10-301/601: Introduction to Machine Learning Lecture 17 – Convolutional Neural Networks

Matt Gormley & Henry Chai

3/17/25

Front Matter

- Announcements
  - HW6 released 3/16, due 3/22 at 11:59 PM
    - **You can only use at most two late days on HW6**
  - Exam 2 on 3/26 (next Wednesday) from **7 -9 PM**
    - All topics from Lecture 8 to Lecture 16 (inclusive) **+ the portion of today's lecture on MLE/MAP** are in-scope
    - Exam 1 content may be referenced but will not be the primary focus of any question

Recall: Bernoulli MLE

- A Bernoulli random variable takes value 1 with probability ϕ and value 0 with probability $1-\phi$
- The pmf of the Bernoulli distribution is

$$p(x|\phi) = \phi^x(1-\phi)^{1-x}$$

- Given N iid samples $\{x^{(1)}, \ldots, x^{(N)}\}$, the log-likelihood is

$$\ell(\phi) = \sum_{n=1}^{N} \log p\left(x^{(n)}|\phi\right) = \sum_{n=1}^{N} \log \phi^{x^{(n)}}(1-\phi)^{1-x^{(n)}}$$

$$= \sum_{n=1}^{N} x \log \phi + (1-x)\log(1-\phi)$$

$$= N_1 \log \phi + N_0 \log(1-\phi)$$

where N_1 is the number of 1's in $\{x^{(1)}, \ldots, x^{(N)}\}$ and N_0 is the number of 0's

Recall: Bernoulli MLE

- A Bernoulli random variable takes value 1 with probability ϕ and value 0 with probability $1-\phi$
- The pmf of the Bernoulli distribution is

$$p(x|\phi) = \phi^x(1-\phi)^{1-x}$$

- The partial derivative of the log-likelihood is

$$\frac{N_1}{\hat{\phi}} - \frac{N_0}{1-\hat{\phi}} = 0 \rightarrow \frac{N_1}{\hat{\phi}} = \frac{N_0}{1-\hat{\phi}}$$

$$\rightarrow N_1\left(1-\hat{\phi}\right) = N_0\hat{\phi} \rightarrow N_1 = \hat{\phi}(N_0 + N_1)$$

$$\rightarrow \hat{\phi} = \frac{N_1}{N_0 + N_1} = \frac{N_1}{N}$$

where N_1 is the number of 1's in $\left\{x^{(1)}, \ldots, x^{(N)}\right\}$ and N_0 is the number of 0's

Coin Flipping MLE

- A Bernoulli random variable takes value 1 (or heads) with probability ϕ and value 0 (or tails) with probability $1-\phi$
- The pmf of the Bernoulli distribution is

$$p(x|\phi) = \phi^x(1-\phi)^{1-x}$$

- The partial derivative of the log-likelihood is

$$\frac{N_1}{\hat{\phi}} - \frac{N_0}{1-\hat{\phi}} = 0 \rightarrow \frac{N_1}{\hat{\phi}} = \frac{N_0}{1-\hat{\phi}}$$

$$\rightarrow N_1\left(1-\hat{\phi}\right) = N_0\hat{\phi} \rightarrow N_1 = \hat{\phi}(N_0 + N_1)$$

$$\rightarrow \hat{\phi} = \frac{N_1}{N_0 + N_1} = \frac{N_1}{N}$$

where N_1 is the number of heads in our dataset and N_0 is the number of tails

Poll Question 1:

Flip your coin 5 times and based on the results of your flip, report the MLE of your coin

A. 0/5
B. 1/5
C. 2/5
D. 3/5
E. $\pi/5$ **(TOXIC)**
F. 4/5
G. 5/5

- A Bernoulli random variable takes value 1 (or heads) with probability ϕ and value 0 (or tails) with probability $1-\phi$
- The pmf of the Bernoulli distribution is

$$p(x|\phi) = \phi^x(1-\phi)^{1-x}$$

- The partial derivative of the log-likelihood is

$$\frac{N_1}{\hat{\phi}} - \frac{N_0}{1-\hat{\phi}} = 0 \rightarrow \frac{N_1}{\hat{\phi}} = \frac{N_0}{1-\hat{\phi}}$$

$$\rightarrow N_1\left(1-\hat{\phi}\right) = N_0\hat{\phi} \rightarrow N_1 = \hat{\phi}(N_0 + N_1)$$

$$\rightarrow \hat{\phi} = \frac{N_1}{N_0 + N_1} = \frac{N_1}{N}$$

where N_1 is the number of heads in our dataset and N_0 is the number of tails

Maximum a Posteriori (MAP) Estimation

- Insight: sometimes we have *prior* information we want to incorporate into parameter estimation
- Idea: use Bayes rule to reason about the *posterior* distribution over the parameters
  - MLE finds $\hat{\theta} = \underset{\theta}{\text{argmax}}\ p(\mathcal{D}|\theta)$
  - MAP finds $\hat{\theta} = \underset{\theta}{\text{argmax}}\ p(\theta|\mathcal{D})$

$$= \underset{\theta}{\text{argmax}}\ p(\mathcal{D}|\theta)p(\theta)/p(\mathcal{D})$$

$$= \underset{\theta}{\text{argmax}}\ p(\mathcal{D}|\theta)p(\theta)$$

likelihood prior

$$= \underset{\theta}{\text{argmax}}\ \log p(\mathcal{D}|\theta) + \log p(\theta)$$

log-posterior

Coin Flipping MAP

- A Bernoulli random variable takes value 1 (or heads) with probability ϕ and value 0 (or tails) with probability $1 - \phi$
- The pmf of the Bernoulli distribution is

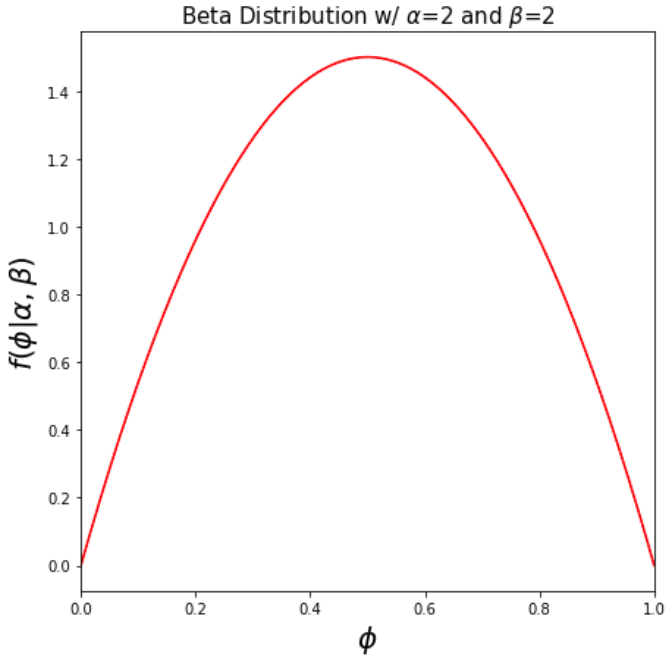
$$p(x|\phi) = \phi^x(1-\phi)^{1-x}$$

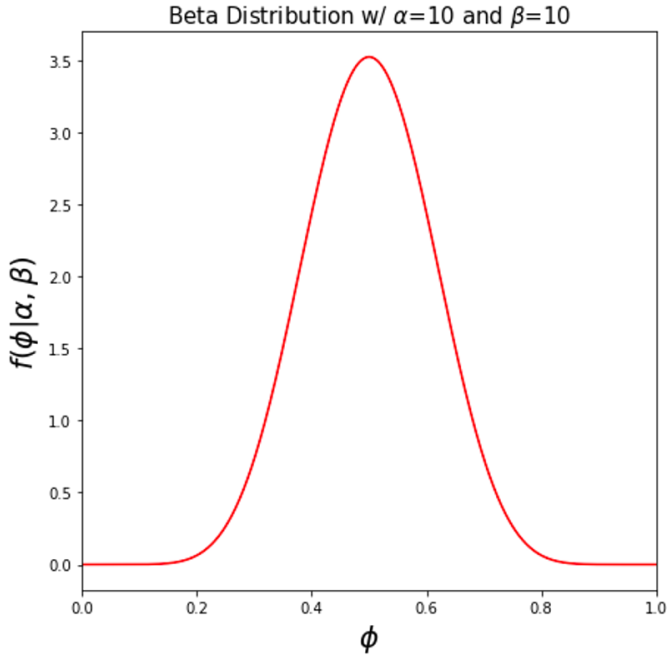
- Assume a *Beta* prior over the parameter ϕ, which has pdf

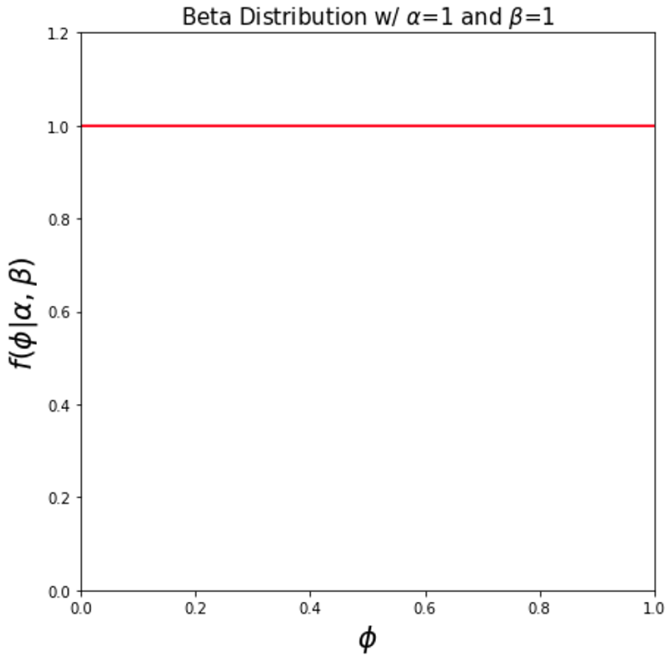
$$f(\phi|\alpha,\beta) = \frac{\phi^{\alpha-1}(1-\phi)^{\beta-1}}{\mathrm{B}(\alpha,\beta)}$$

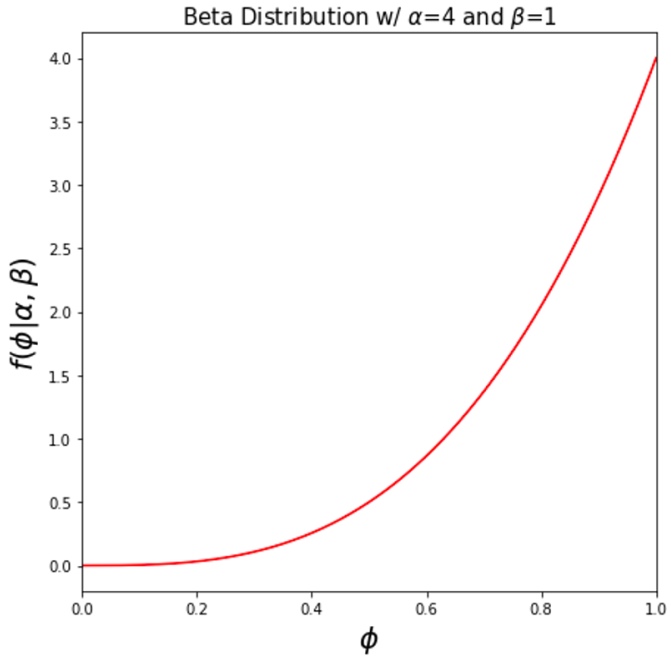
where $\mathrm{B}(\alpha,\beta) = \int_0^1 \phi^{\alpha-1}(1-\phi)^{\beta-1}d\phi$ is a normalizing constant to ensure the distribution integrates to 1

Beta Distribution

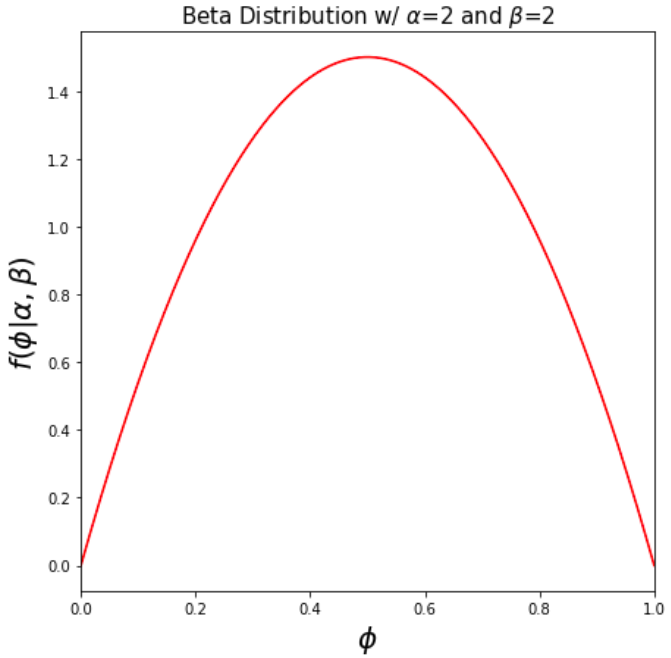


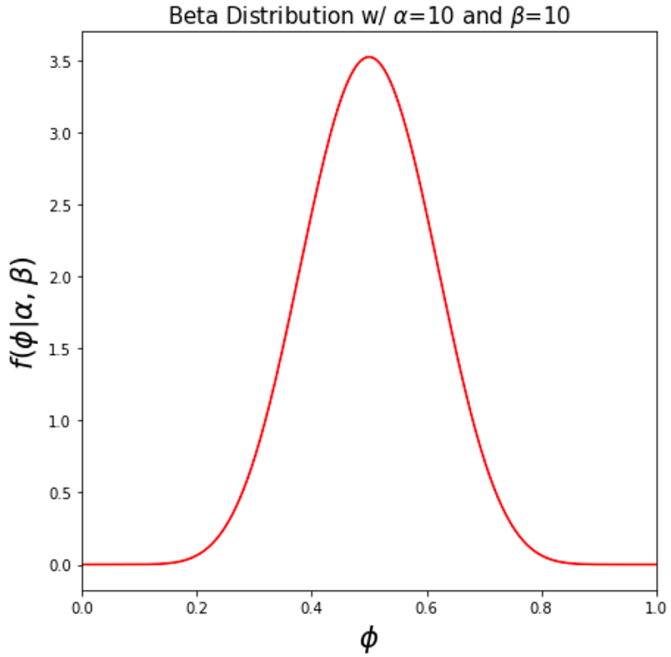


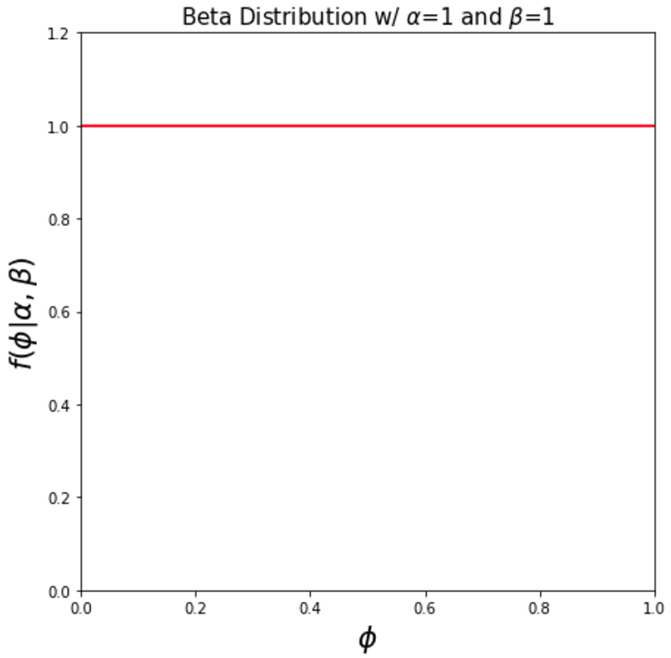


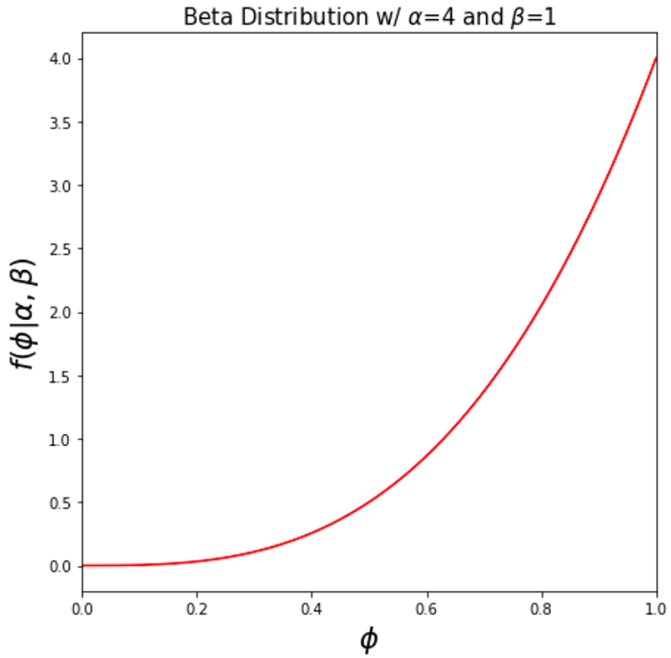


So why use this strange looking Beta prior?









The Beta distribution is the *conjugate prior* for the Bernoulli distribution!

- A Bernoulli random variable takes value 1 (or heads) with probability ϕ and value 0 (or tails) with probability $1 - \phi$
- The pmf of the Bernoulli distribution is

$$p(x|\phi) = \phi^x(1-\phi)^{1-x}$$

- Assume a Beta prior over the parameter ϕ, which has pdf

$$f(\phi|\alpha,\beta) = \frac{\phi^{\alpha-1}(1-\phi)^{\beta-1}}{\mathrm{B}(\alpha,\beta)}$$

where $\mathrm{B}(\alpha,\beta) = \int_0^1 \phi^{\alpha-1}(1-\phi)^{\beta-1}d\phi$ is a normalizing constant to ensure the distribution integrates to 1

Coin Flipping MAP

- Given N iid samples $\{x^{(1)}, \ldots, x^{(N)}\}$, the log-posterior is

$$\ell(\phi) = \log f(\phi|\alpha,\beta) + \sum_{n=1}^{N} \log p\left(x^{(n)}\middle|\phi\right)$$

$$= \log \frac{\phi^{\alpha-1}(1-\phi)^{\beta-1}}{\mathrm{B}(\alpha,\beta)} + \sum_{n=1}^{N} \log \phi^{x^{(n)}}(1-\phi)^{1-x^{(n)}}$$

$$= (\alpha-1)\log\phi + (\beta-1)\log(1-\phi) - \log \mathrm{B}(\alpha,\beta)$$

$$+ \sum_{n=1}^{N} x^{(n)}\log\phi + \left(1-x^{(n)}\right)\log(1-\phi)$$

$$= (\alpha-1+N_1)\log\phi + (\beta-1+N_0)\log(1-\phi)$$

$$- \log \mathrm{B}(\alpha,\beta)$$

Coin Flipping MAP

- Given N iid samples $\{x^{(1)}, \ldots, x^{(N)}\}$, the partial derivative of the log-posterior is

$$\frac{\partial \ell}{\partial \phi} = \frac{(\alpha - 1 + N_1)}{\phi} - \frac{(\beta - 1 + N_0)}{1 - \phi}$$

$$\vdots$$

$$\rightarrow \hat{\phi}_{MAP} = \frac{(\alpha - 1 + N_1)}{(\beta - 1 + N_0) + (\alpha - 1 + N_1)}$$

- $\alpha - 1$ is a "pseudocount" of the number of 1's (or heads) you've "observed"
- $\beta - 1$ is a "pseudocount" of the number of 0's (or tails) you've "observed"

Coin Flipping MAP: Example

- Suppose $\mathcal{D}$ consists of ten 1's or heads ($N_1 = 10$) and two 0's or tails ($N_0 = 2$):

$$\phi_{MLE} = \frac{10}{10+2} = \frac{10}{12}$$

- Using a Beta prior with $\alpha = 101$ and $\beta = 101$, then

$$\phi_{MAP} = \frac{(101-1+10)}{(101-1+10)+(101-1+2)} = \frac{110}{212} \approx \frac{1}{2}$$

Coin Flipping MAP: Example

- Suppose $\mathcal{D}$ consists of ten 1's or heads ($N_1 = 10$) and two 0's or tails ($N_0 = 2$):

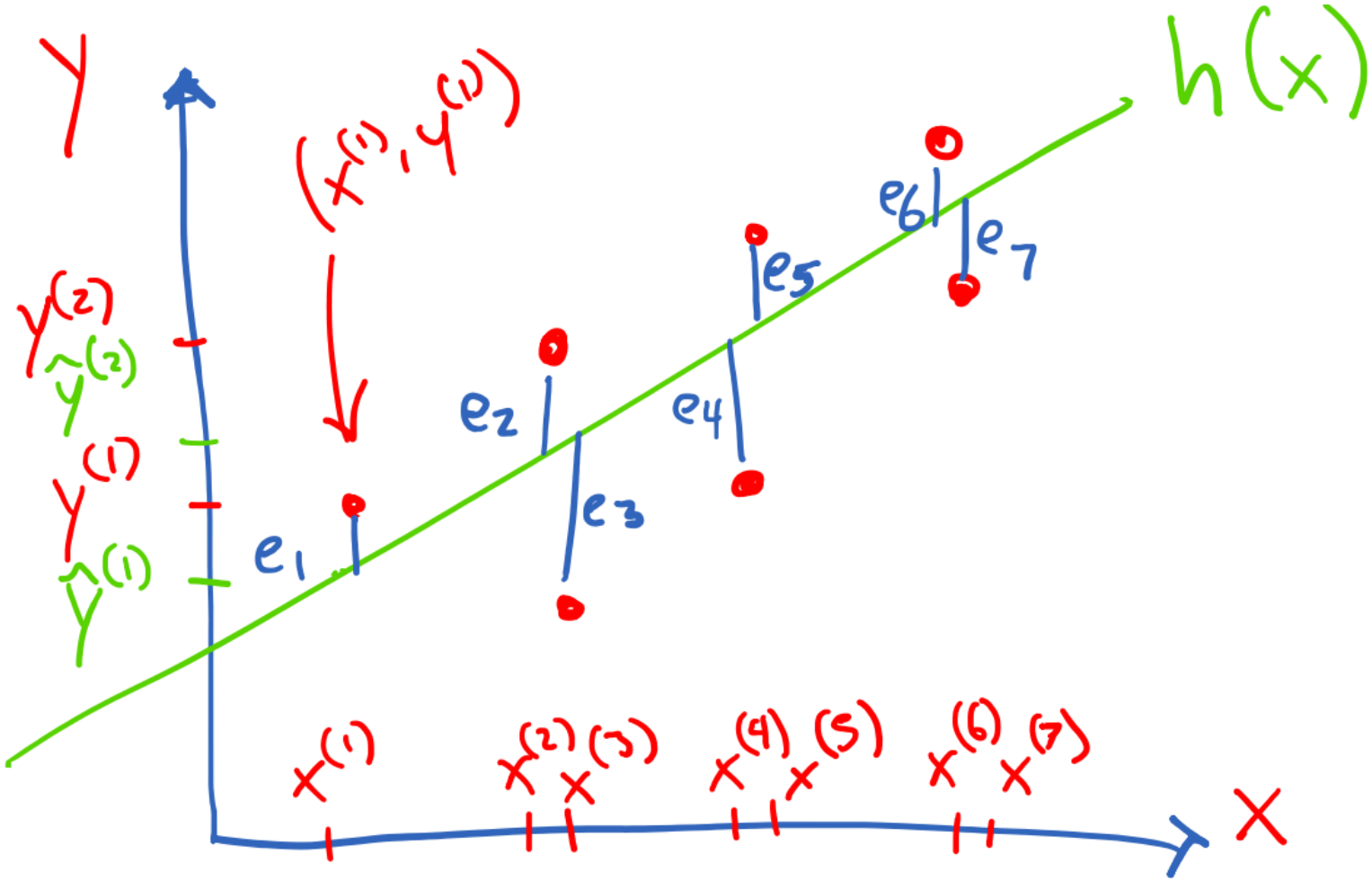
$$\phi_{MLE} = \frac{10}{10+2} = \frac{10}{12}$$

- Using a Beta prior with $\alpha = 1$ and $\beta = 1$, then

$$\phi_{MAP} = \frac{(1-1+10)}{(1-1+10)+(1-1+2)} = \frac{10}{12} = \phi_{MLE}$$

M(C)LE for Linear Regression

- One way of conceptualizing linear regression is that the residuals are random variables!



M(C)LE for Linear Regression

- One way of conceptualizing linear regression is that the residuals are random variables!
- If the residuals are Gaussian …

$$y = \boldsymbol{\theta}^T \boldsymbol{x} + \epsilon \text{ where } \epsilon \sim N(0, \sigma^2) \rightarrow y \sim N(\boldsymbol{w}^T \boldsymbol{x}, \sigma^2)$$

- … then given $X = \begin{bmatrix} 1 & \boldsymbol{x}^{(1)^T} \\ \vdots & \vdots \\ 1 & \boldsymbol{x}^{(N)^T} \end{bmatrix}$ and $\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(N)} \end{bmatrix}$ the MLE of $\boldsymbol{\theta}$ is

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\text{argmax}} \log P(\boldsymbol{y}|X, \boldsymbol{\theta})$$

$$\vdots$$

$$= \underset{\boldsymbol{\theta}}{\text{argmin}} (X\boldsymbol{\theta} - \boldsymbol{y})^T (X\boldsymbol{\theta} - \boldsymbol{y}) = (X^T X)^{-1} X^T \boldsymbol{y}$$

a.k.a. the OLS solution!

MAP for Linear Regression

- One way of conceptualizing linear regression is that the residuals are random variables!
- If the residuals are Gaussian …

$$y = \boldsymbol{\theta}^T \boldsymbol{x} + \epsilon \text{ where } \epsilon \sim N(0, \sigma^2) \rightarrow y \sim N(\boldsymbol{w}^T \boldsymbol{x}, \sigma^2)$$

- … and we use a Gaussian prior on the weights …

$$\theta_d \sim N\left(0, \frac{\sigma^2}{\lambda}\right) \forall\, d \in 0, \ldots, D$$

- … then, the MAP of $\boldsymbol{\theta}$ is the ridge regression solution!

$$\widehat{\boldsymbol{\theta}}_{MAP} = (X^T X + \lambda I_{D+1})^{-1} X^T \boldsymbol{y}$$

MLE/MAP Learning Objectives

You should be able to…

- Recall probability basics, including but not limited to: discrete and continuous random variables, probability mass functions, probability density functions, events vs. random variables, expectation and variance, joint probability distributions, marginal probabilities, conditional probabilities, independence, conditional independence
- State the principle of maximum likelihood estimation and explain what it tries to accomplish
- State the principle of maximum a posteriori estimation and explain why we use it
- Derive the MLE or MAP parameters of a simple model in closed form

Deep Learning

- From Wikipedia's page on Deep Learning…

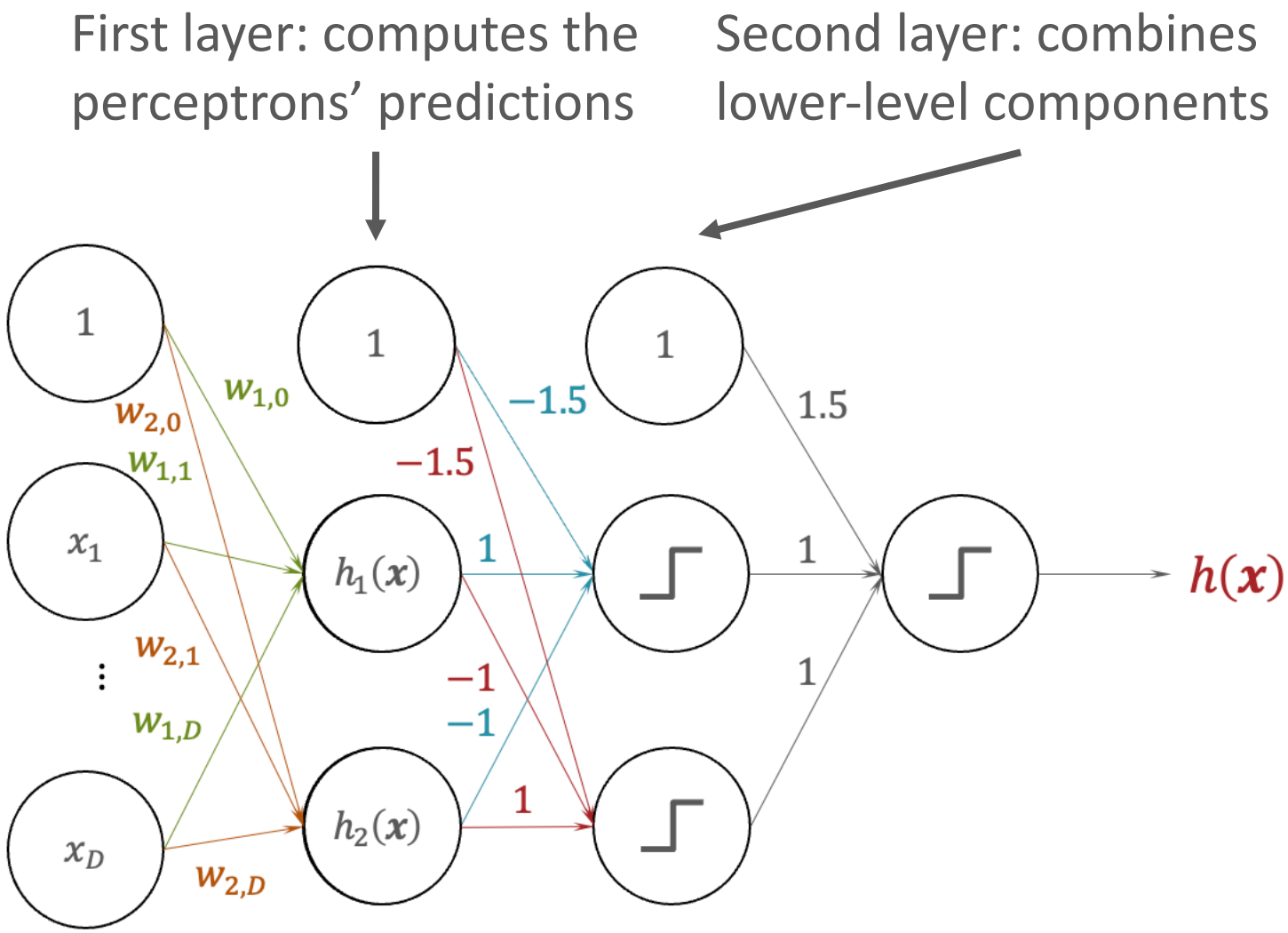
Definition [edit]

Deep learning is a class of machine learning algorithms that[11](pp199–200) uses multiple layers to progressively extract higher level features from the raw input. For example, in image processing, lower layers may identify edges, while higher layers may identify the concepts relevant to a human such as digits or letters or faces.

- Deep learning = more than one layer

Source: https://en.wikipedia.org/wiki/Deep_learning

Deep Learning



Convolutional Neural Networks

- Neural networks are frequently applied to inputs with some inherent spatial structure, e.g., images
- Idea: use the first few layers to identify relevant macro-features, e.g., edges
- Insight: for spatially-structured inputs, many useful macro-features are shift or location-invariant, e.g., an edge in the upper left corner of a picture looks like an edge in the center
- Strategy: learn a filter for macro-feature detection in a small window and apply it over the entire image

Convolutional Filters

- Images can be represented as matrices, where each element corresponds to a pixel
- A filter is just a small matrix that is convolved with same-sized sections of the image matrix

0	0	0	0	0	0
0	1	2	2	1	0
0	2	4	4	2	0
0	1	3	3	1	0
0	1	2	3	1	0
0	0	1	1	0	0

$*$

0	1	0
1	-4	1
0	1	0

Convolutional Filters

- Images can be represented as matrices, where each element corresponds to a pixel
- A filter is just a small matrix that is convolved with same-sized sections of the image matrix

0	0	0	0	0	0
0	1	2	2	1	0
0	2	4	4	2	0
0	1	3	3	1	0
0	1	2	3	1	0
0	0	1	1	0	0

$*$

0	1	0
1	-4	1
0	1	0

$=$

0			

$$(0 * 0) + (0 * 1) + (0 * 0) + (0 * 1) + (1 * -4) + (2 * 1) + (0 * 0) + (2 * 1) + (4 * 0) = 0$$

Convolutional Filters

- Images can be represented as matrices, where each element corresponds to a pixel
- A filter is just a small matrix that is convolved with same-sized sections of the image matrix

0	0	0	0	0	0
0	1	2	2	1	0
0	2	4	4	2	0
0	1	3	3	1	0
0	1	2	3	1	0
0	0	1	1	0	0

$*$

0	1	0
1	-4	1
0	1	0

$=$

0	-1		

$$(0 * 0) + (0 * 1) + (0 * 0) + (1 * 1) + (2 * -4) + (2 * 1) + (2 * 0) + (4 * 1) + (4 * 0) = -1$$

Convolutional Filters

- Images can be represented as matrices, where each element corresponds to a pixel
- A filter is just a small matrix that is convolved with same-sized sections of the image matrix

0	0	0	0	0	0
0	1	2	2	1	0
0	2	4	4	2	0
0	1	3	3	1	0
0	1	2	3	1	0
0	0	1	1	0	0

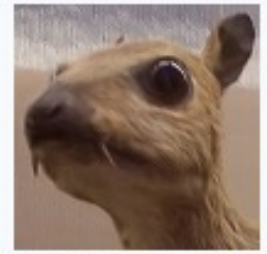
$*$

0	1	0
1	-4	1
0	1	0

$=$

0	-1	-1	0
-2	-5	-5	-2
2	-2	-1	3
-1	0	-5	0

Convolutional Filters

Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

Source: https://en.wikipedia.org/wiki/Kernel_(image_processing)

Poll Question 2:

What effect do you think the following filter will have on an image?

A. Sharpen the image

B. Blur the image

C. Shift the image left

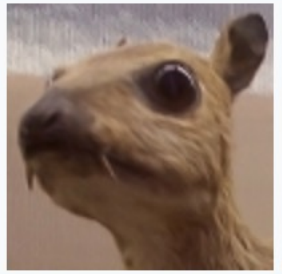
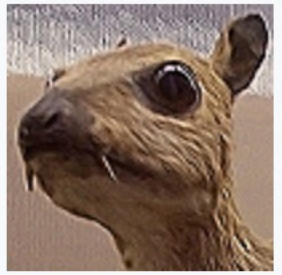
D. Rotate the image clockwise

E. Nothing (TOXIC)

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Source: https://en.wikipedia.org/wiki/Kernel_(image_processing)

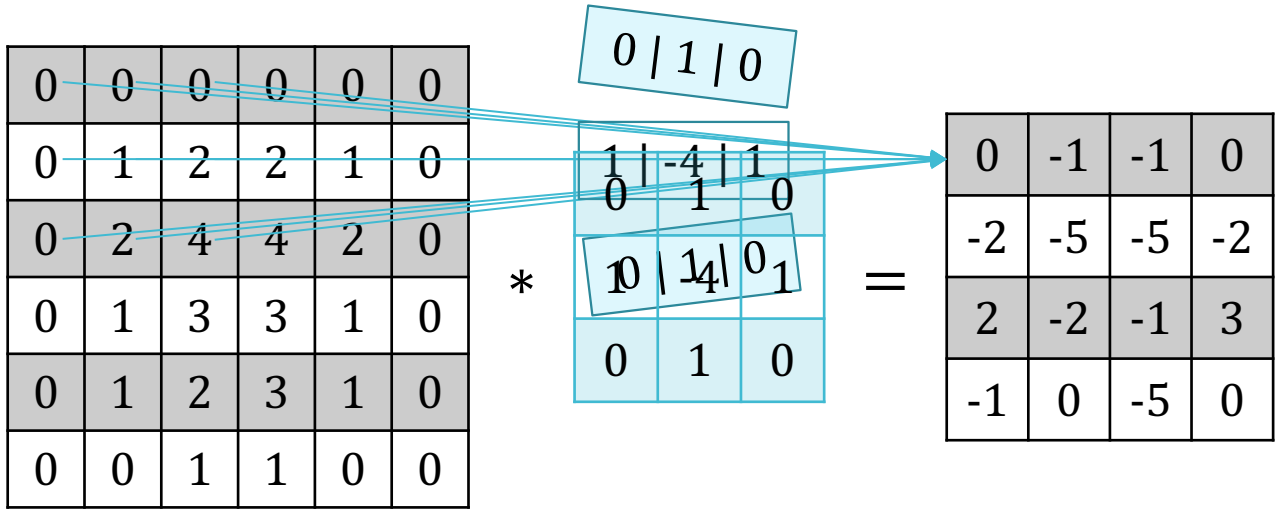
More Filters

Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

Source: https://en.wikipedia.org/wiki/Kernel_(image_processing)

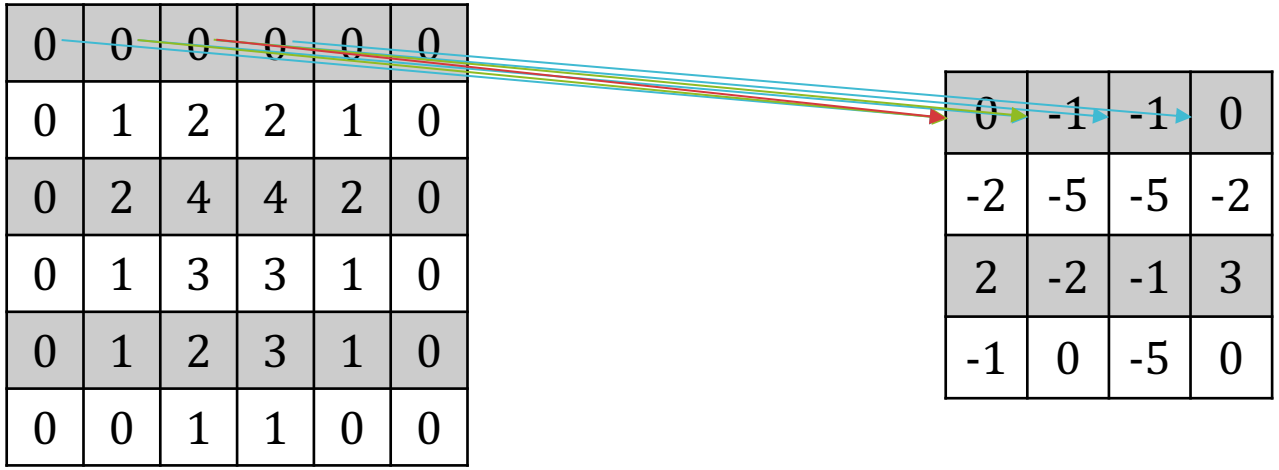
Convolutional Filters

- Images can be represented as matrices, where each element corresponds to a pixel
- A filter is just a small matrix that is convolved with same-sized sections of the image matrix



Convolutional Filters

- Convolutions can be represented by a feed forward neural network where:
  1. Nodes in the input layer are only connected to some nodes in the next layer but not all nodes.
  2. Many of the weights have the same value.



- Many fewer weights than a fully connected layer!
- Convolution weights are learned using gradient descent/ backpropagation, not prespecified

Convolutional Filters: Padding

- What if relevant features exist at the border of our image?
- Add zeros around the image to allow for the filter to be applied “everywhere” e.g. a *padding* of 1 with a 3x3 filter preserves image size and allows every pixel to be the center

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	2	2	1	0	0
0	0	2	4	4	2	0	0
0	0	1	3	3	1	0	0
0	0	1	2	3	1	0	0
0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	0

$*$

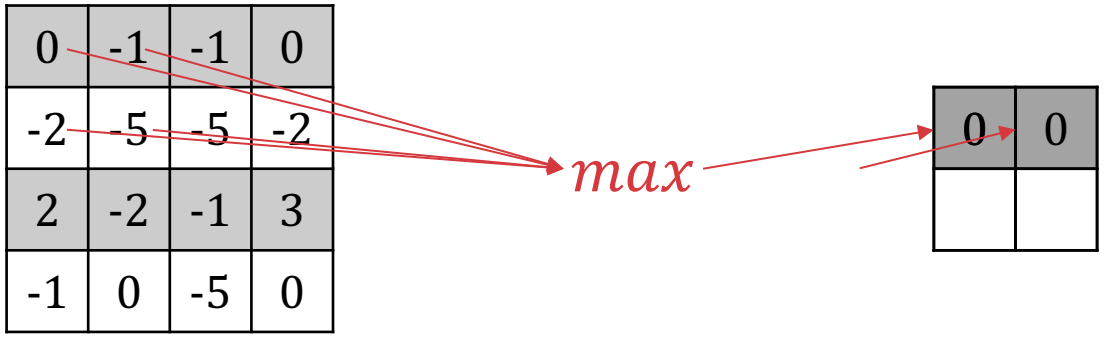
0	1	0
1	-4	1
0	1	0

$=$

0	1	2	2	1	0
1	0	-1	-1	0	1
2	-2	-5	-5	-2	2
1	2	-2	-1	3	1
1	-1	0	-5	0	1
0	2	-1	0	2	0

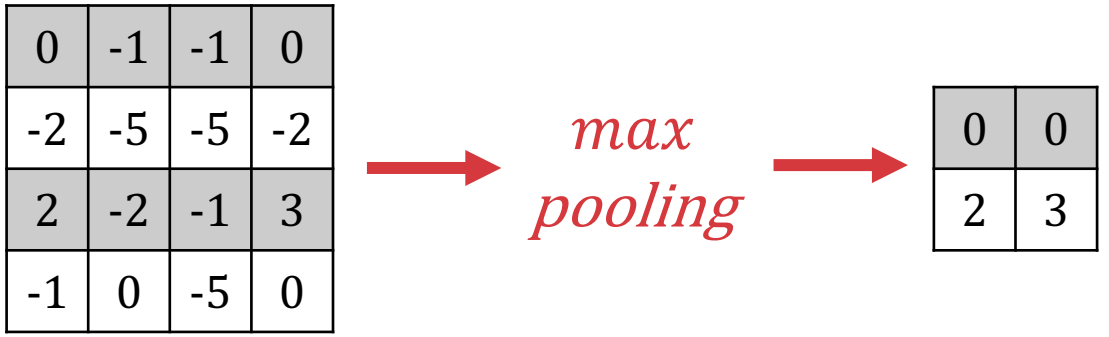
Downsampling: Pooling

- Combine multiple adjacent nodes into a single node



Downsampling: Pooling

- Combine multiple adjacent nodes into a single node

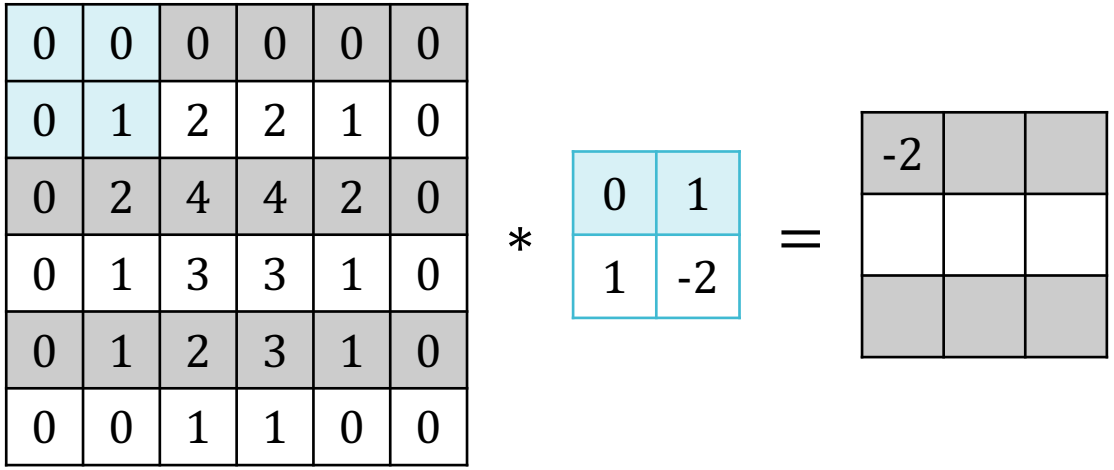


- Reduces the dimensionality of the input to subsequent layers and thus, the number of weights to be learned
  - Protects the network from (slightly) noisy inputs

Downsampling: Stride

- Only apply the convolution to some subset of the image

  e.g., every other column and row = a *stride* of 2



Downsampling: Stride

- Only apply the convolution to some subset of the image

  e.g., every other column and row = a *stride* of 2

0	0	0	0	0	0
0	1	2	2	1	0
0	2	4	4	2	0
0	1	3	3	1	0
0	1	2	3	1	0
0	0	1	1	0	0

$*$

0	1
1	-2

$=$

-2	-2	

Downsampling: Stride

- Only apply the convolution to some subset of the image

  e.g., every other column and row = a *stride* of 2

0	0	0	0	0	0
0	1	2	2	1	0
0	2	4	4	2	0
0	1	3	3	1	0
0	1	2	3	1	0
0	0	1	1	0	0

$*$

0	1
1	-2

$=$

-2	-2	1

Downsampling: Stride

- Only apply the convolution to some subset of the image

  e.g., every other column and row = a *stride* of 2

0	0	0	0	0	0
0	1	2	2	1	0
0	2	4	4	2	0
0	1	3	3	1	0
0	1	2	3	1	0
0	0	1	1	0	0

$*$

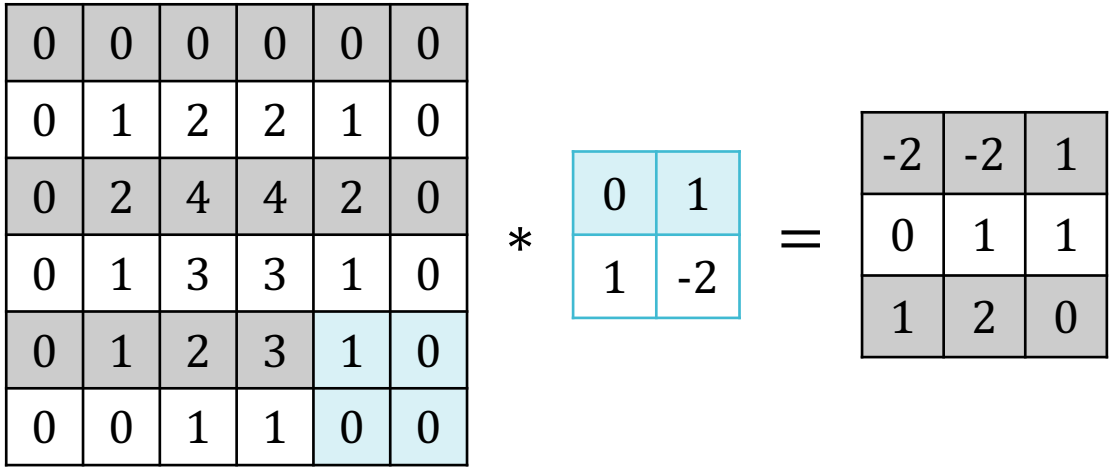
0	1
1	-2

$=$

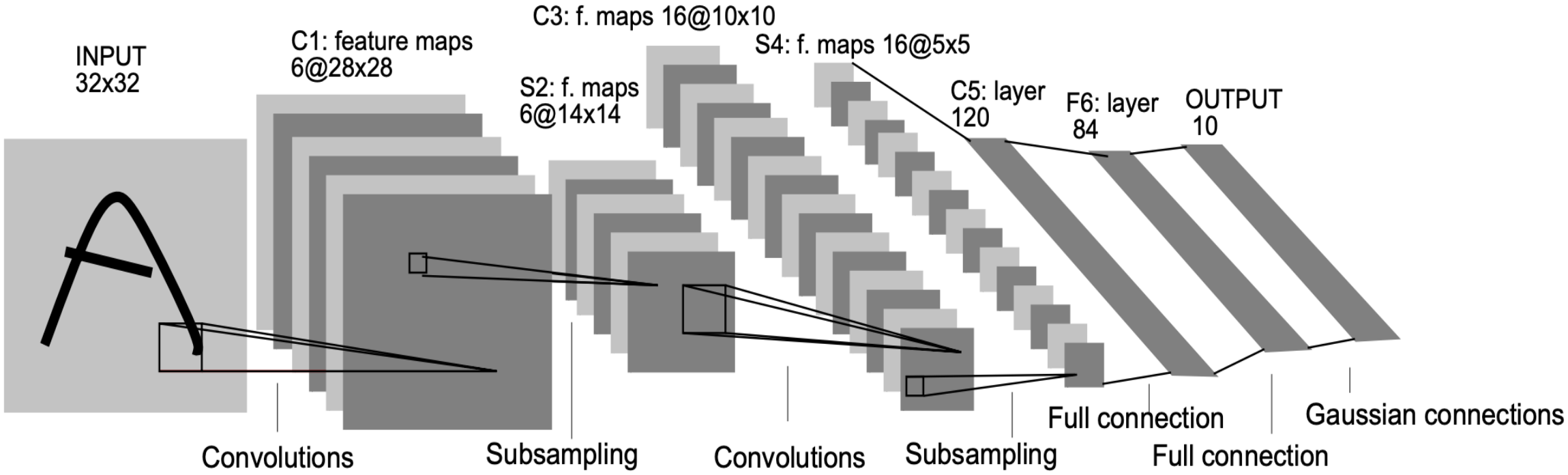
-2	-2	1
0		

Downsampling: Stride

- Only apply the convolution to some subset of the image
  e.g., every other column and row = a *stride* of 2

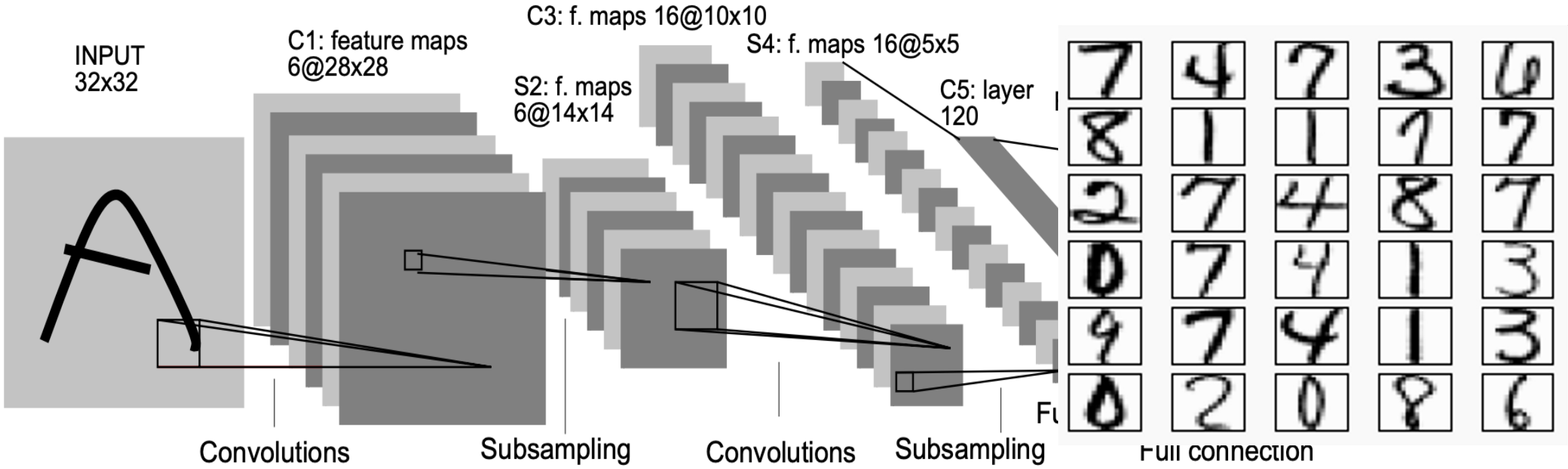


- Reduces the dimensionality of the input to subsequent layers and thus, the number of weights to be learned
- Many relevant macro-features will tend to span large portions of the image, so taking strides with the convolution tends not to miss out on too much



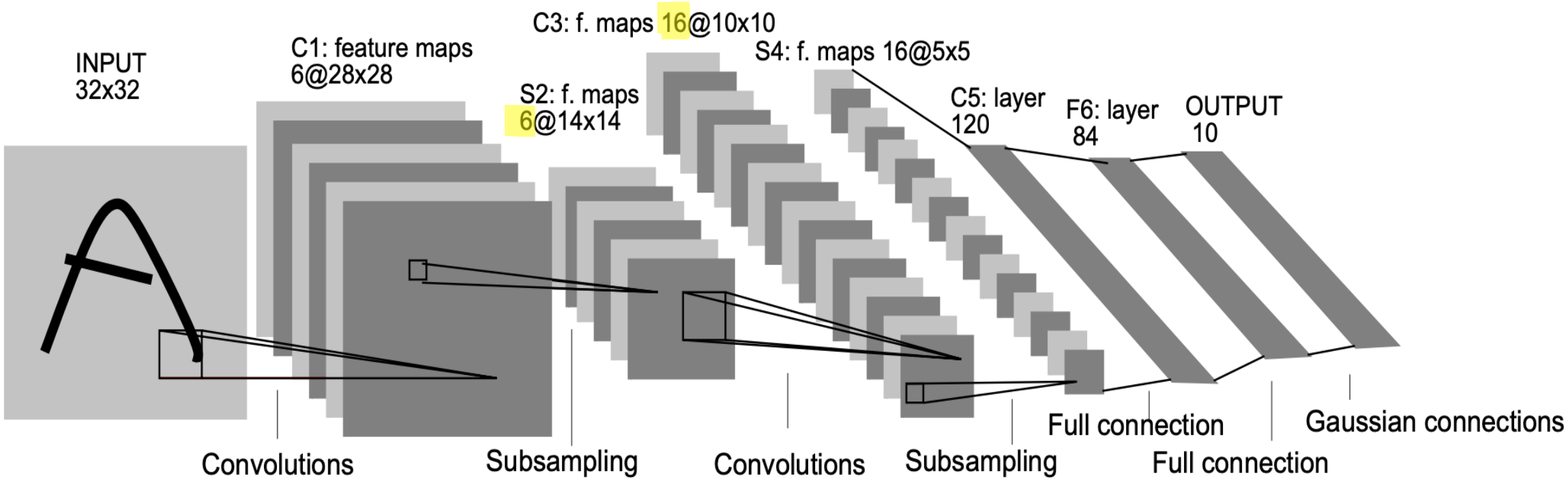
LeNet (LeCun et al., 1998)

Source: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf



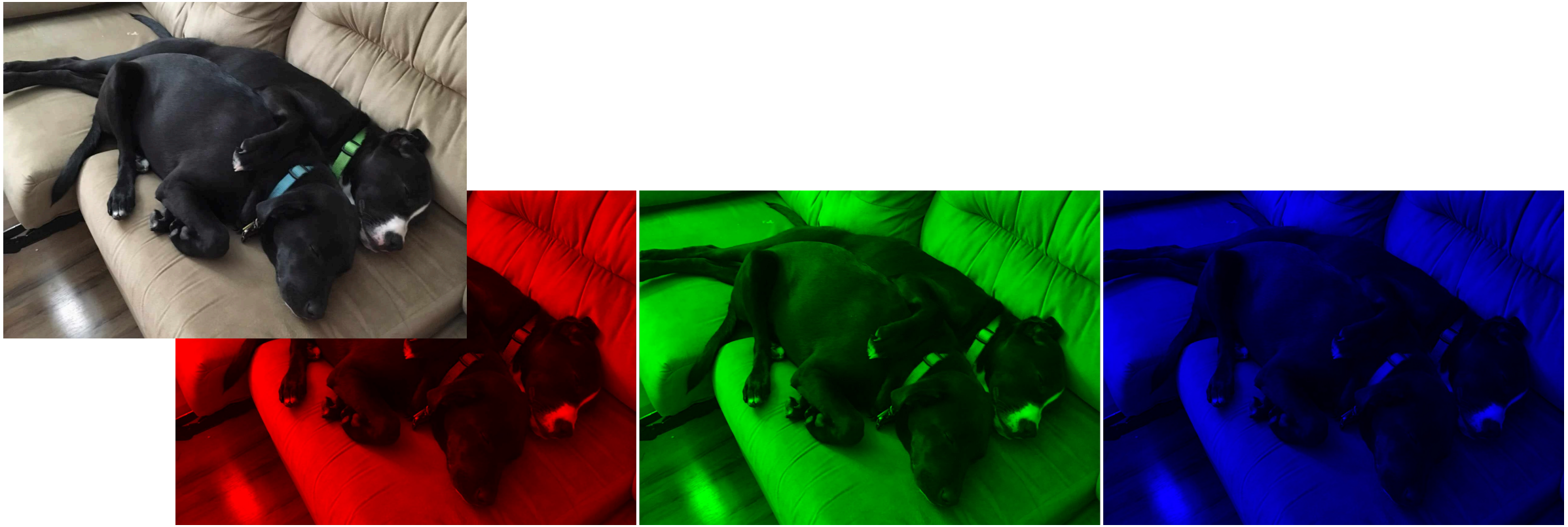
- One of the earliest, most famous deep learning models – achieved remarkable performance at handwritten digit recognition (< 1% test error rate on MNIST)
- Used sigmoid (or logistic) activation functions between layers and mean-pooling, both of which are pretty uncommon in modern architectures

Source: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf

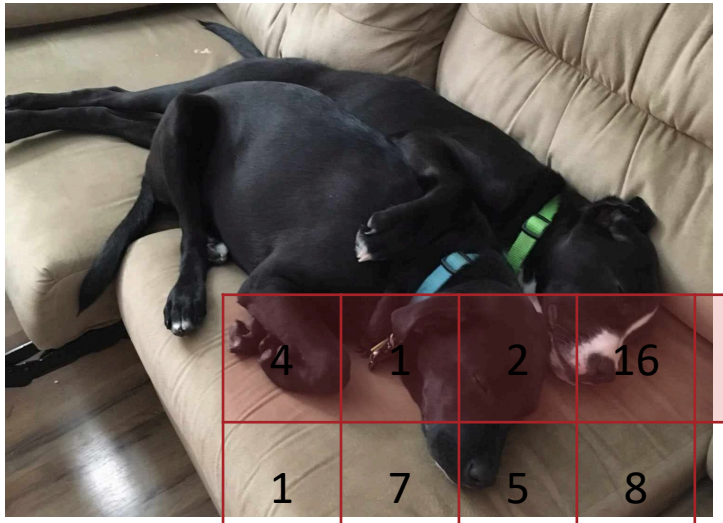


Wait how did we go from 6 to 16?

Source: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf



Channels



4	1	2	16	3	6
1	7	5	8	19	27
5	2	5	12	17	8
0	4	9	9	6	11

5	2	6	14	15	8
26	3	6	8	4	9
0	15	24	6	1	8
7	4	9	5	24	17

4	6	8	9	5	3
16	5	2	8	2	1
5	2	14	11	7	8
15	2	5	0	9	8

- An image can be represented as the sum of red, green and blue pixel intensities
- Each color corresponds to a *channel*

Example:

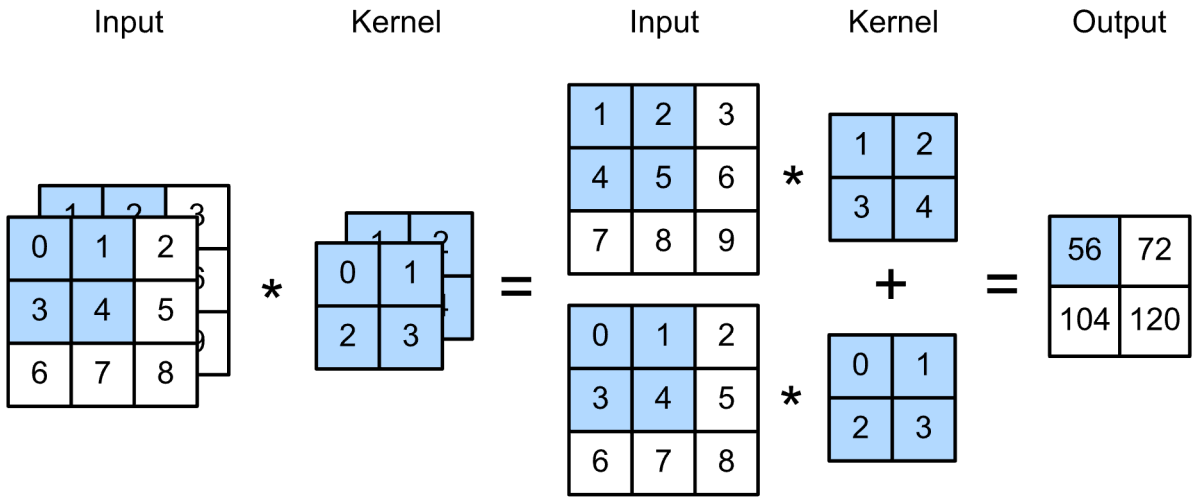
$3 \times 4 \times 6$ tensor

4	1	2	16	3	6
1	5	2	6	14	15	8
5	26	4	6	8	9	5	3
0	0	16	5	2	8	2	1
	7	5	2	14	11	7	8
		15	2	5	0	9	8

- An image can be represented as a *tensor* or multidimensional array

Convolutions on Multiple Input Channels

- Given multiple input channels, we can specify a filter for each one and sum the results to get a 2-D output tensor

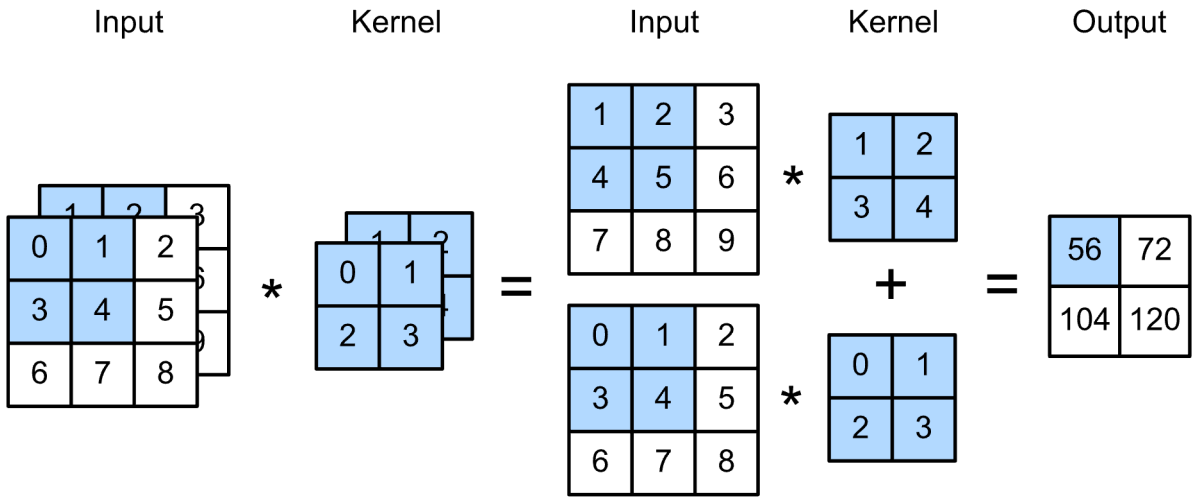


- For c channels and $h \times w$ filters, we have $chw + c$ learnable parameters (each filter has a bias term)

Source: http://preview.d2l.ai/d2l-en/master/chapter_convolutional-neural-networks/channels.html

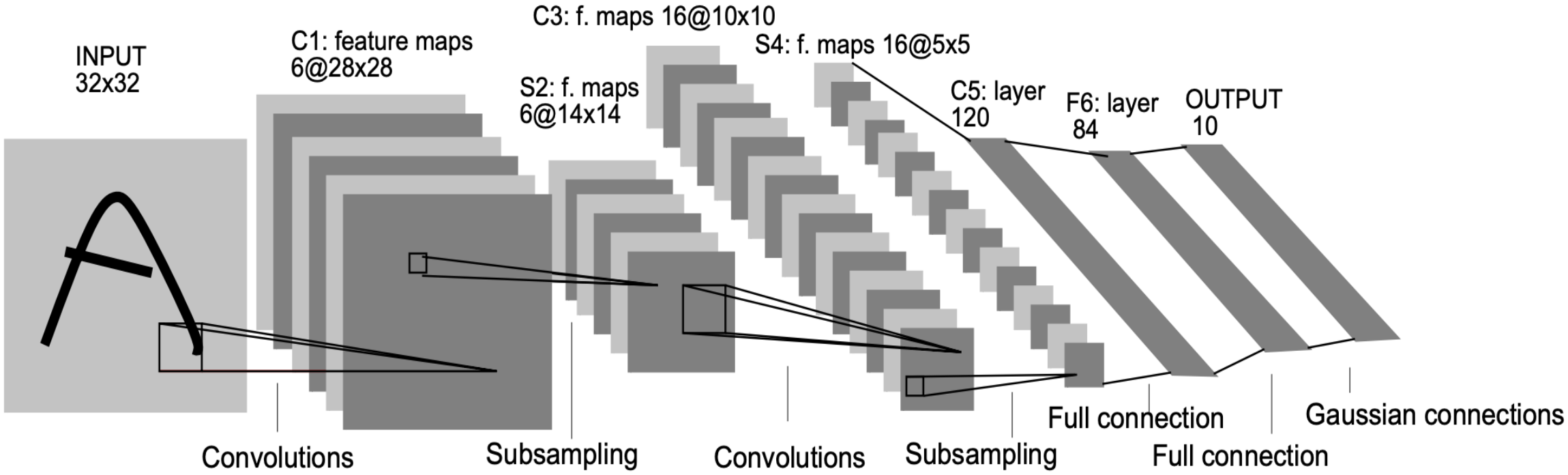
Convolutions on Multiple Input Channels

- Given multiple input channels, we can specify a filter for each one and sum the results to get a 2-D output tensor



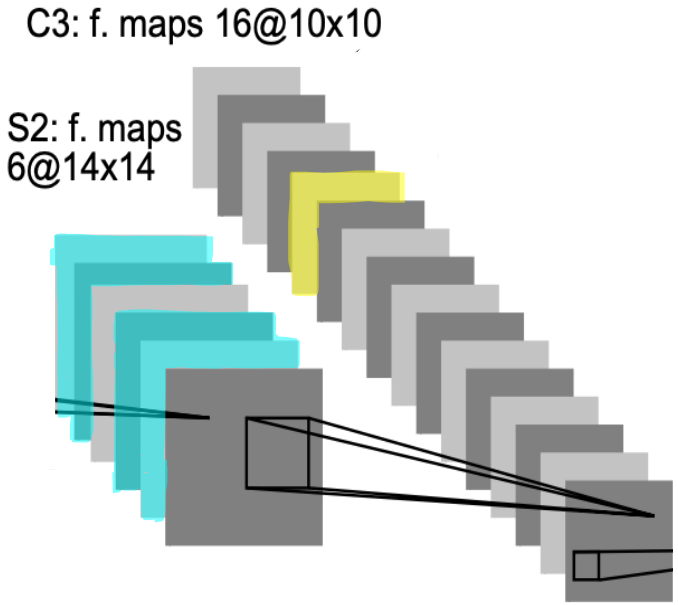
- Questions:
  1. Why might we want a different filter for each input?
  2. Why do we combine them together into a single output channel?

Source: http://preview.d2l.ai/d2l-en/master/chapter_convolutional-neural-networks/channels.html



- Channels in hidden layers correspond to different macro-features, which we might want to manipulate differently → one filter per channel

Source: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

TABLE I

EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.

- We can combine these macro-features into a new, interesting, "higher-level" feature
  - But we don't always need to combine all of them!
  - Different combinations → multiple output channels
  - Common architecture: more output channels and smaller outputs in deeper layers

Source: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf