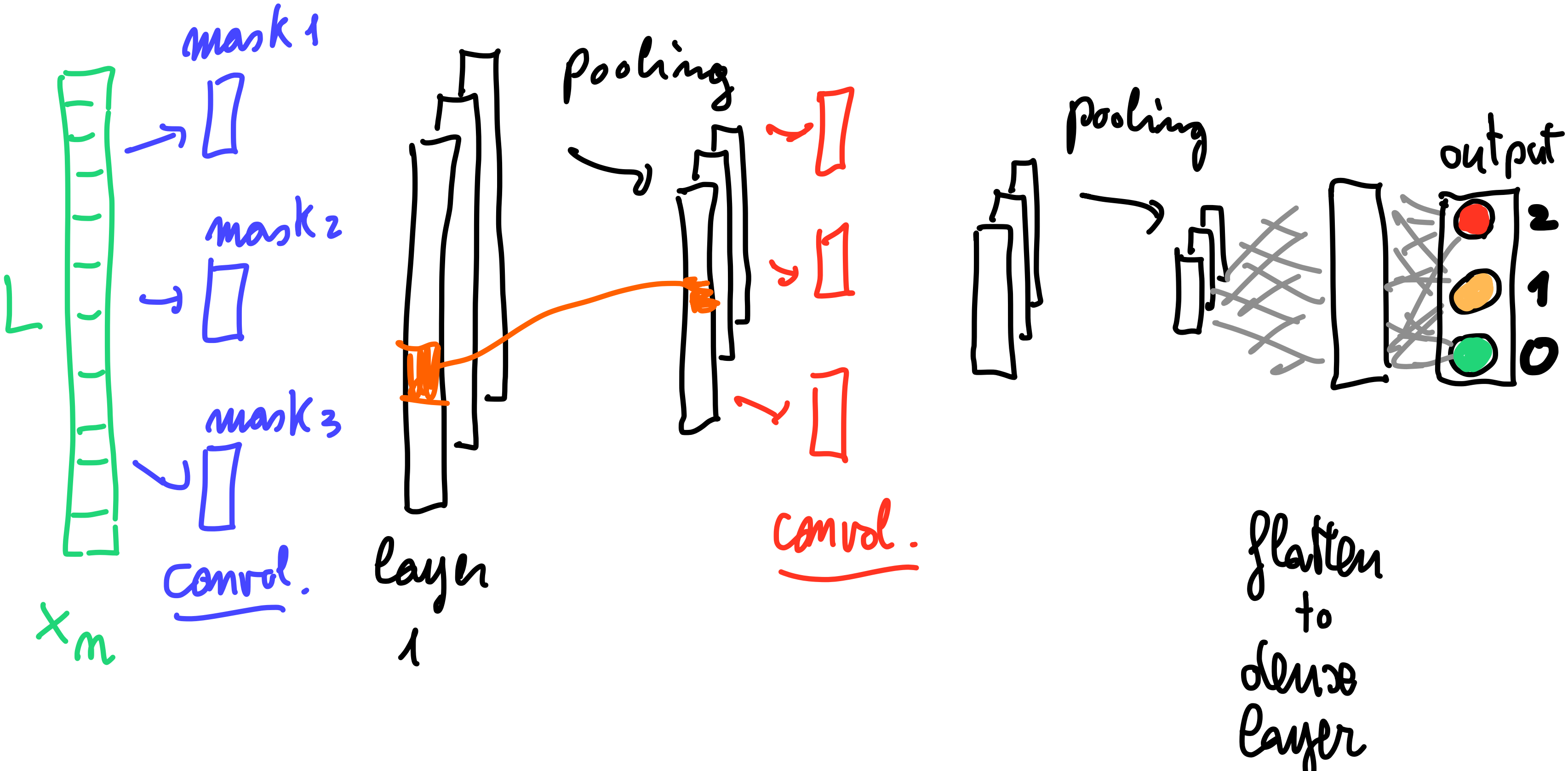
n : $y_n = (0, 1, 0)$ | $p(y_n | x_n) = (0.2, \underline{0.5}, 0.3)$

$p(y_{nm} | x_n)$

θ = parameters (w, b)

$$C_\theta = -\sum_{n=1}^{N} \sum_{m=0}^{M-1} \left[y_{nm} \log p(y_{nm} | x_n) + (1 - y_{nm}) \log\left(1 - p(y_{nm} | x_n)\right) \right]$$

CNN recap



mask 1
mask 2
mask 3
L
x_n
convol.
layer 1
pooling
convol.
pooling
output
2
1
0
flatten to dense layer

CNN

& previous exercise

A	C	G	A	T ....

L_s

$N_c = 4$

$\nearrow$

1000	0100	0010	1000	...

$L = L_s \cdot N_c$

$\Longrightarrow$ standard DNN

+ data augmentation

CNN

& previous exercise

A	C	G	A	T ....

L_s

$N_c = 4$

1000	0100	0010	1000	...

$L = L_s \cdot N_c$

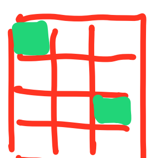
$\Longrightarrow$ standard DNN

+ data augmentation

$$N_c \left[\begin{array}{ccccc} 1 & 0 & 0 & 1 & \\ 0 & 1 & 0 & 0 & \\ 0 & 0 & 1 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \end{array}\right]$$

L_s

$\Longrightarrow$ 2D CNN, no augment.

mask  if ...A.G.... key was required

Regularization:

cost function C_θ $\longrightarrow$ $\underbrace{C_\theta + \lambda R_\theta}_{\text{to be minimized}}$
(loss)

- **Ridge** : $R_\theta = \sum_i |\theta_i|^2$

(θ_i = one of the weights in W, b)

- **LASSO** : $R_\theta = \sum_i |\theta_i|$

λ: multiplier, tunes strength of regulariz.

Regularization:

cost function $C_\theta \longrightarrow \underbrace{C_\theta + \lambda R_\theta}_{\text{to be minimized}}$

- <u>Ridge</u> : $R_\theta = \sum_i |\theta_i|^2$
- <u>LASSO</u> : $R_\theta = \sum_i |\theta_i|$

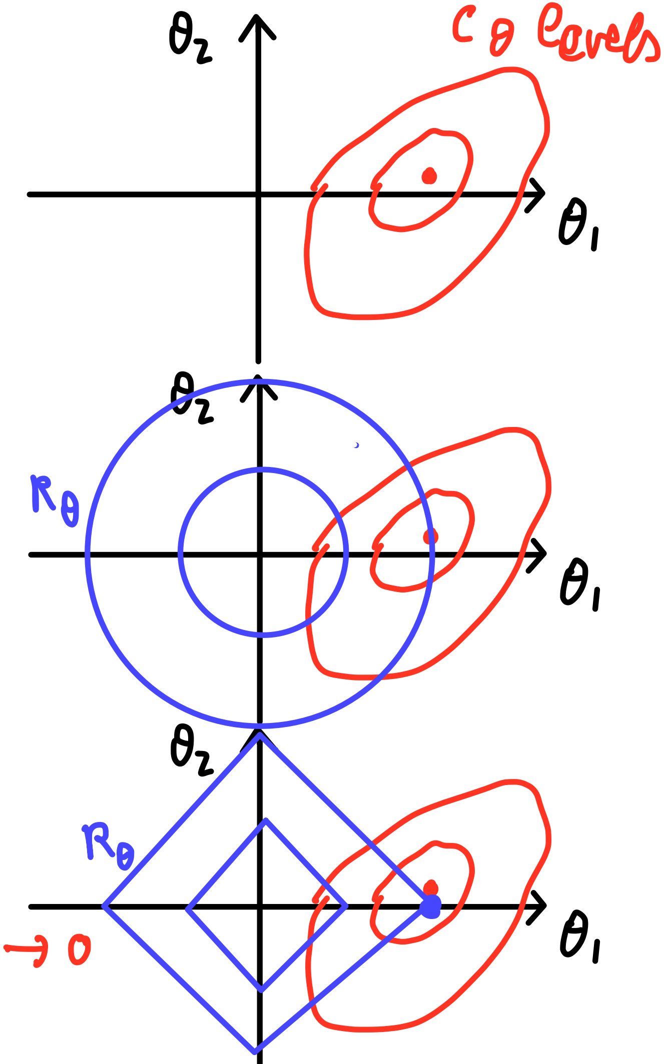
- <u>reduce overfitting</u>
- <u>improve readability</u>

Regularization:

$$\underbrace{C_\theta + \lambda R_\theta}_{\text{to be minimized}}$$

- Ridge : $R_\theta = \sum_i |\theta_i|^2$

- LASSO : $R_\theta = \sum_i |\theta_i|$



L_1-norm in LASSO forces many weights $\to 0$