MACHINE LEARNING BASICS

CMSC764 / AMSC607

LINEAR CLASSIFIERS

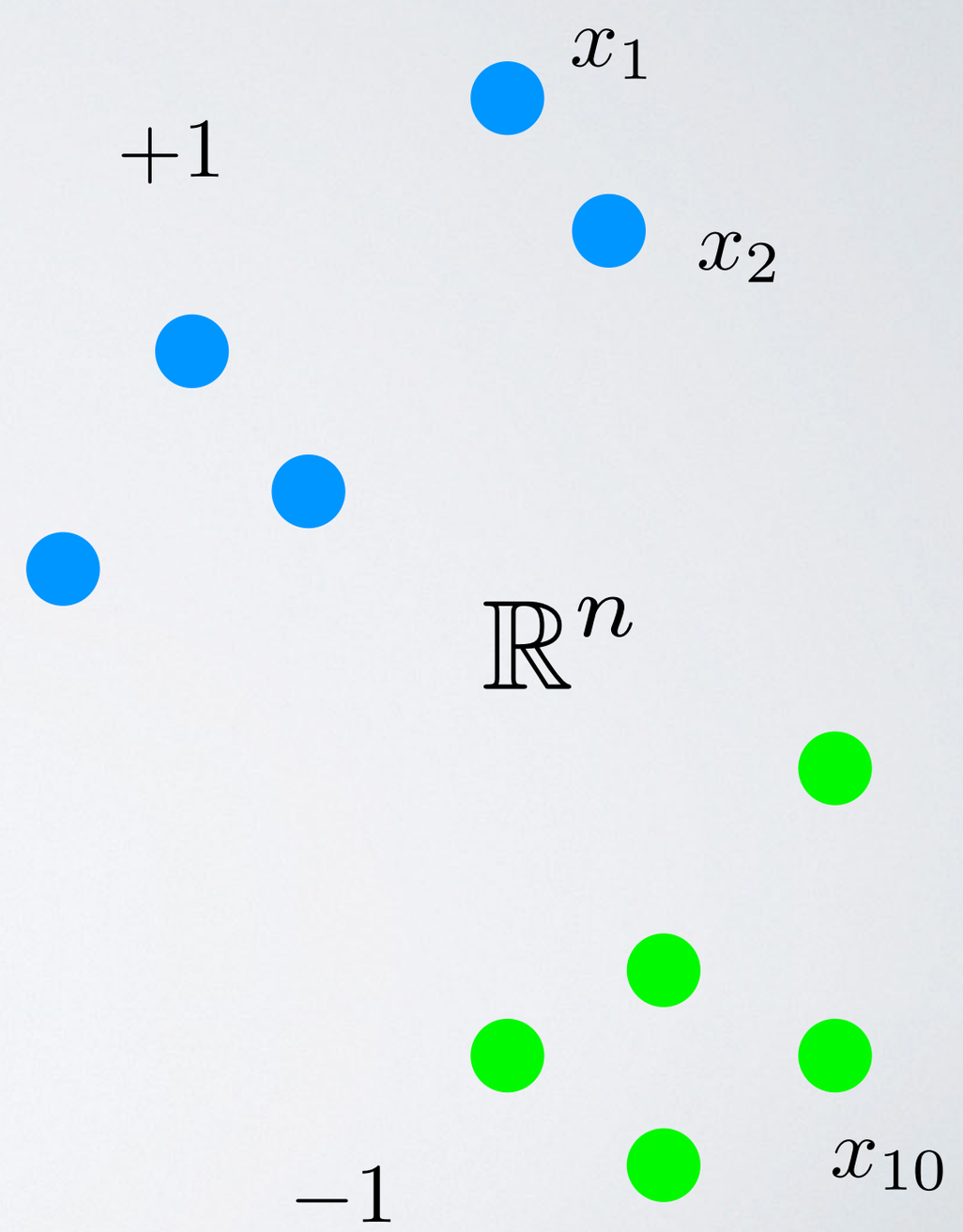
training data

feature vectors

vectors containing descriptions of objects

labels

+1/-1 labels indicating which "class" each vector lies in

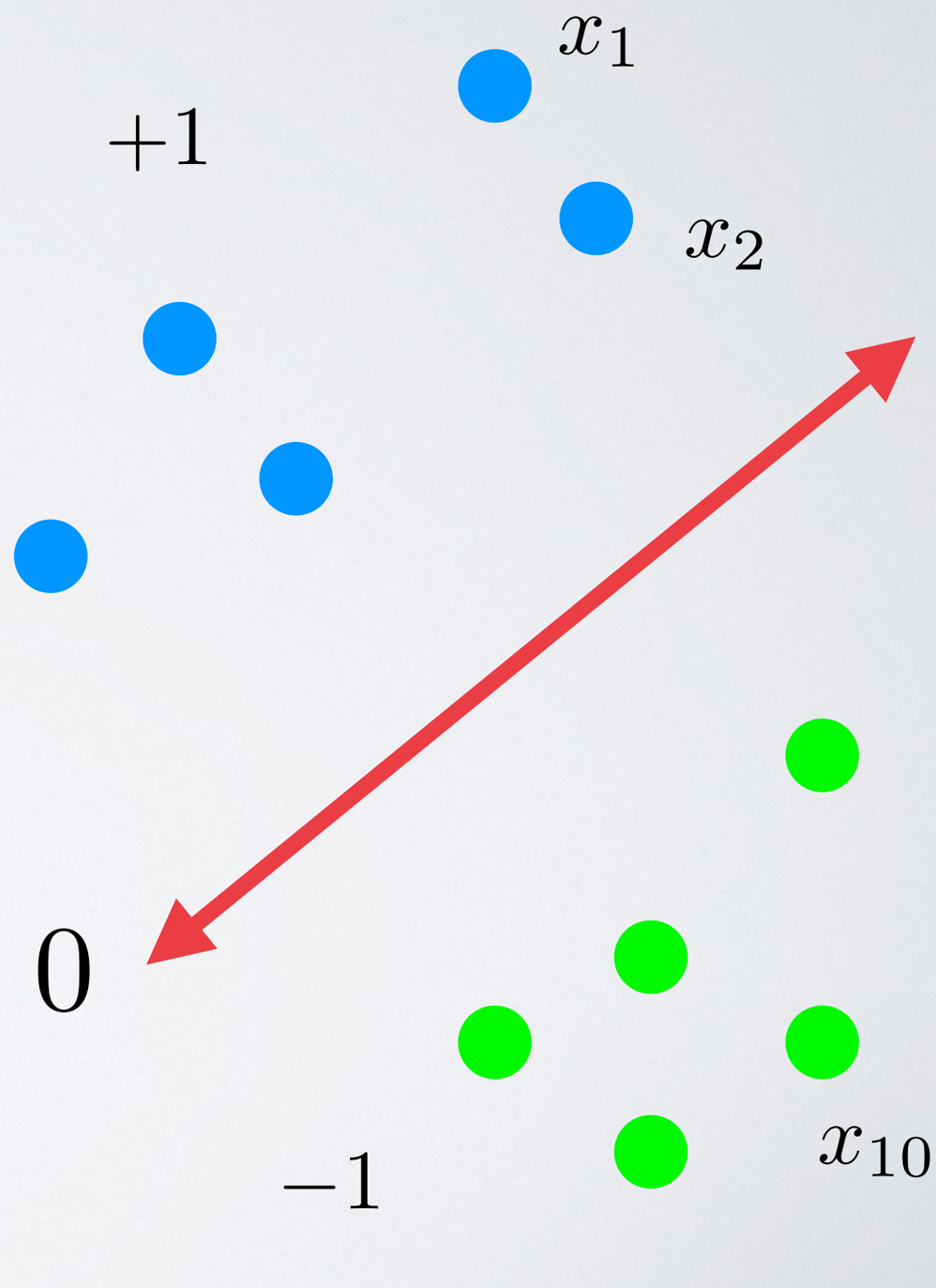


LINEAR CLASSIFIERS

goal

learn a line that separates two object classes

what line is best?

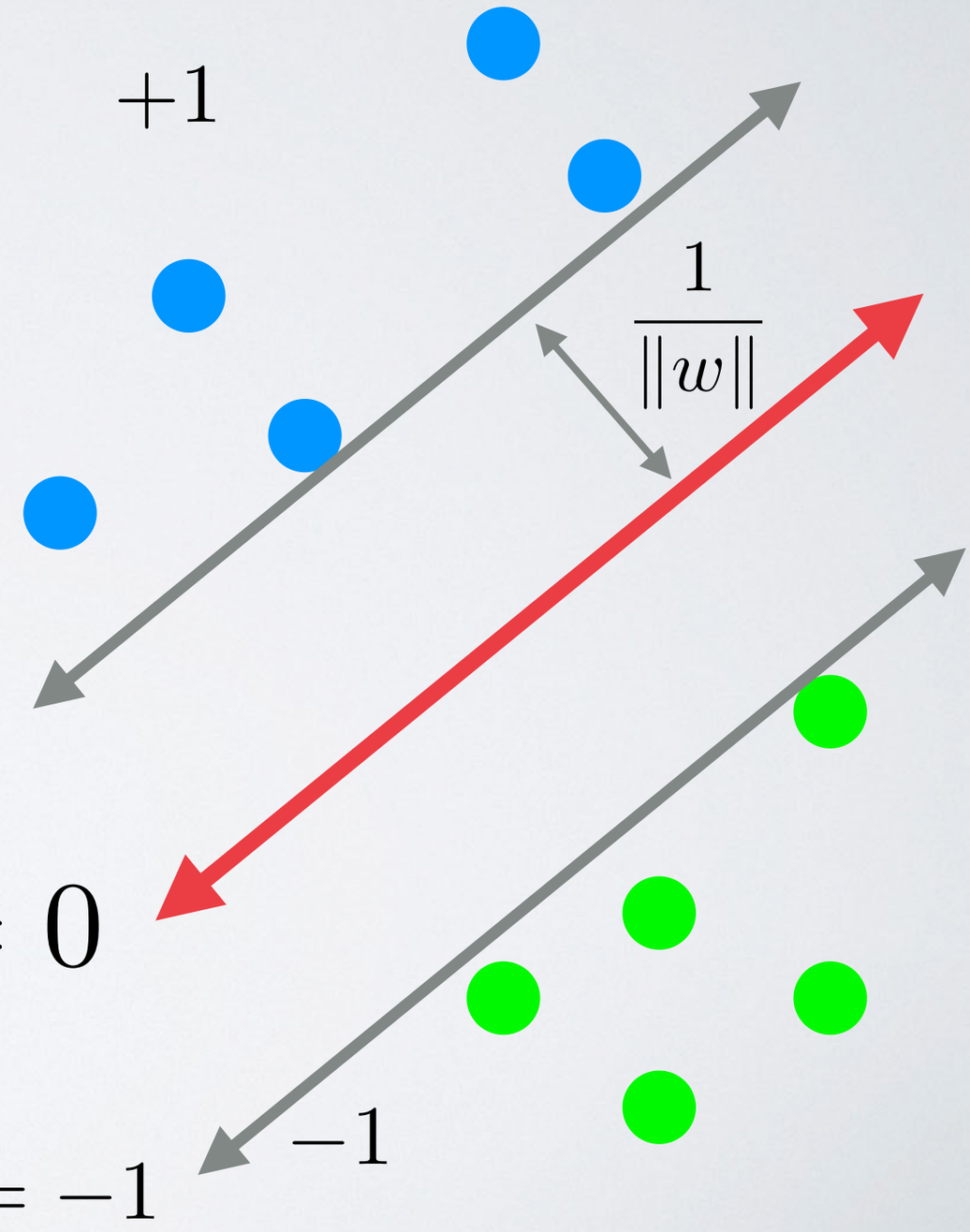


SUPPORT VECTOR MACHINE

SVM

choose line with maximum "margin"

$$\text{margin width} = \frac{1}{\|w\|}$$

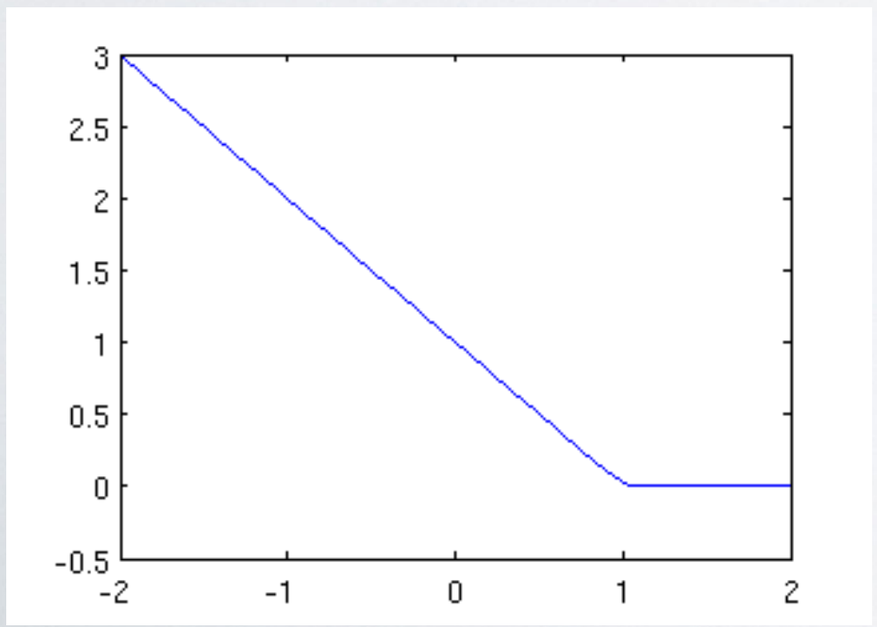


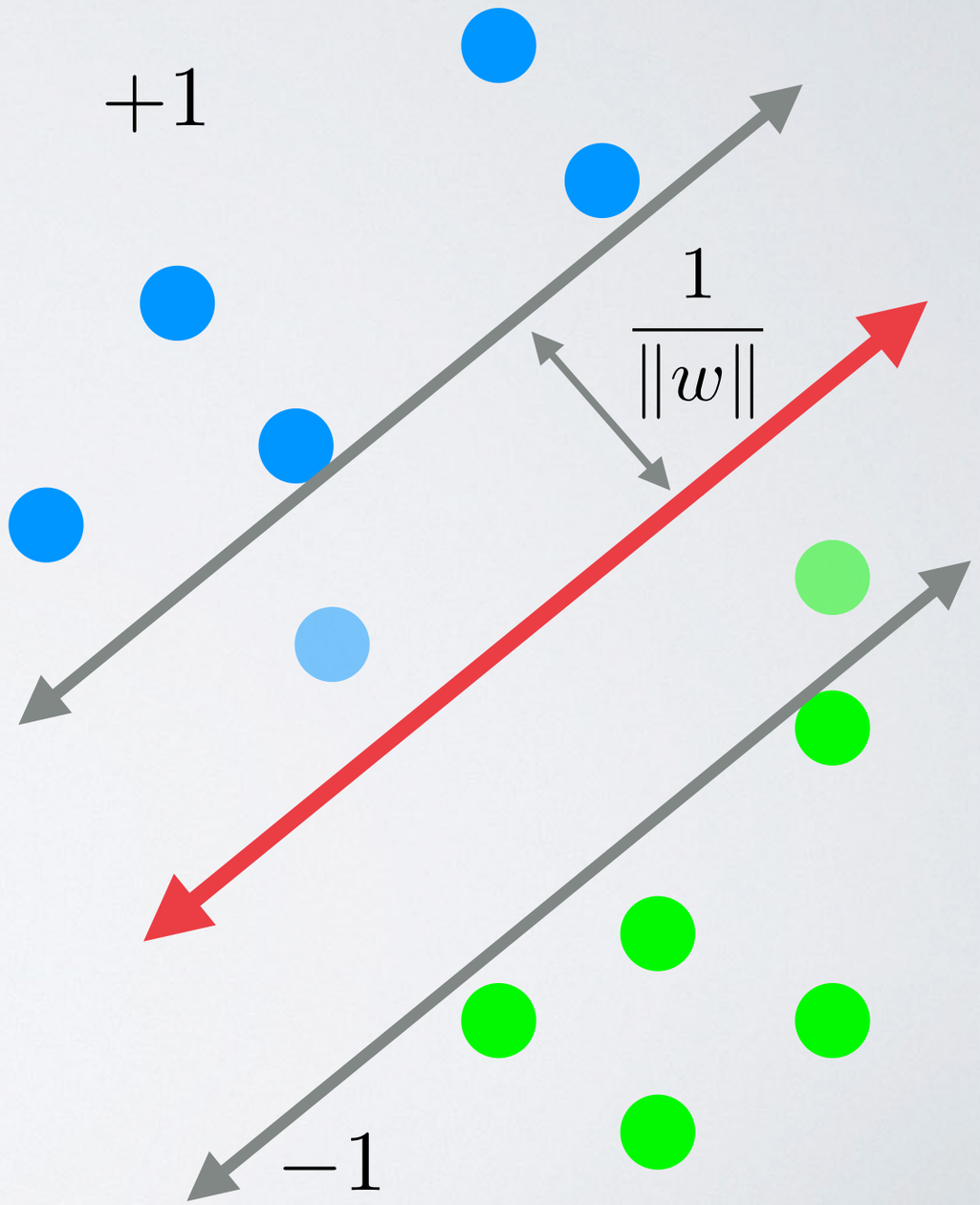
SUPPORT VECTOR MACHINE

hinge loss

penalize points that lie within the margin

$$\sum_i h(x_i^T w)$$





SUPPORT VECTOR MACHINE

hinge loss

penalize points that lie within the margin

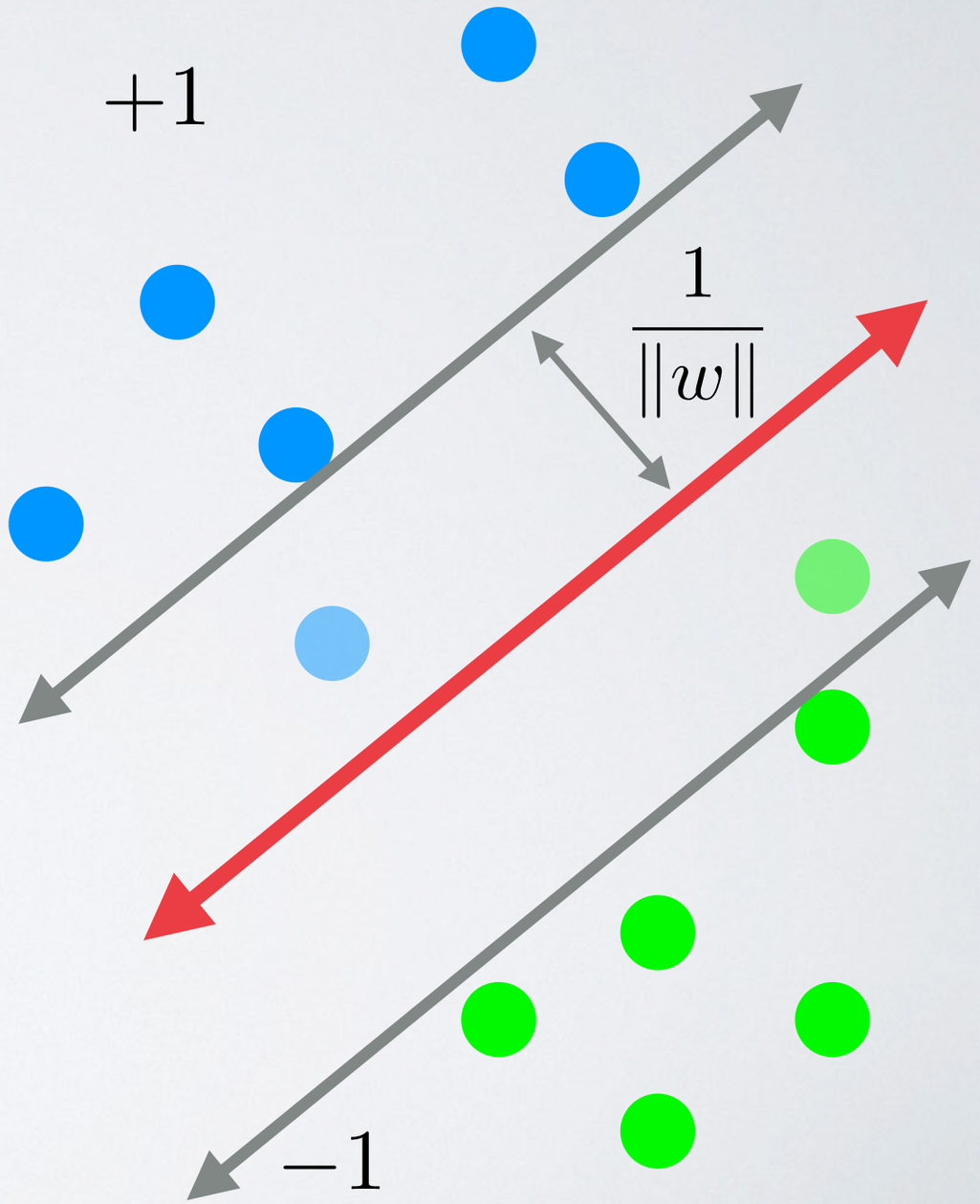
$$\sum_i h(x_i^T w)$$

For class -1 …

$$\sum_i h(-x_i^T w)$$

In general…

$$\sum_i h(y_i \cdot x_i^T w)$$



SUPPORT VECTOR MACHINE

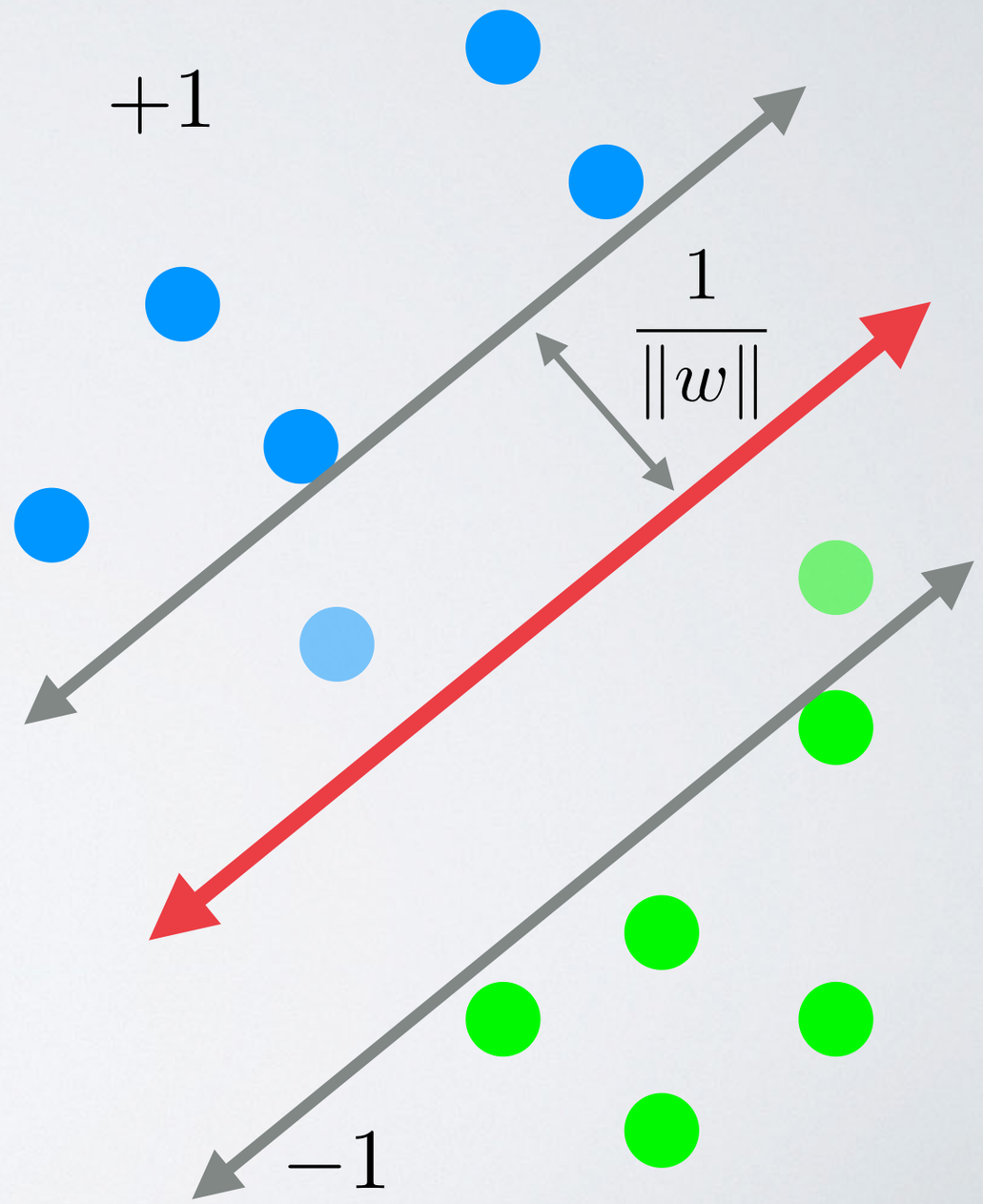
In general…

$$\sum_i h(y_i \cdot x_i^T w)$$

short-hand

$$h(\hat{X}w)$$

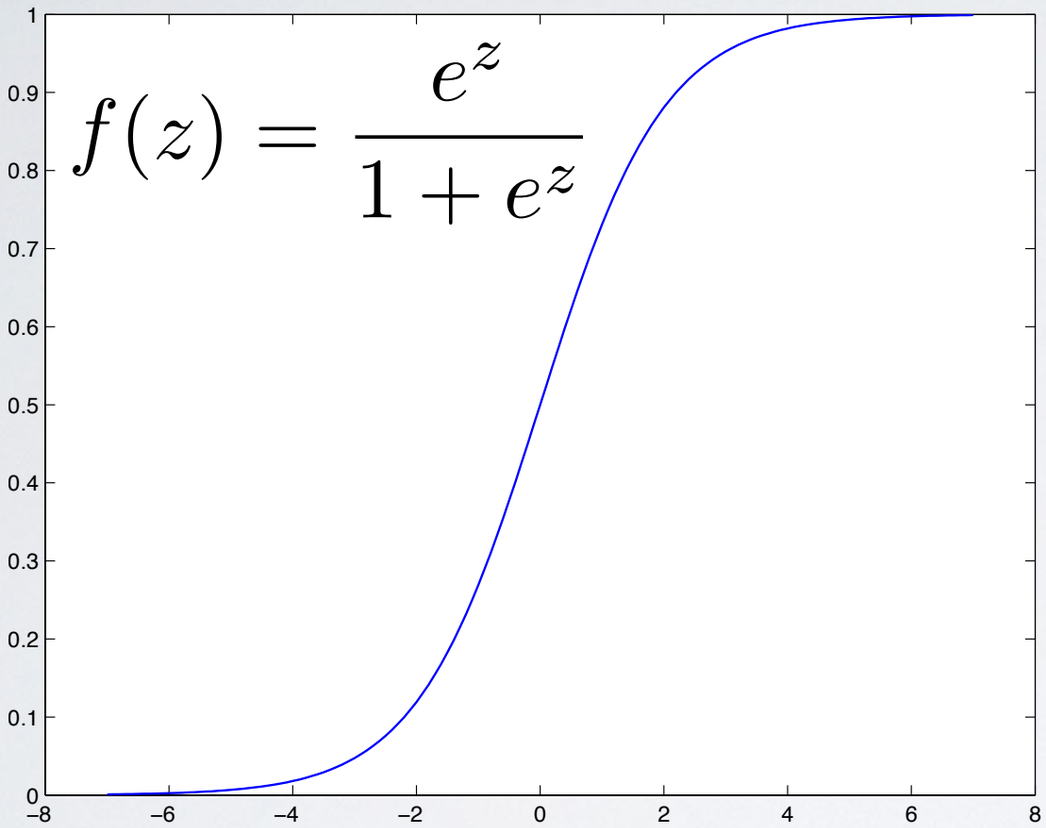
margin width $= \dfrac{1}{\|w\|}$

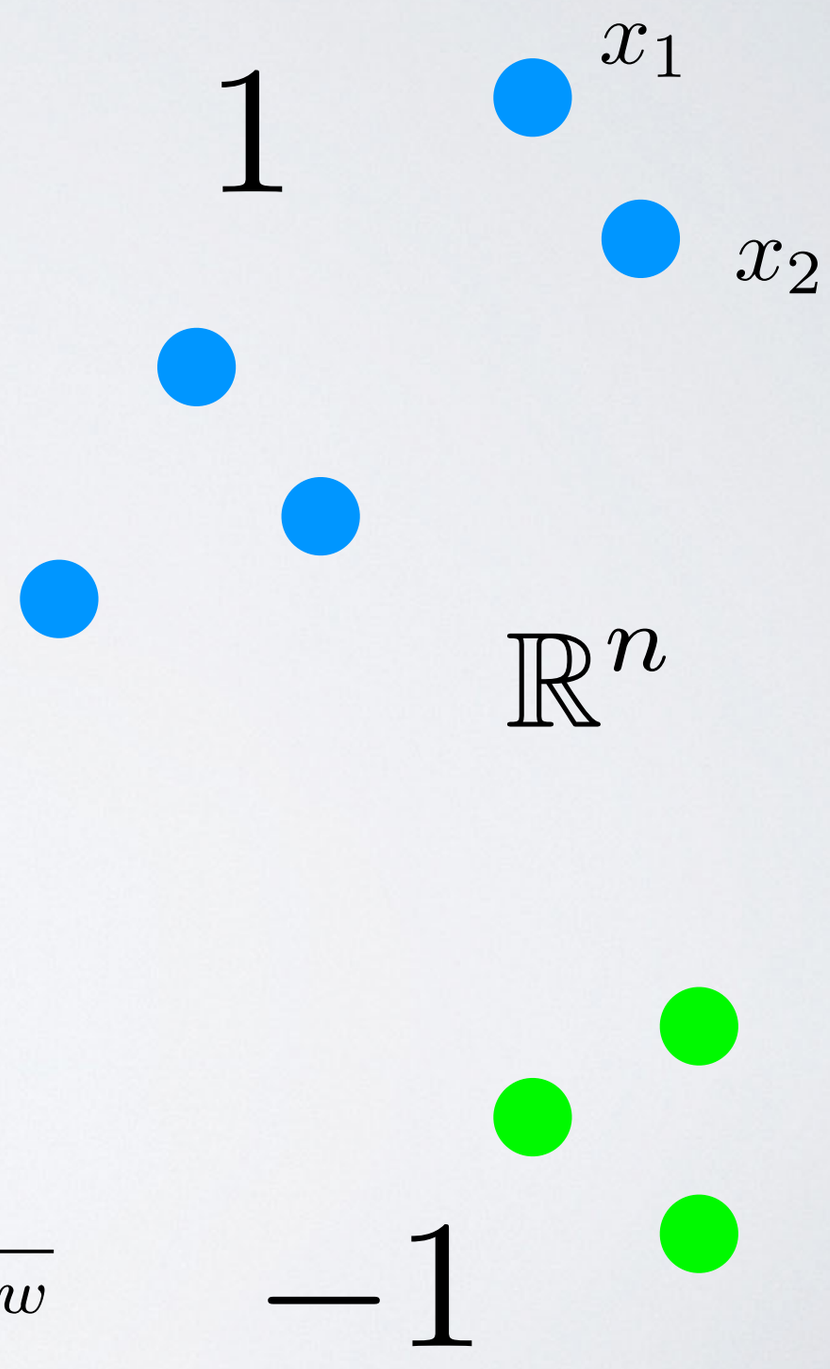


combined objective

minimize $\quad \dfrac{1}{2}\|w\|^2 + h(\hat{X}w)$

LOGISTIC REGRESSION





$$\mathbb{P}[y=1] = \frac{e^{x^T w}}{1+e^{x^T w}}$$

$$\mathbb{P}[y=-1] = 1 - \frac{e^{x^T w}}{1+e^{x^T w}} = \frac{1}{1+e^{x^T w}}$$

LIKELIHOOD FUNCTIONS

probability of observing label given data

case: observed y=1

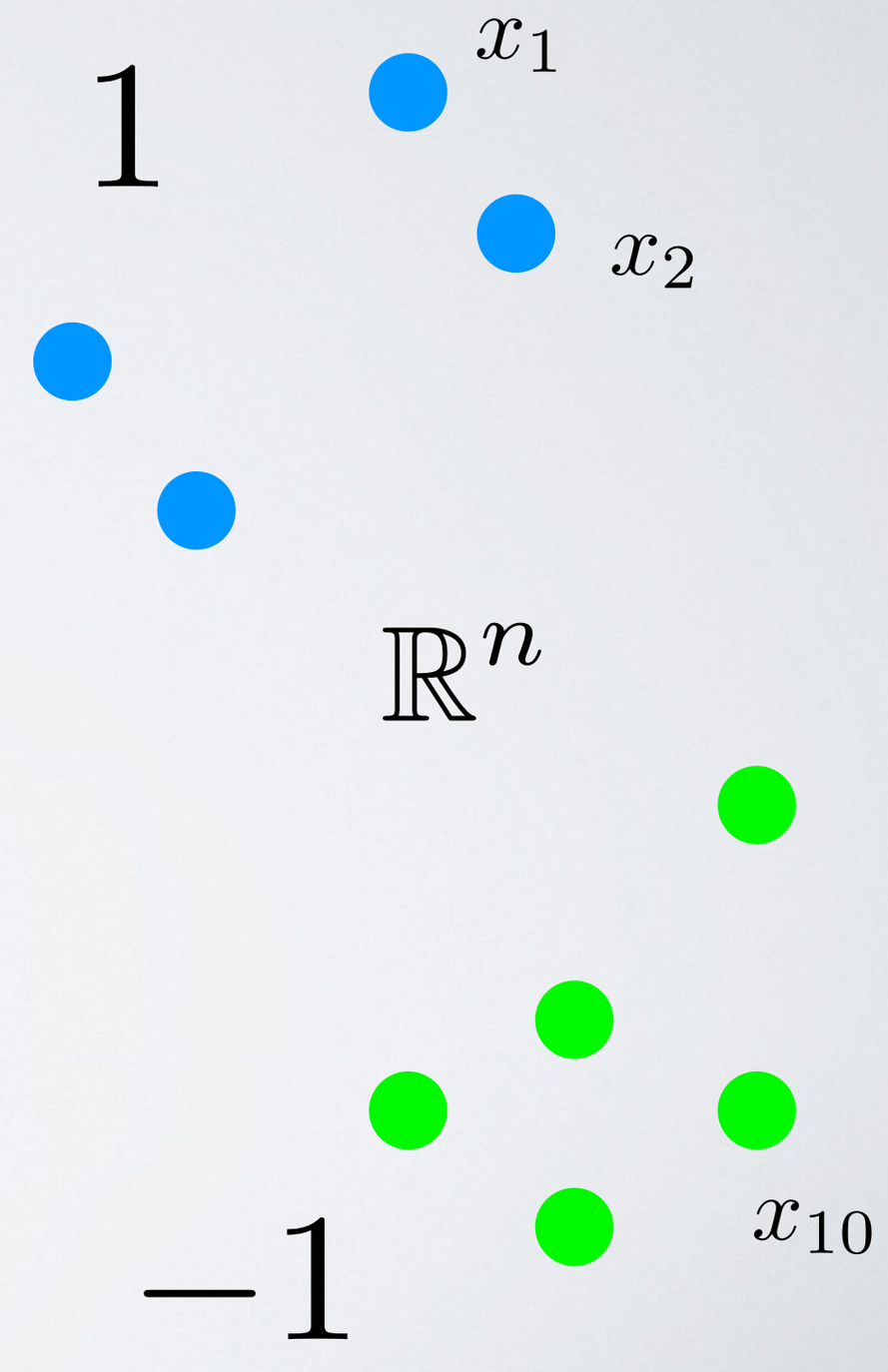
$$\mathbb{P}[y=1] = \frac{e^{x^T w}}{1+e^{x^T w}}$$

$$NLL(x) = \log(1+e^{x^T w}) - x^T w$$

case: observed y=-1

$$\mathbb{P}[y=-1] = 1 - \frac{e^{x^T w}}{1+e^{x^T w}} = \frac{1}{1+e^{x^T w}}$$

$$NLL(x) = \log(1+e^{x^T w})$$



LIKELIHOOD FUNCTIONS

probability of observing label given data

case: observed y=1

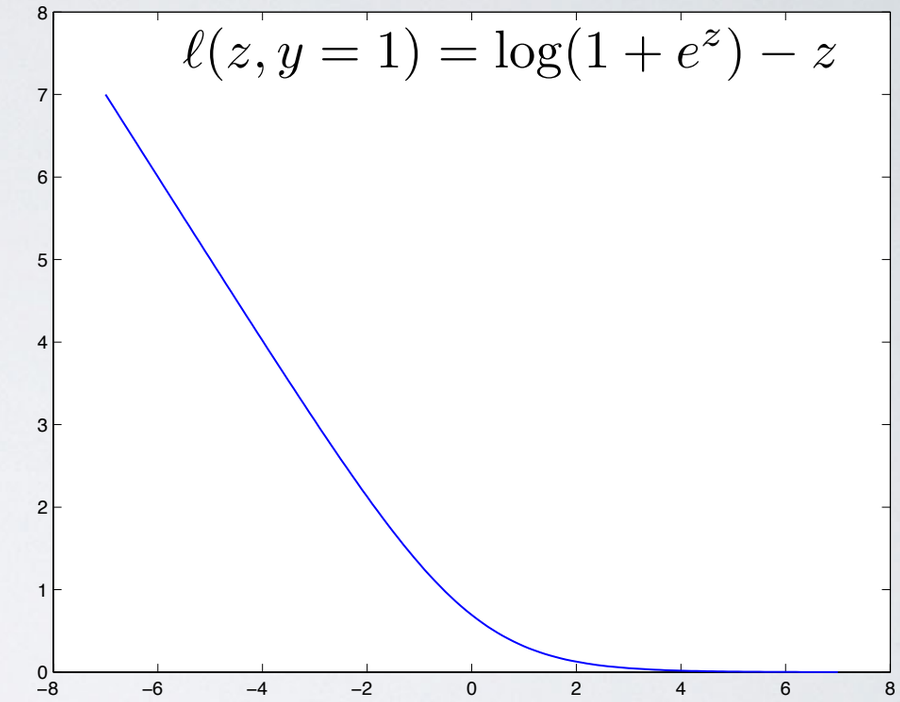
$$\mathbb{P}[y=1] = \frac{e^{x^T w}}{1+e^{x^T w}}$$

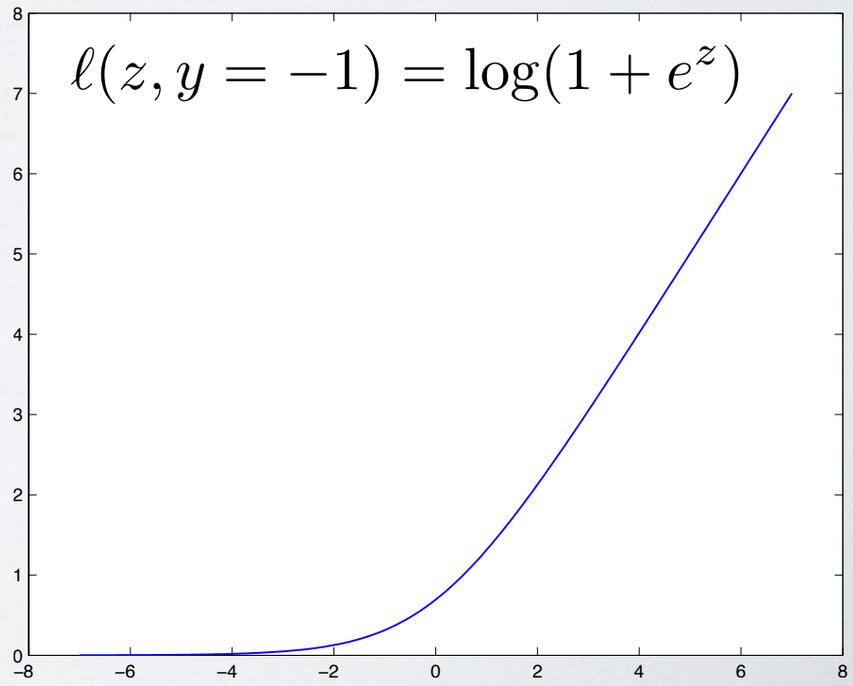
$$NLL(x) = \log(1+e^{x^T w}) - x^T w$$

case: observed y=-1

$$\mathbb{P}[y=-1] = 1 - \frac{e^{x^T w}}{1+e^{x^T w}} = \frac{1}{1+e^{x^T w}}$$

$$NLL(x) = \log(1+e^{x^T w})$$



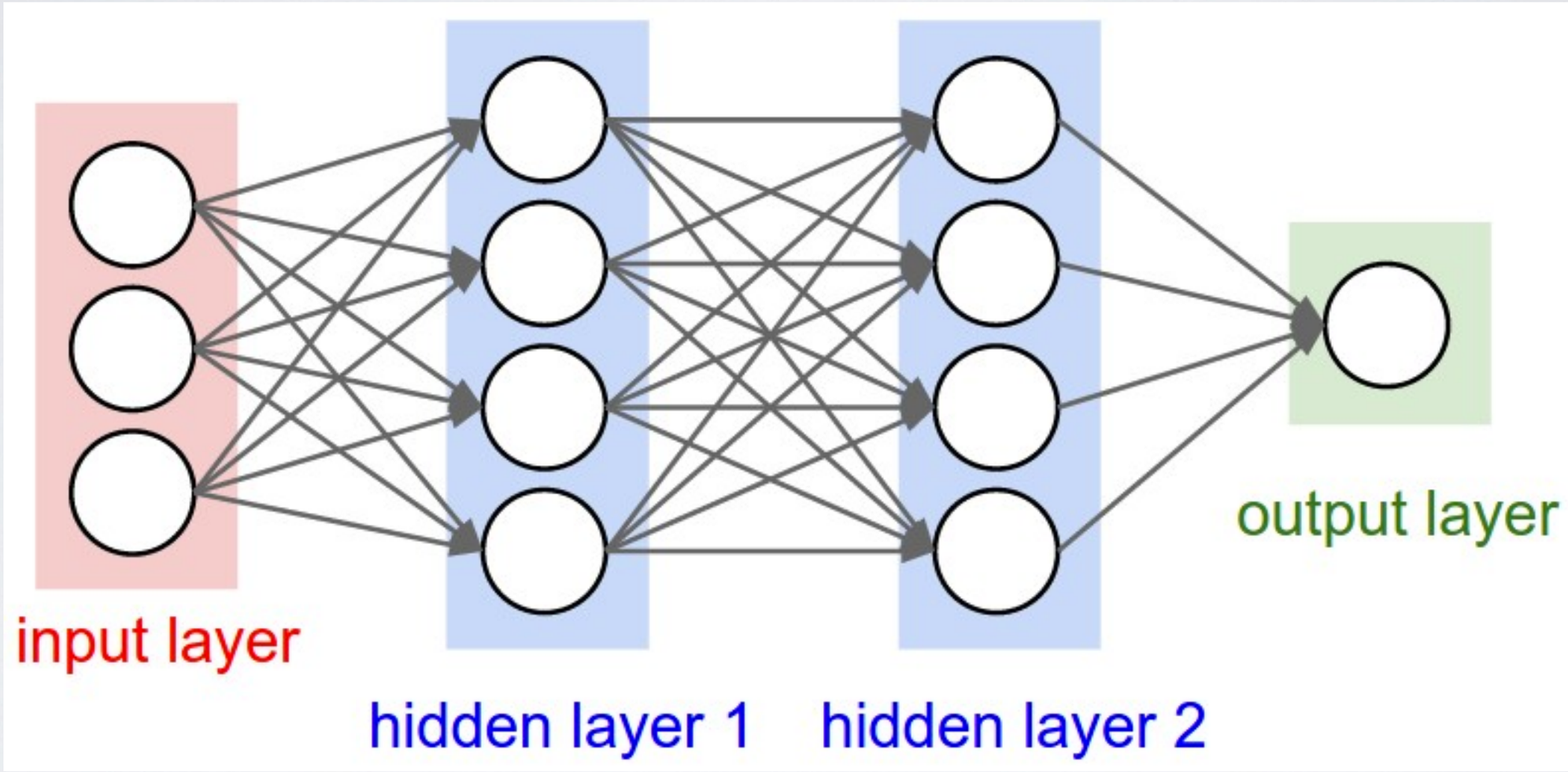


NEURAL NET BASICS

NEURAL NETWORKS

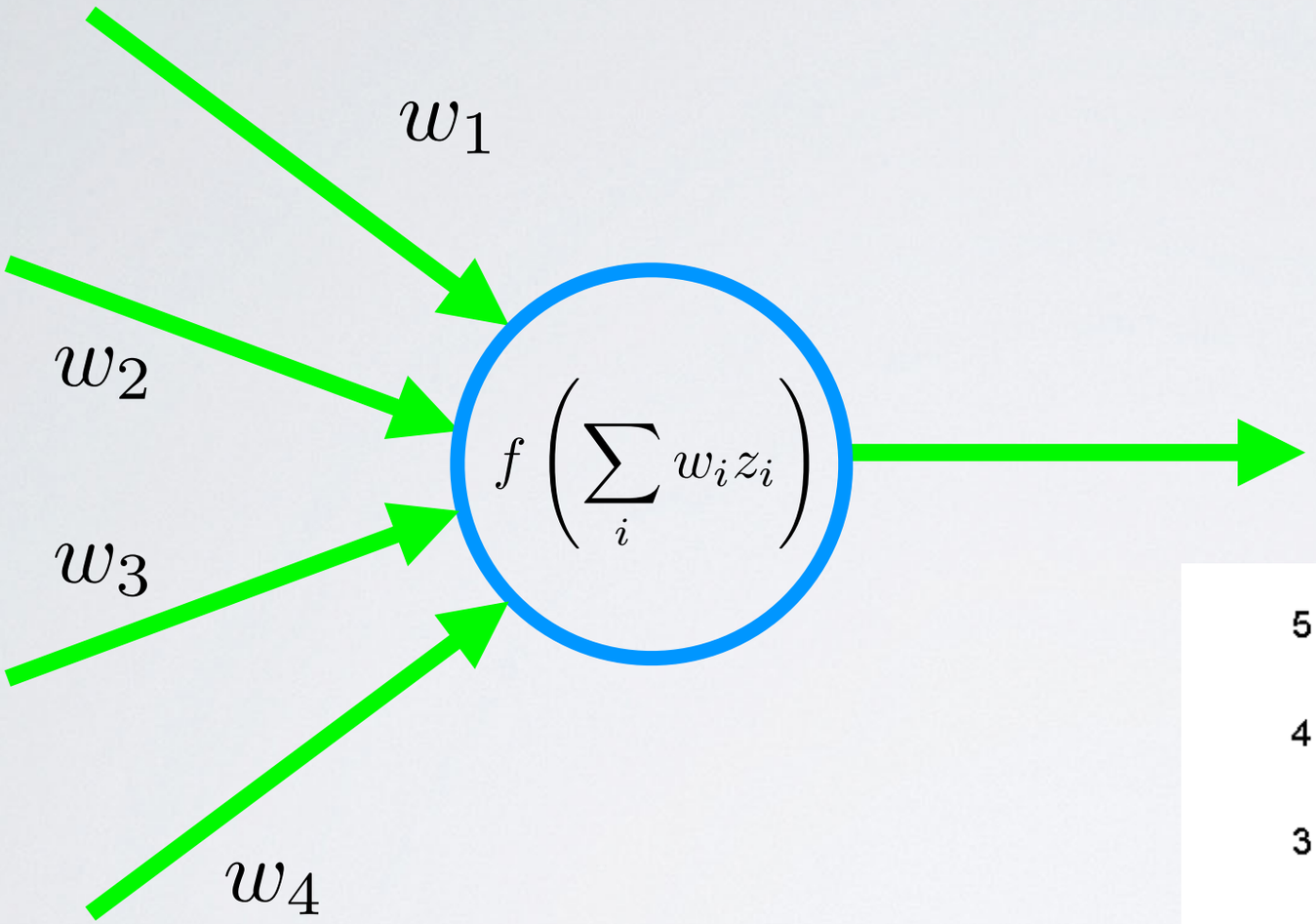
W_1 W_2 W_3

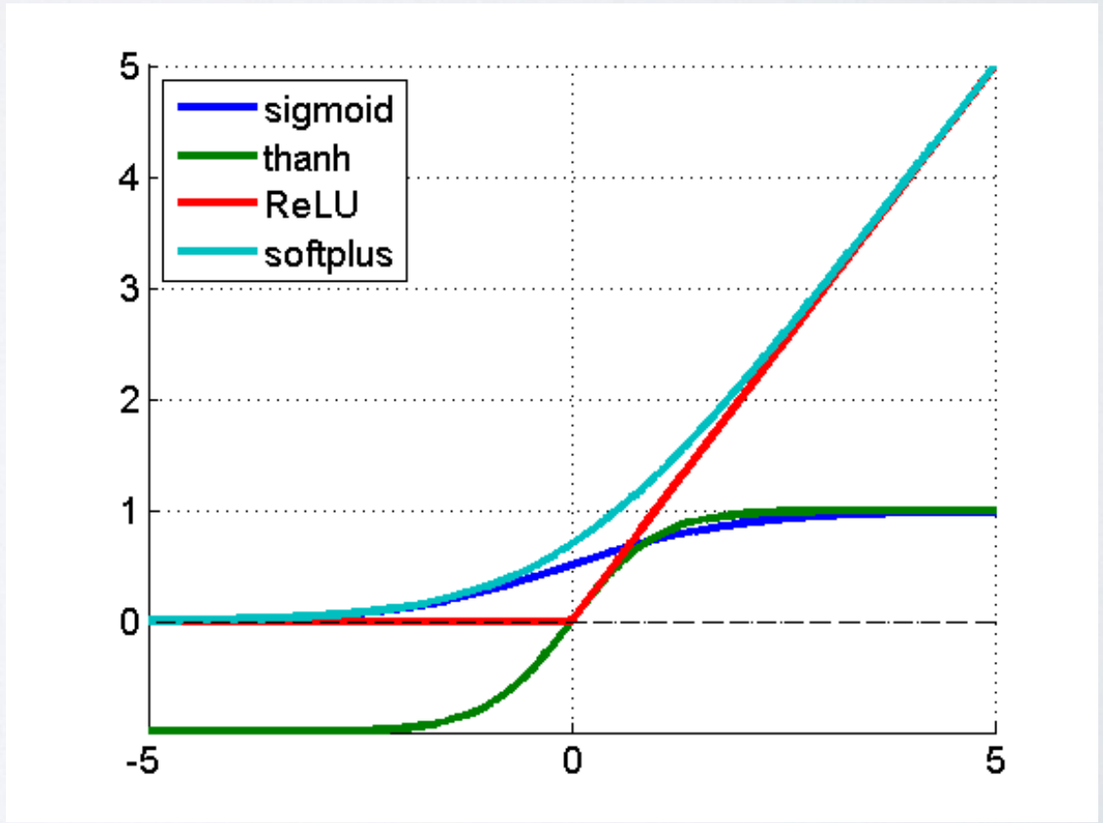
x_i



y_i

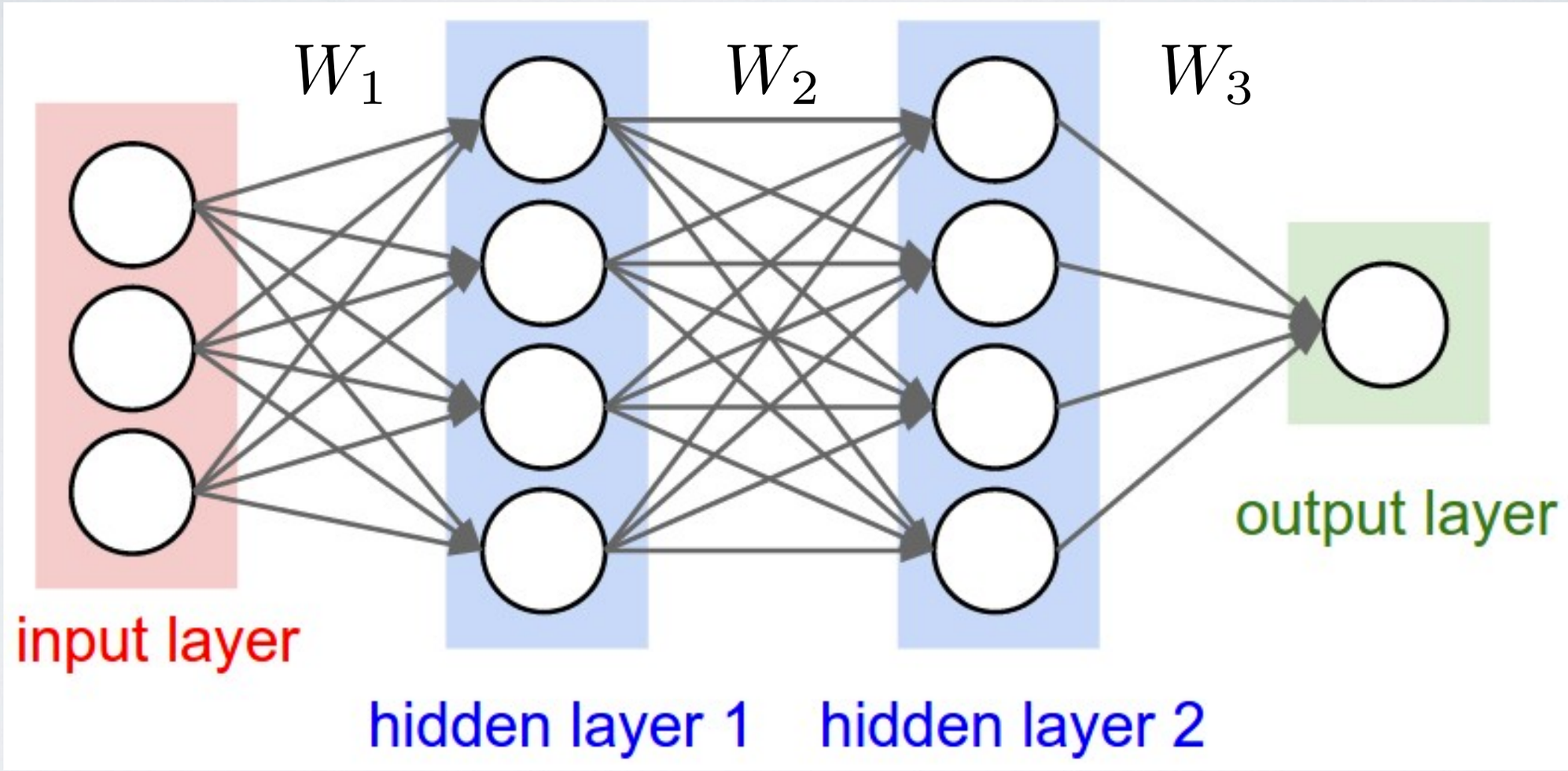
ARTIFICIAL NEURON





NEURAL NETWORKS

x_i



y_i

$$f(f(f(x_i W_1)W_2)W_3) = y_i$$

TRAINING

$$f(f(f(x_i W_1)W_2)W_3) = y_i$$

least-squares loss

$$\ell(z, y) = (z - y)^2$$

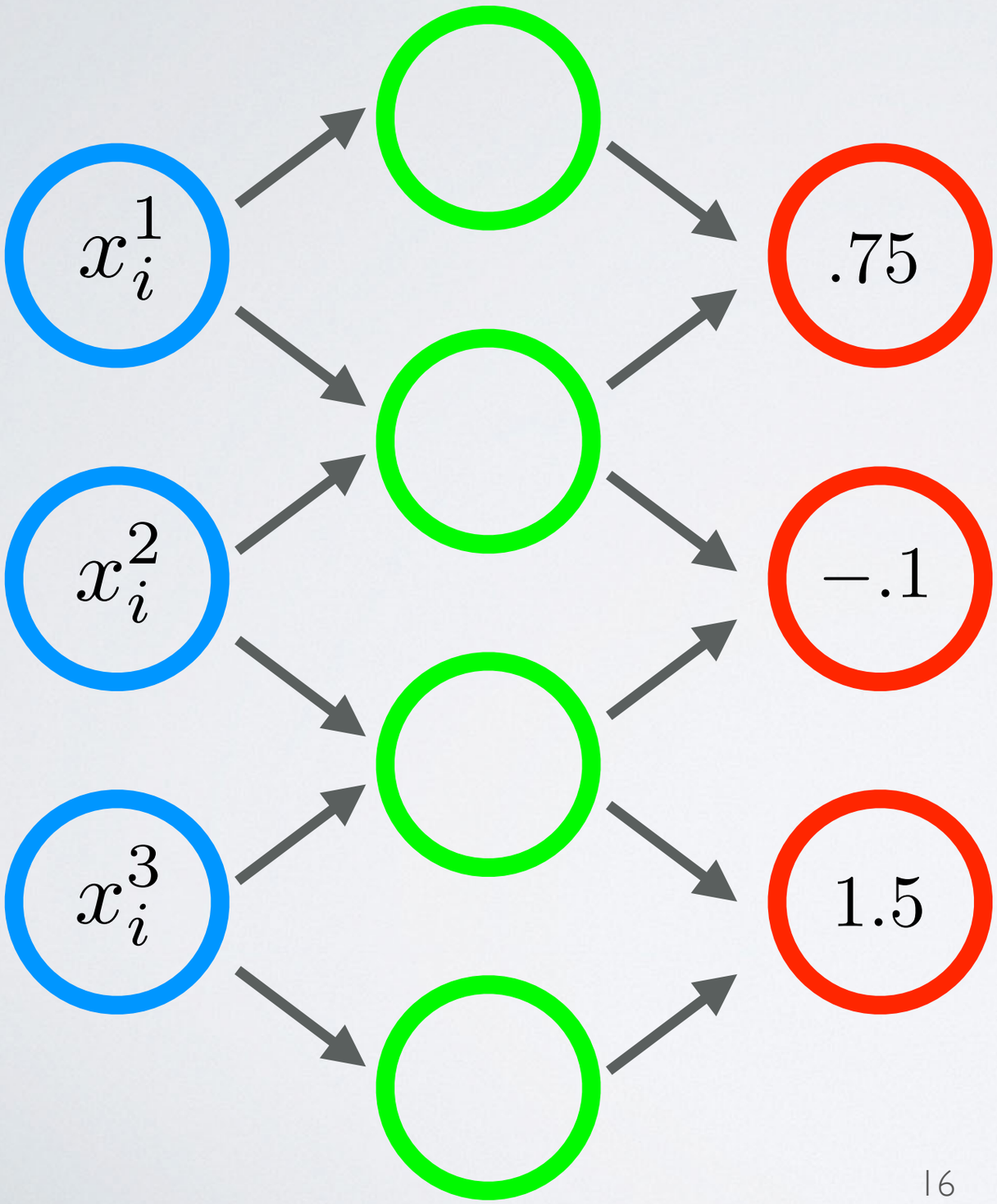
$$\min_{W_1, W_2, W_3} \sum_i \| f(f(f(x_i W_1)W_2)W_3) - y_i \|^2$$

“cross entropy”/log-likelihood loss

$$l(z, y) = \begin{cases} -\log(z), & y = 1 \\ -\log(1 - z), & y = -1 \end{cases}$$

$$\min_{W_1, W_2, W_3} \sum_i \ell(f(f(f(x_i W_1)W_2)W_3), y_i)$$

MULTI-CLASS NETWORK



interpretable?

target label

0

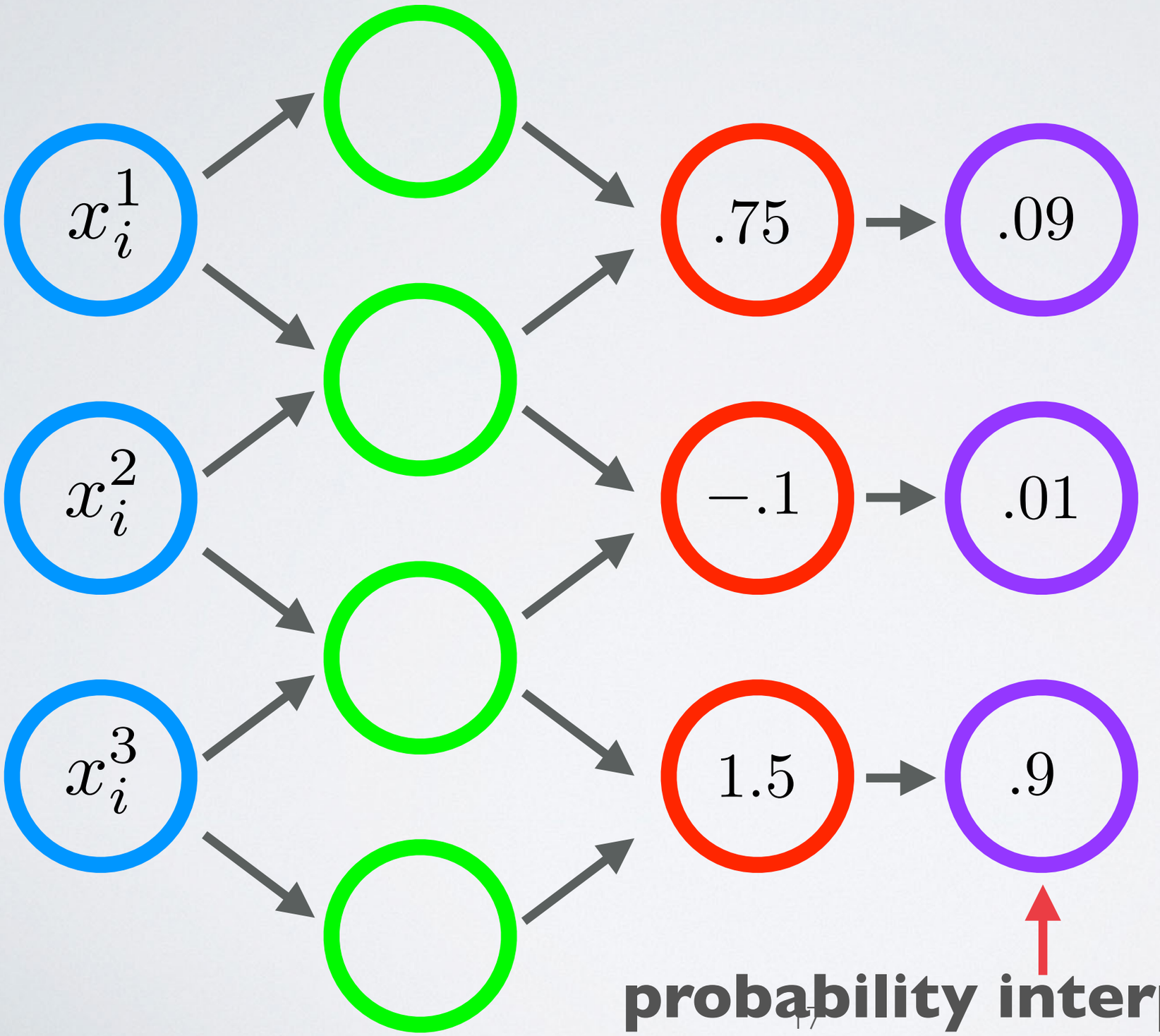
0

1

y_i

“ones hot” representation

MULTI-CLASS NETWORK



target label

0

0

1

“ones hot” representation

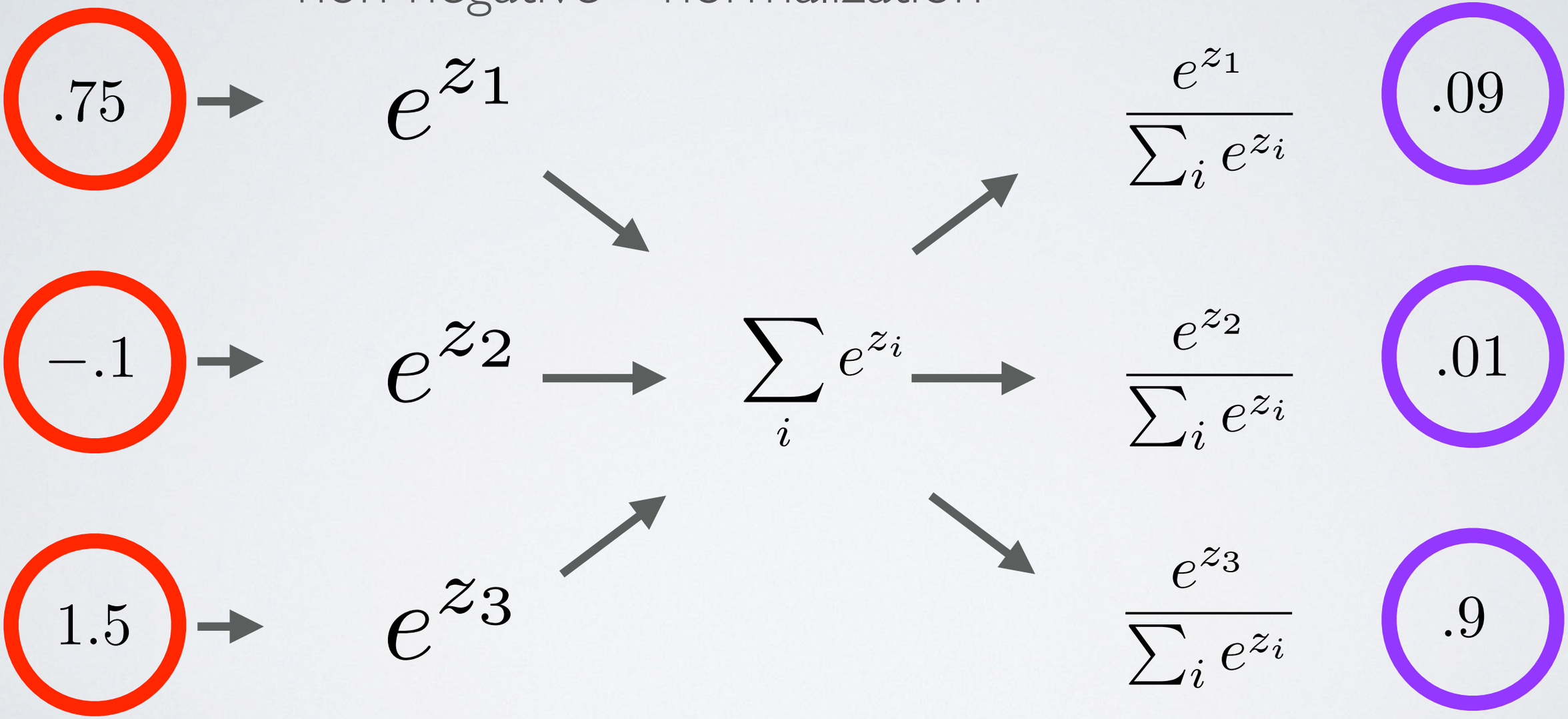
probability interpretation

SOFTMAX

make non-negative

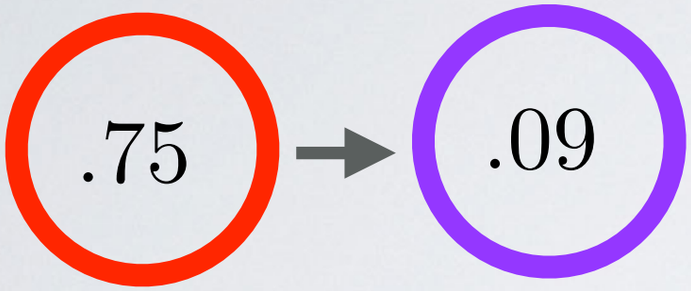
compute normalization

probabilities



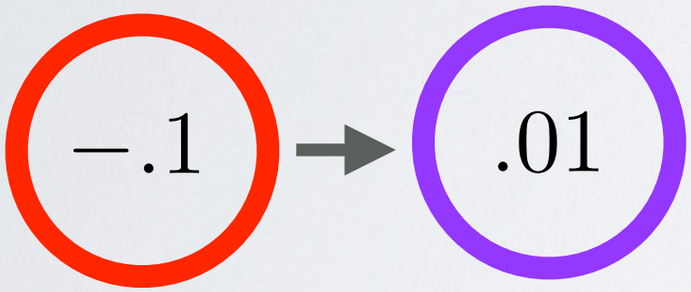
CROSS-ENTROPY LOSS

target label



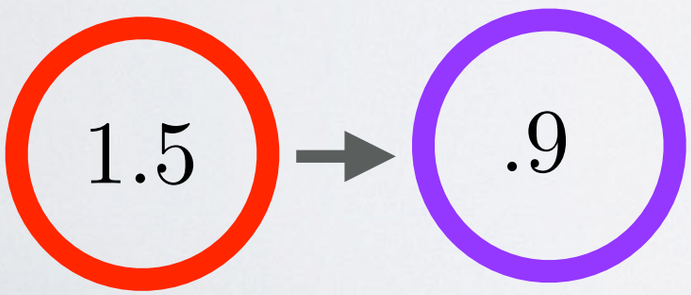
0 y_i^1

"cross entropy loss"
is negative log likelihood



0 y_i^2

$$\ell(x, y) = -\sum_k y^k \log(x^k)$$



1 y_i^3

NLL= $-\log(.9)$

y_i

BUILDING MORE COMPLEX NETS

CONVOLUTIONAL NET

x

w_1

w_i

y

One's hot label

.05	.1	.9	.04
ball	car	hat	

grumpy cat

$$f(x; w) \rightarrow y$$

image weights

CONVOLUTIONAL NET

x

w_1

w_i

y

One's hot label

.05 .1 .9 .04

ball car hat

grumpy cat

Three ingredients

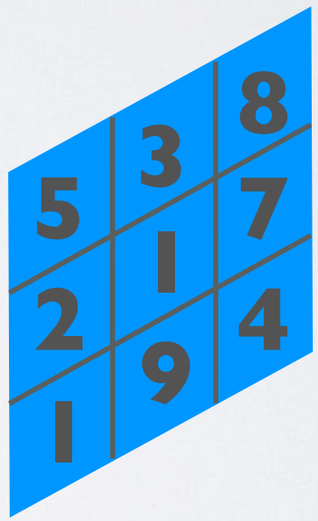
- convolutions
- non-linearities
- skip connections

CONVOLUTION

Input image x

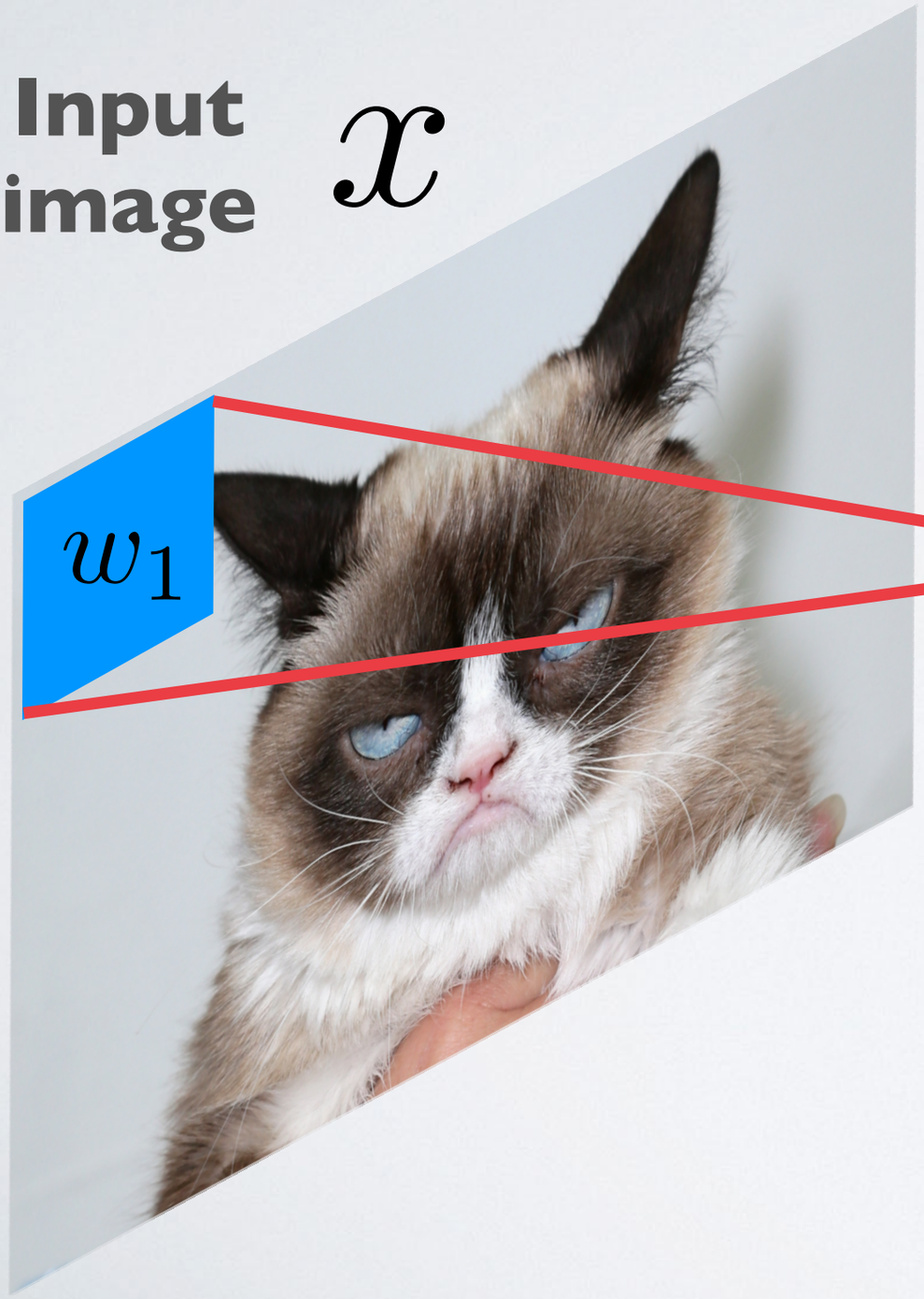
Convolutional filter

w_1



CONVOLUTION

Input image x



CONVOLUTION

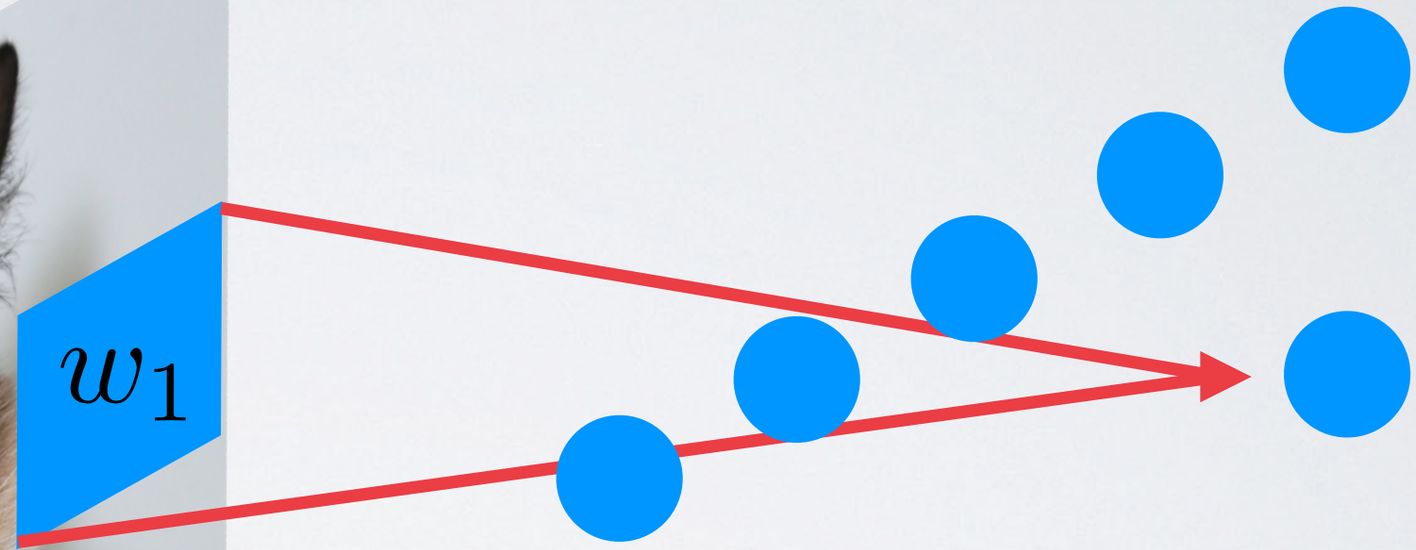
Input image x

w_1

CONVOLUTION

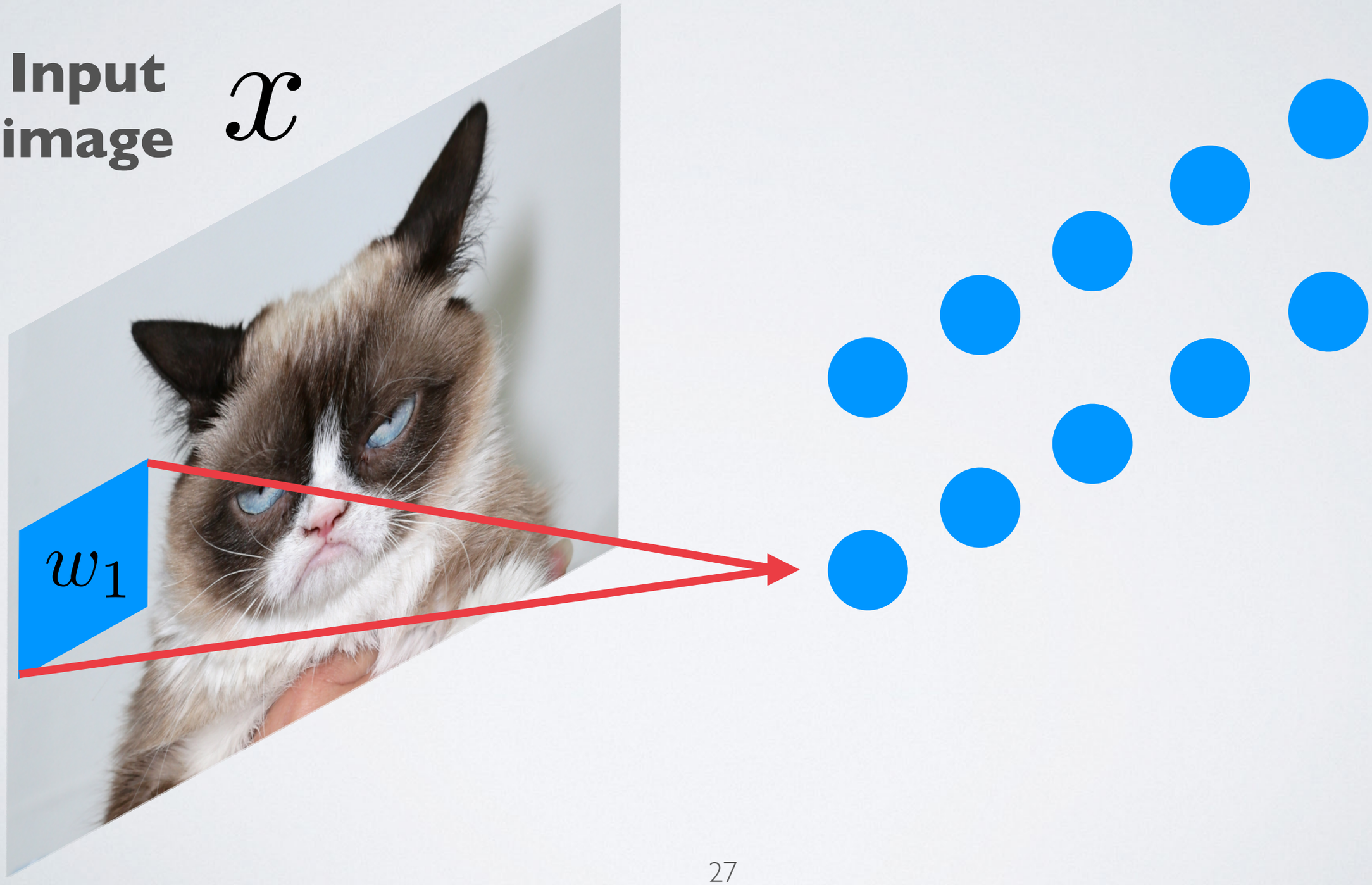
Input image x

w_1



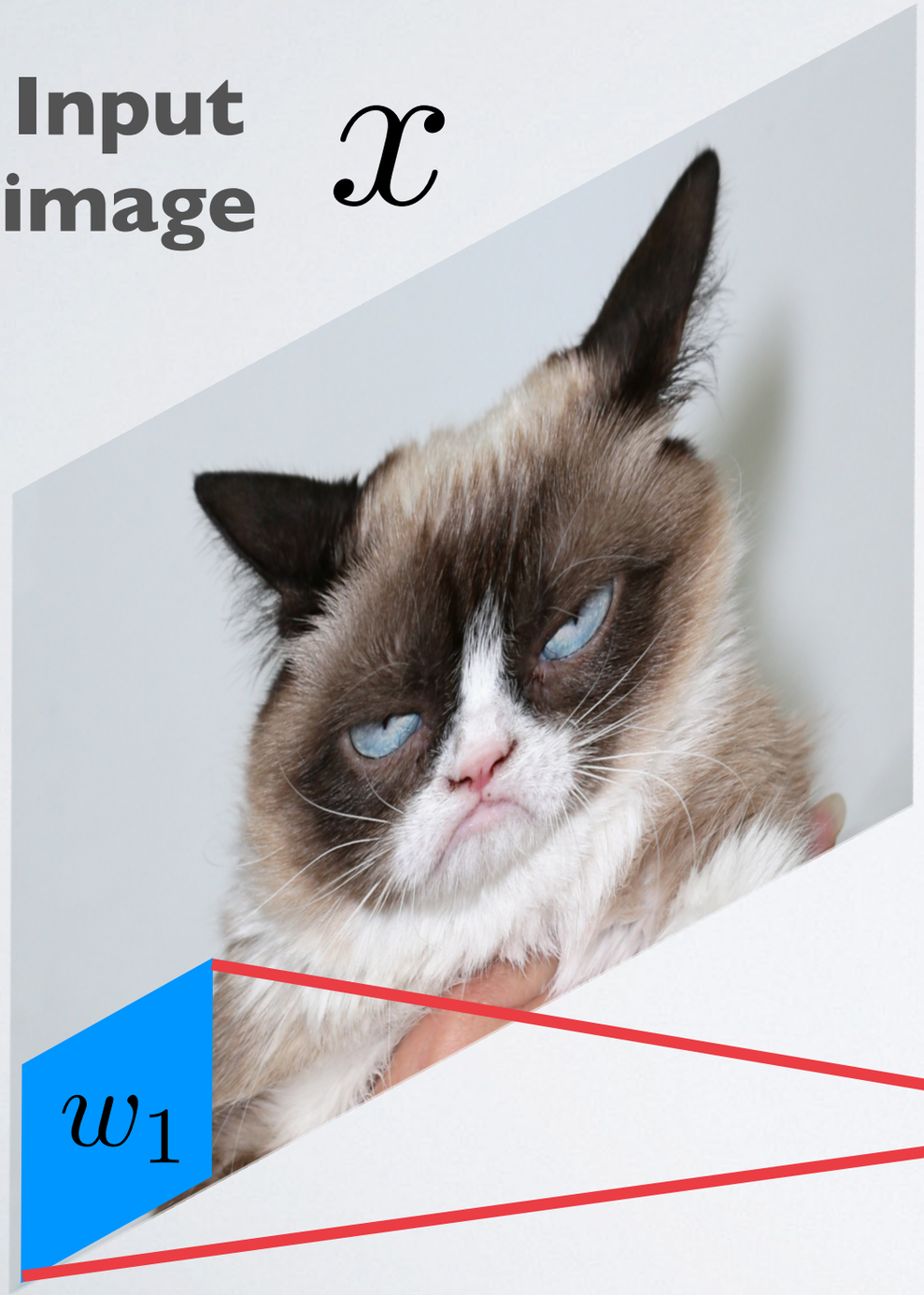
CONVOLUTION

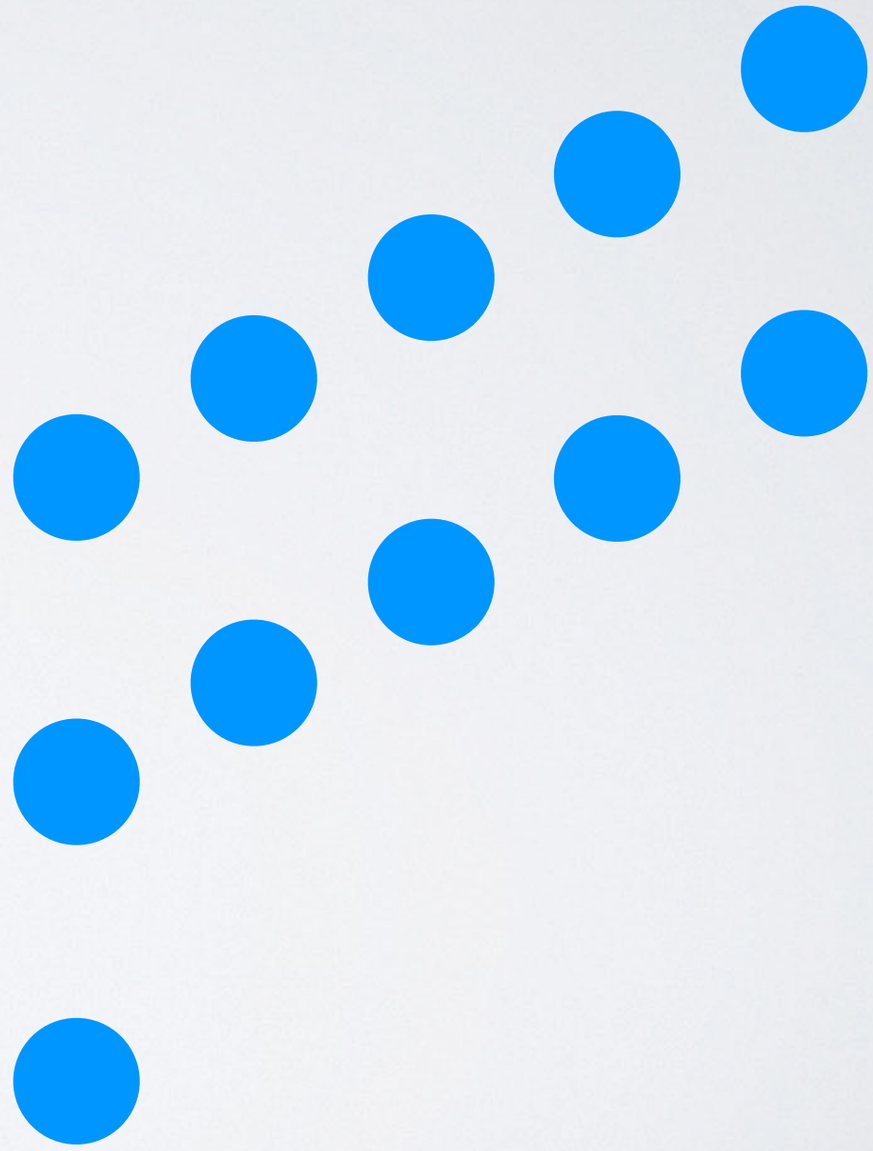
Input image x



CONVOLUTION

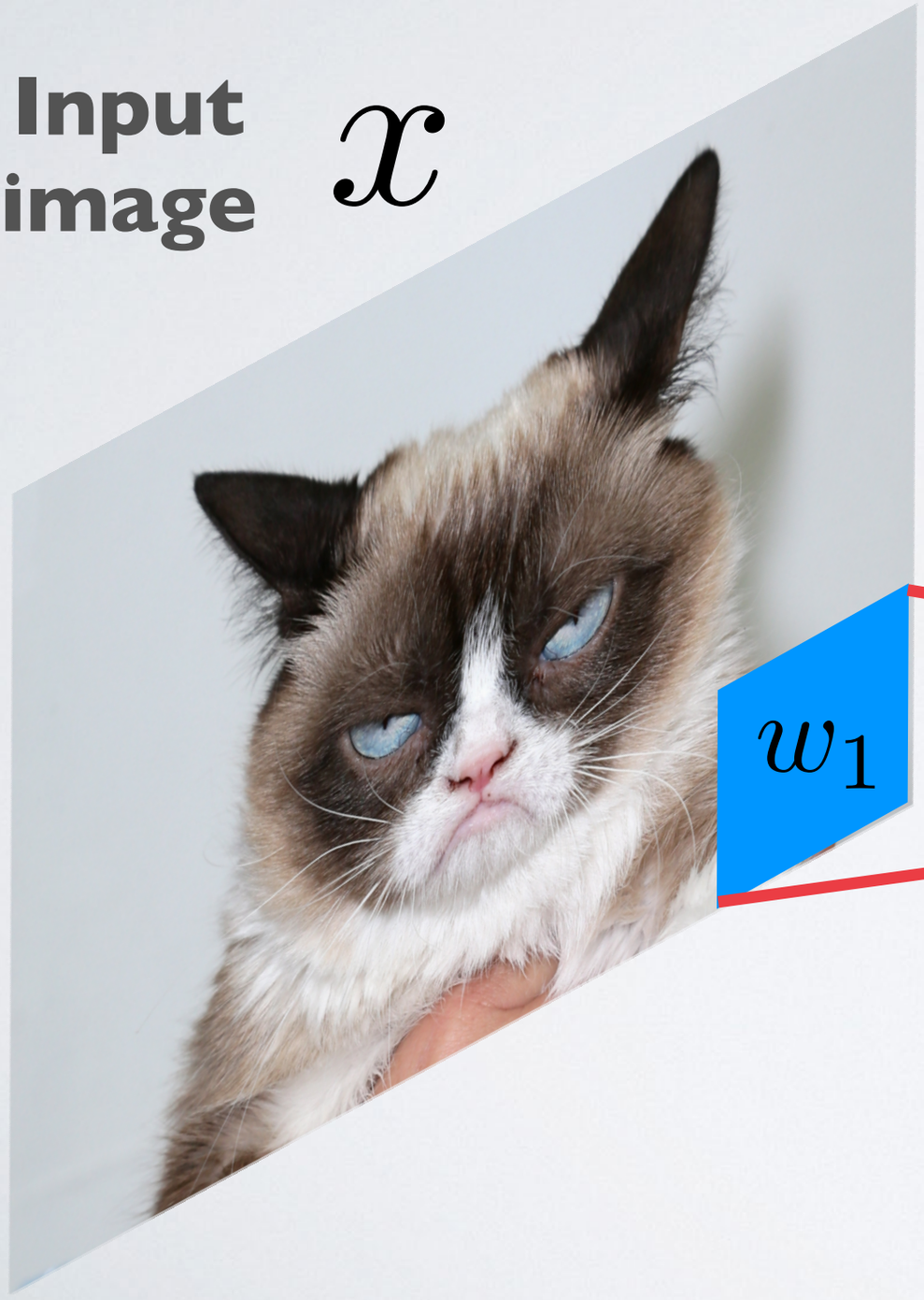
Input image x



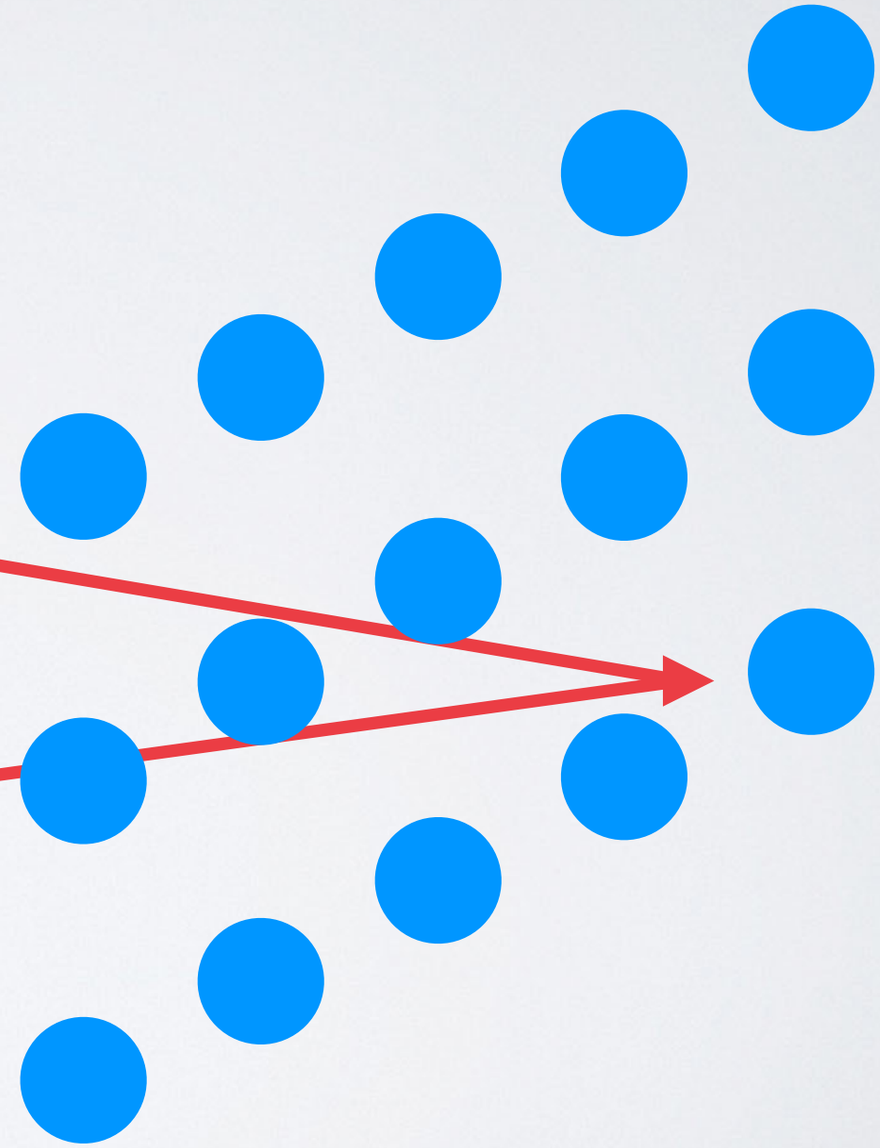


CONVOLUTION

Input image x

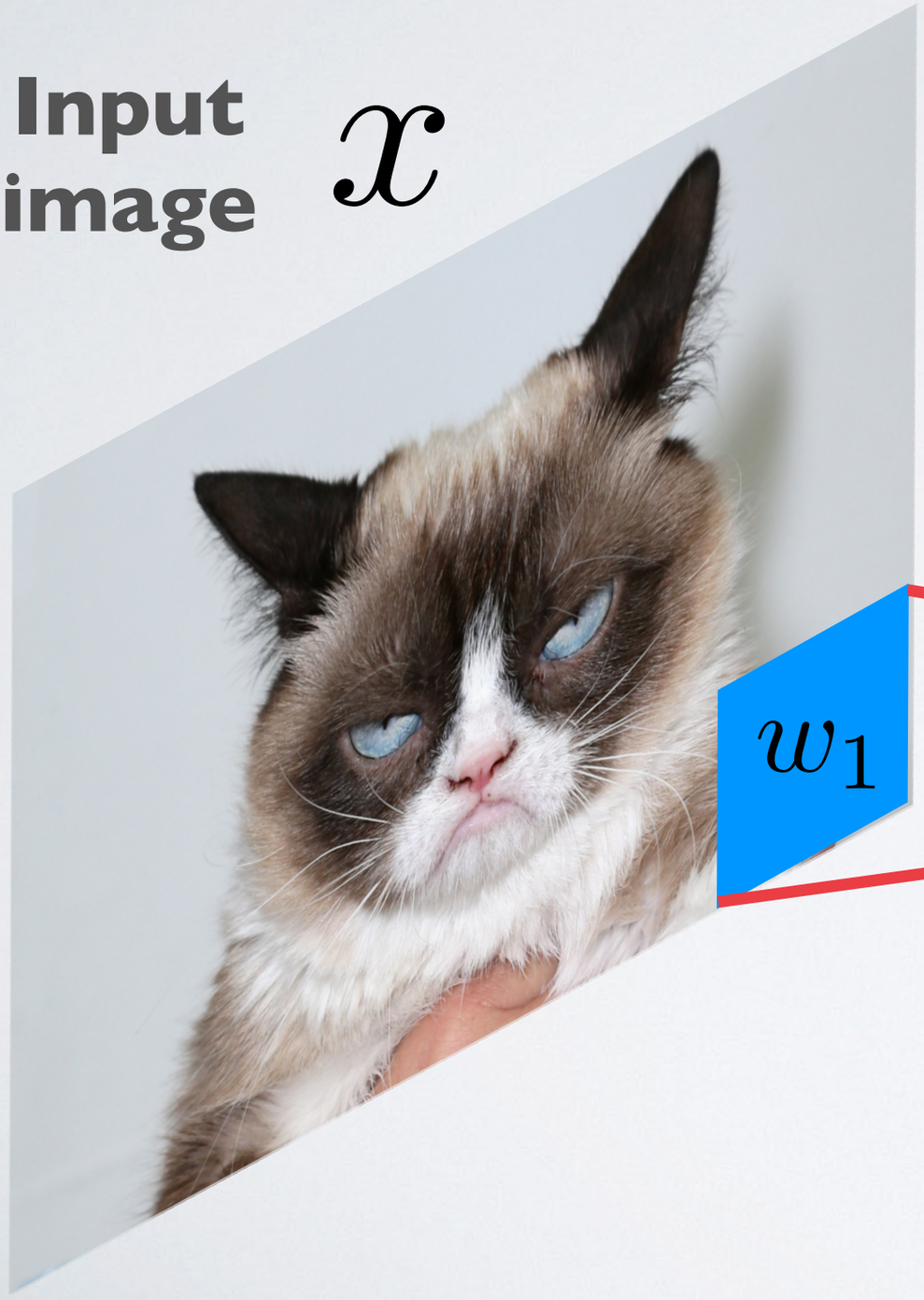


w_1



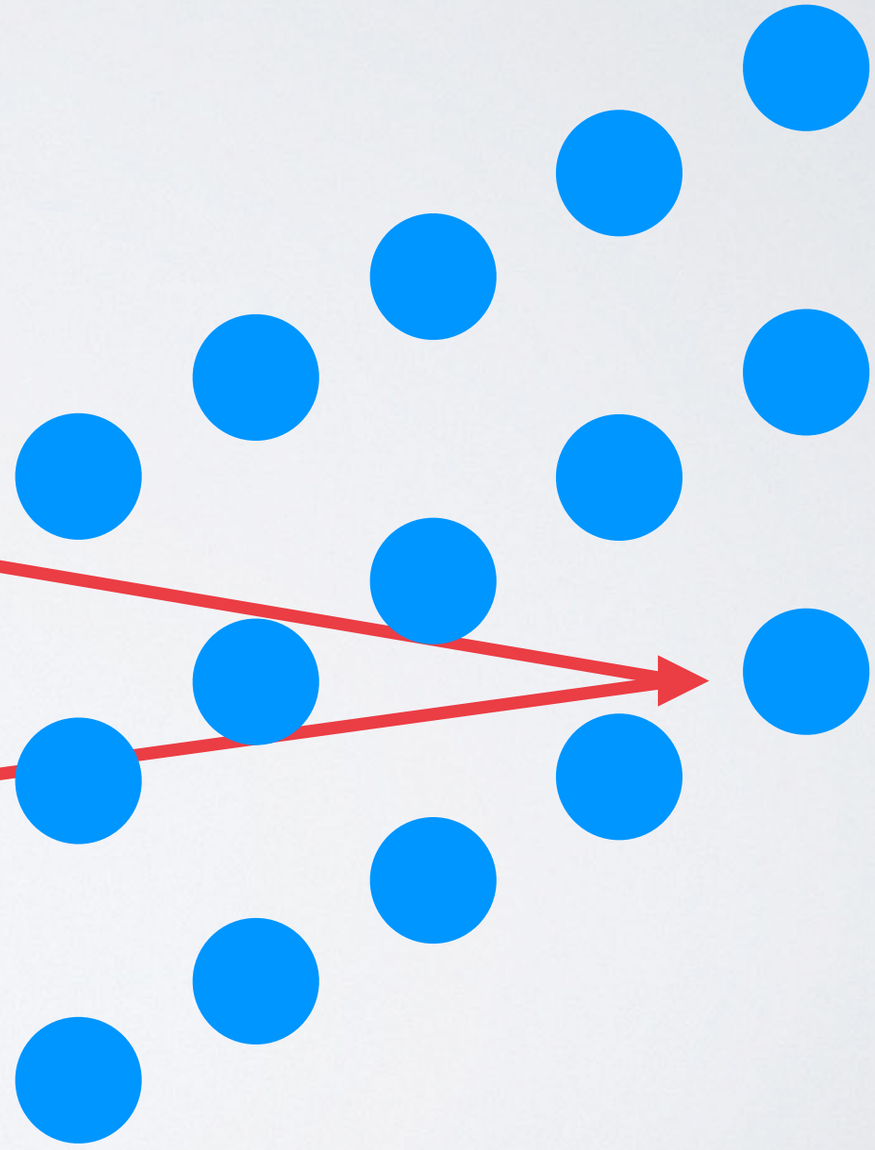
CONVOLUTION

Input image x



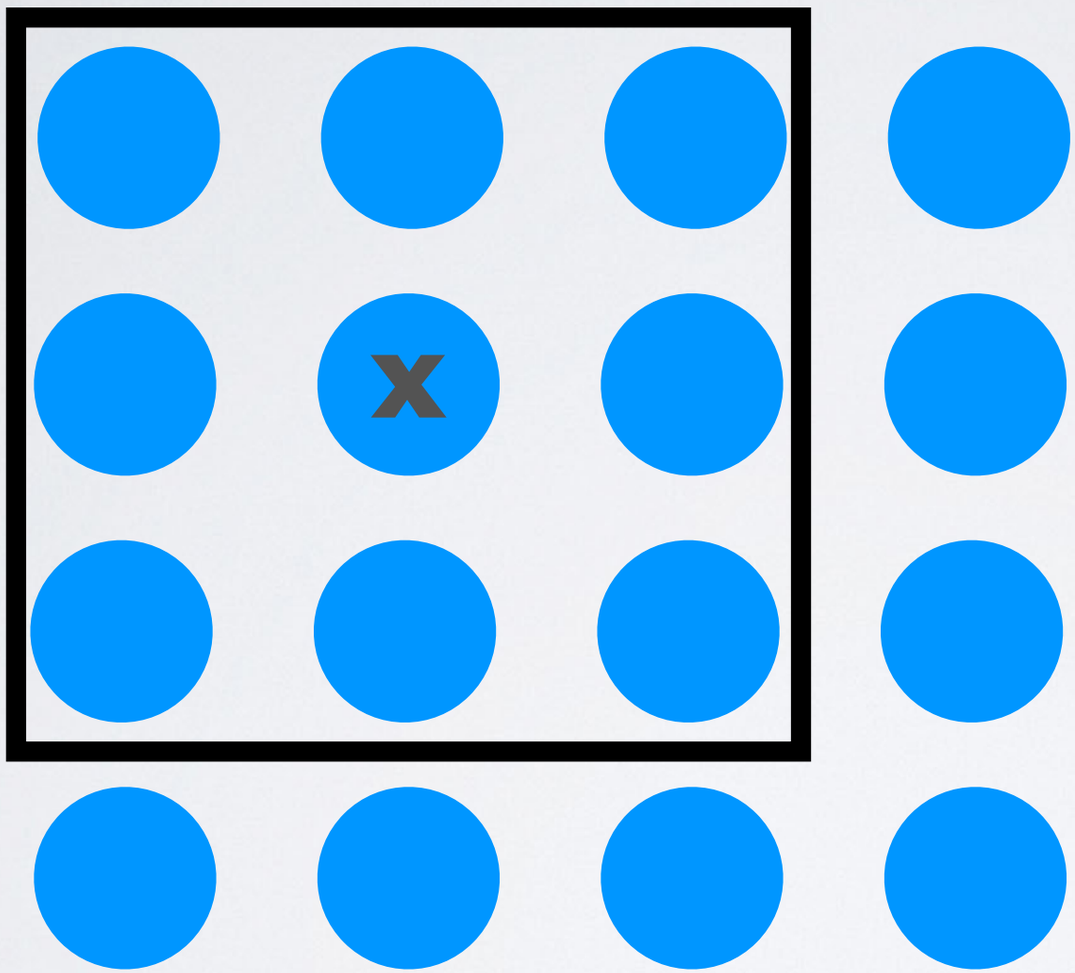
w_1

“Feature map”



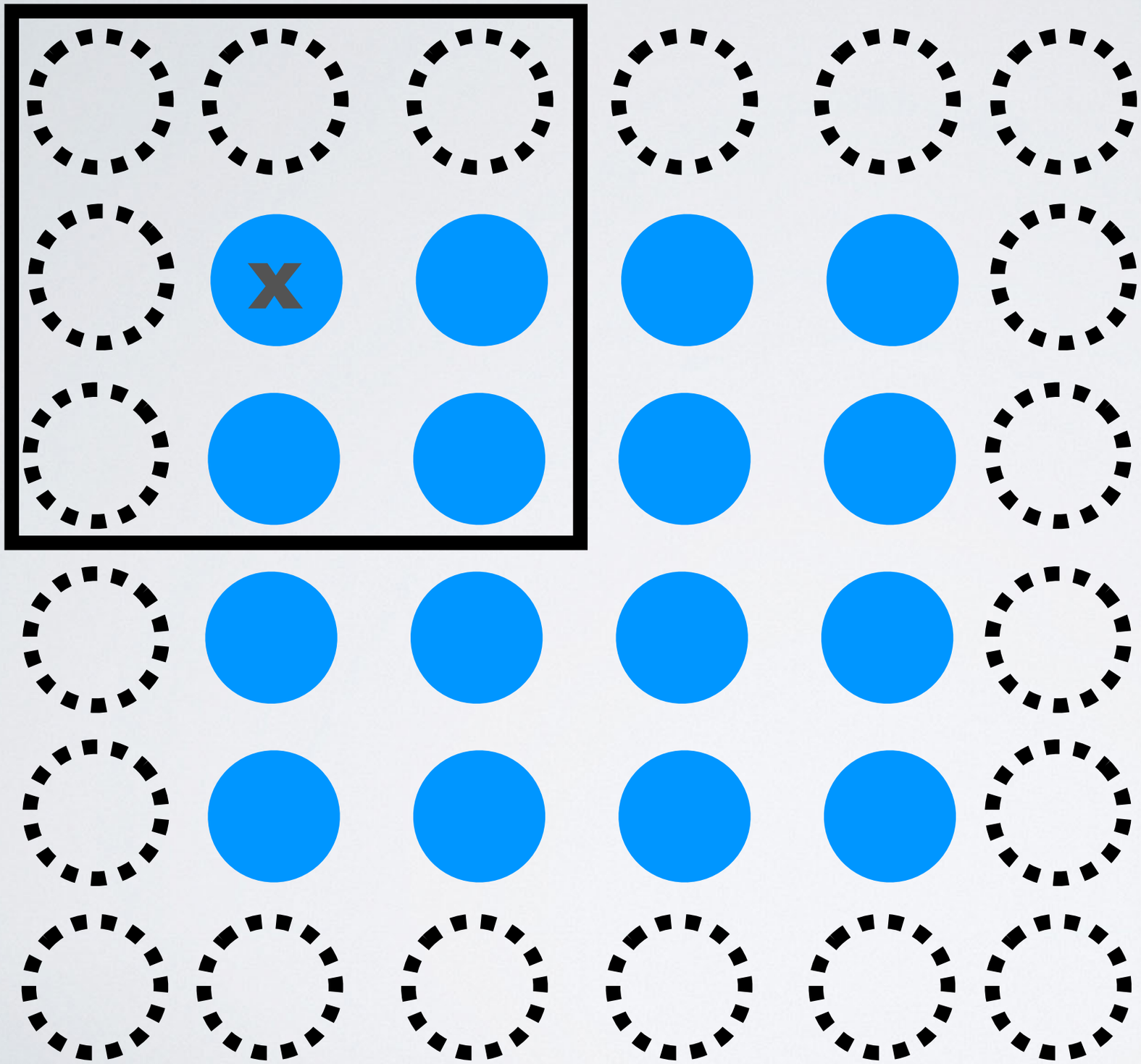
VARIANTS OF CONV

PADDING



3x3 conv produces feature maps that are 2 units smaller

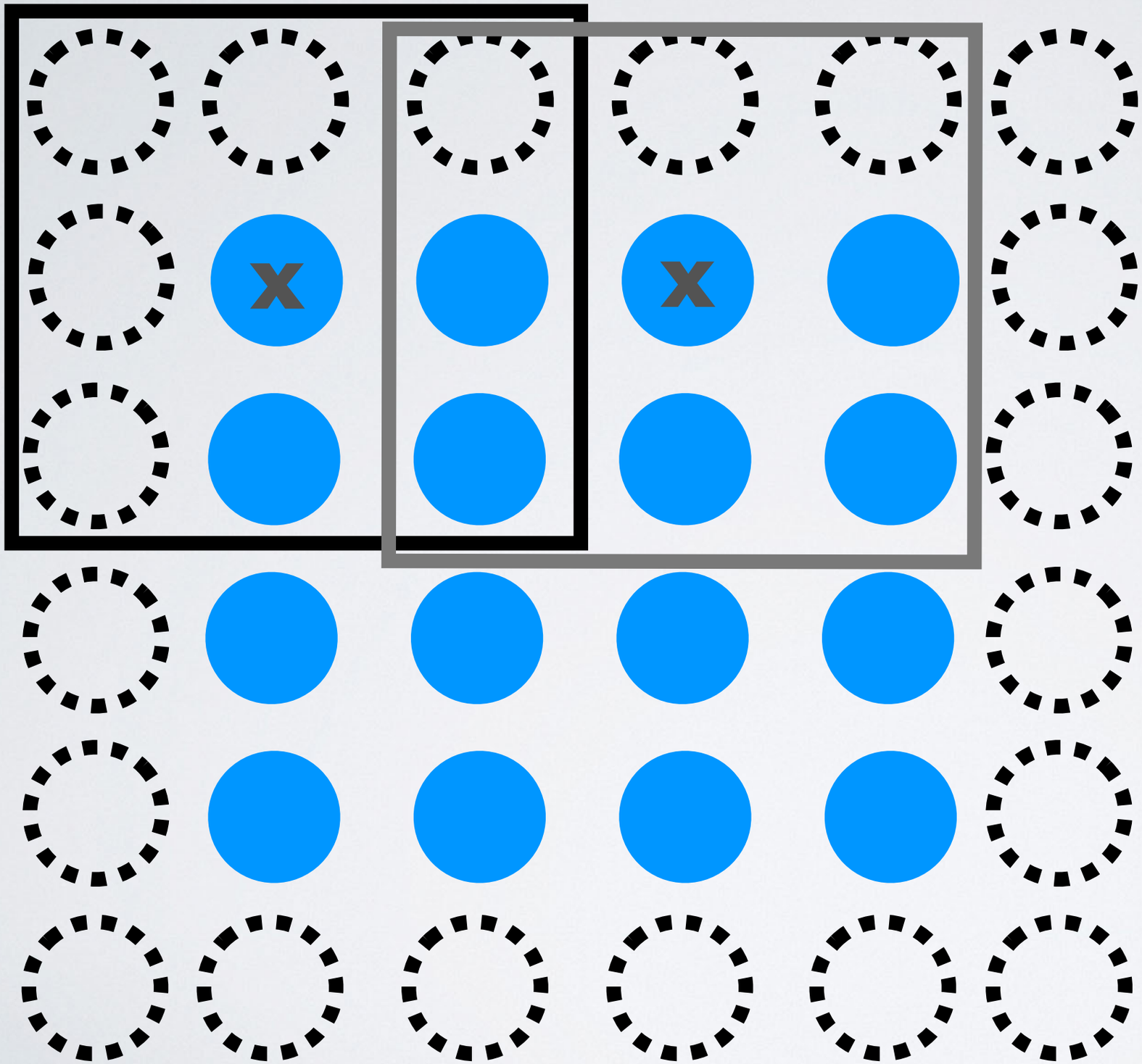
PADDING



padding=1

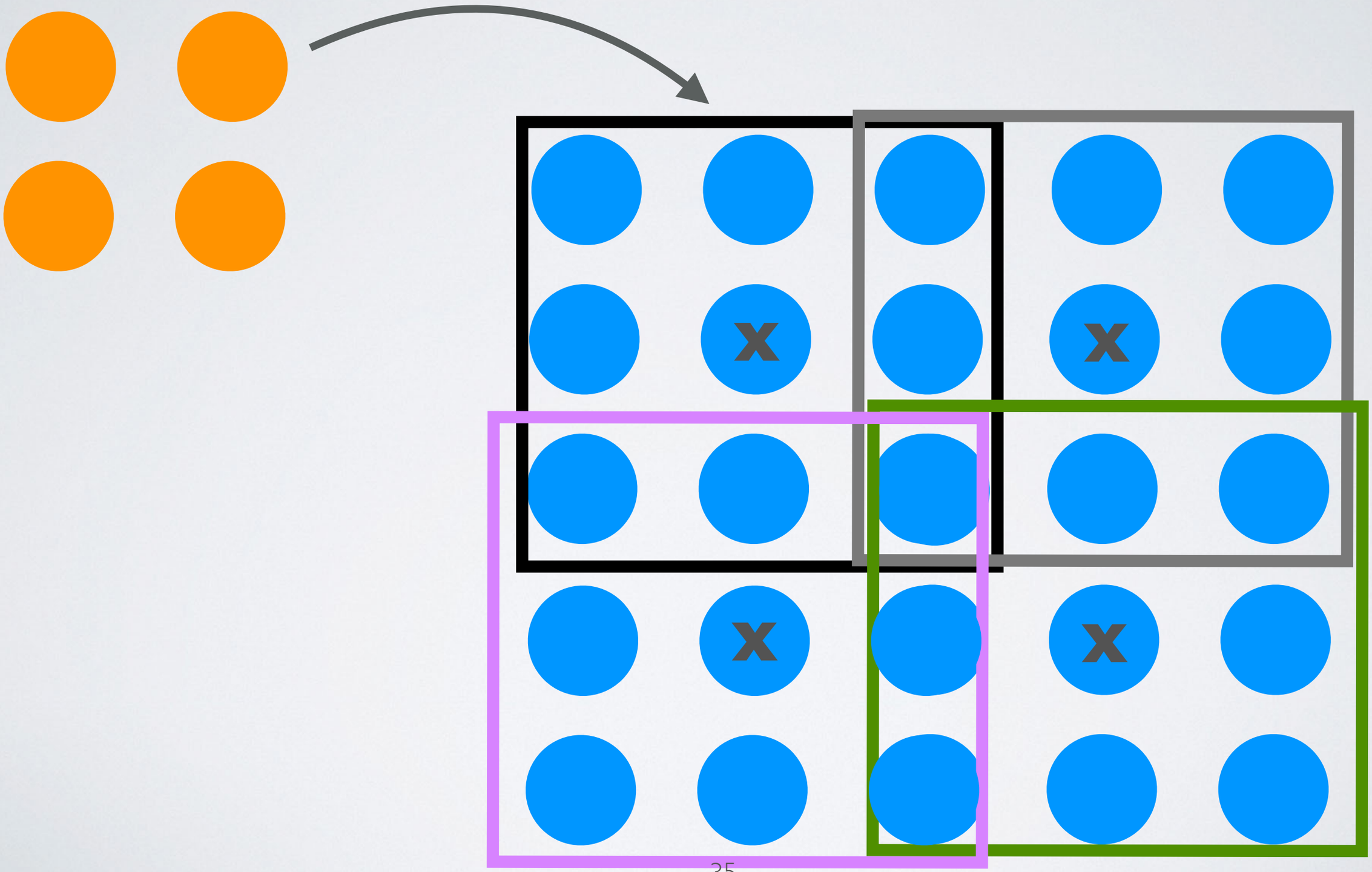
Output is same size as input

STRIDE



stride=2 cuts resolution by half

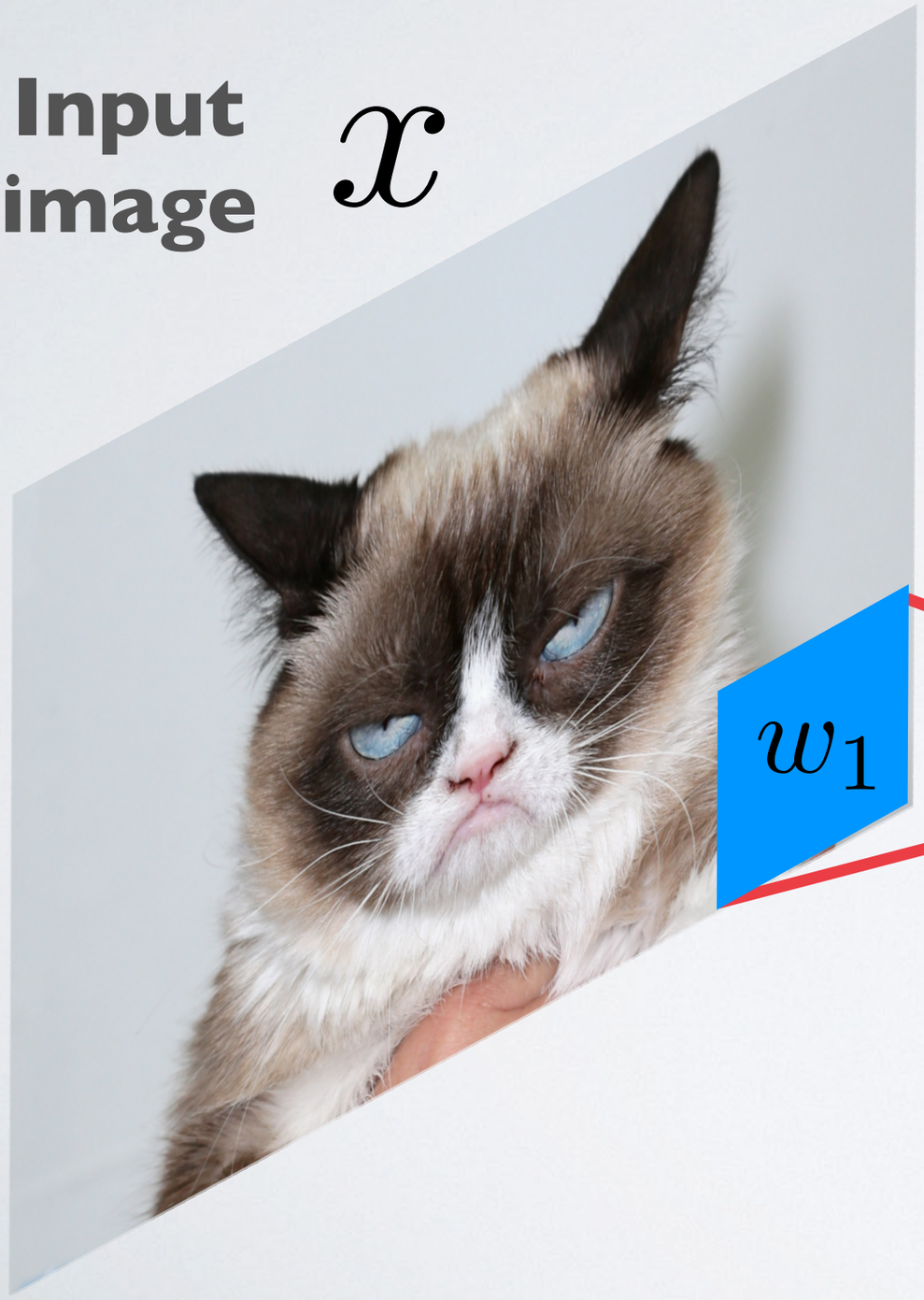
TRANSPOSE / DECONVOLUTION



RELU

RELU NONLINEARITY

Input image x



w_1

“Feature map”

remove negatives

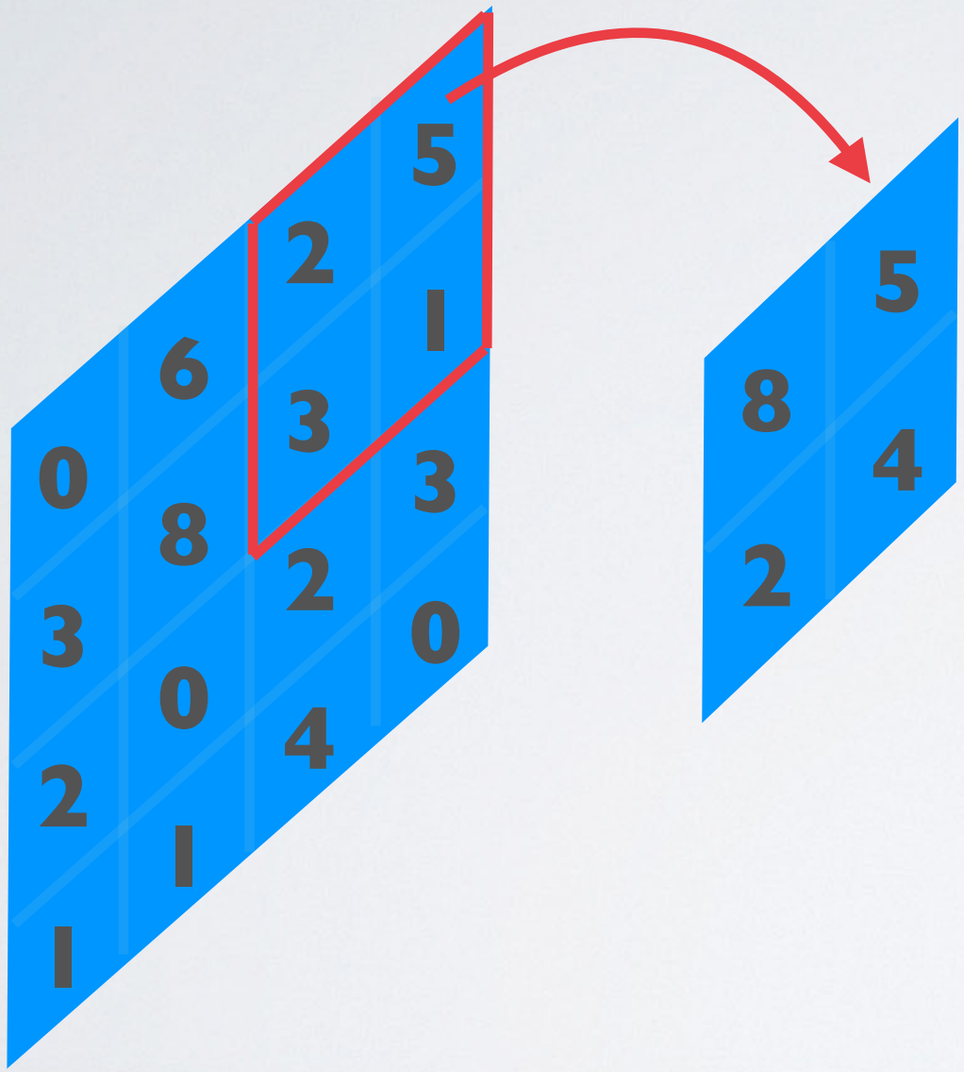
5	-3	-2
1	3	7
-9	2	-8

5	0	0
1	3	7
0	2	0

POOLING

POOLING

moving to lower resolutions



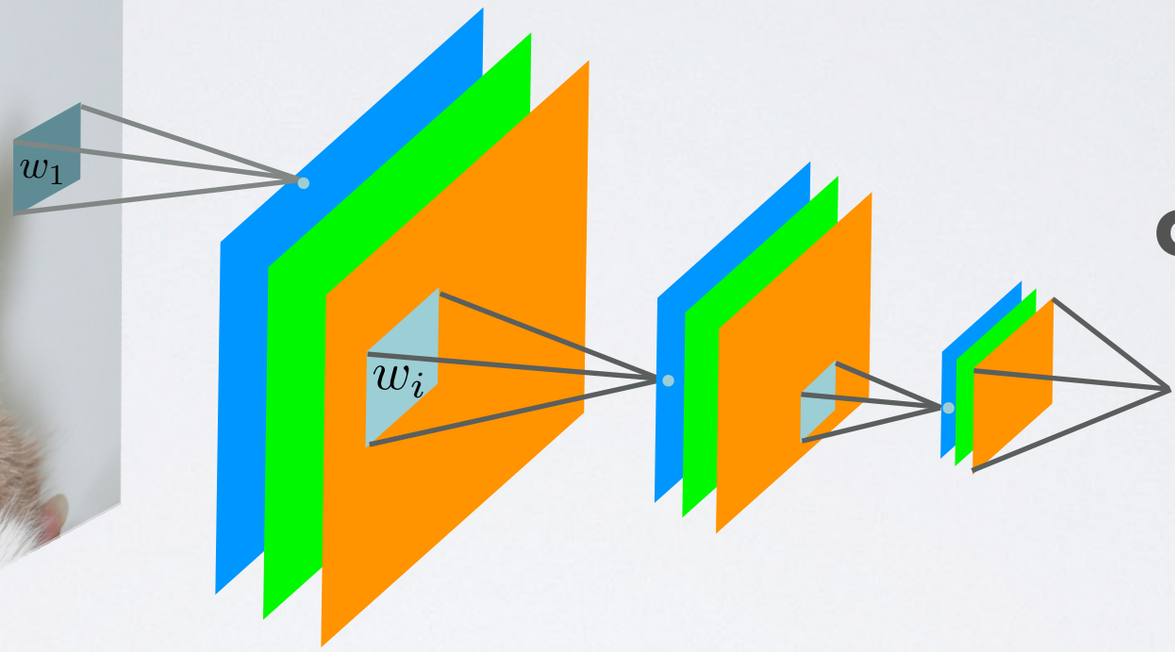
Max pooling

Average pooling

Strided convolution
(aka downsampling)

THAT'S ALL YOU NEED!

x



y

One's hot label

.05	.1	.9	.04
ball	car	hat	

grumpy cat

$$f(x; w) \rightarrow y$$

image weights