Introduction to Neural Networks

Linear Models, MLPs, Backpropagation

EE367/CS448I: Computational Imaging

stanford.edu/class/ee367

Lecture 8

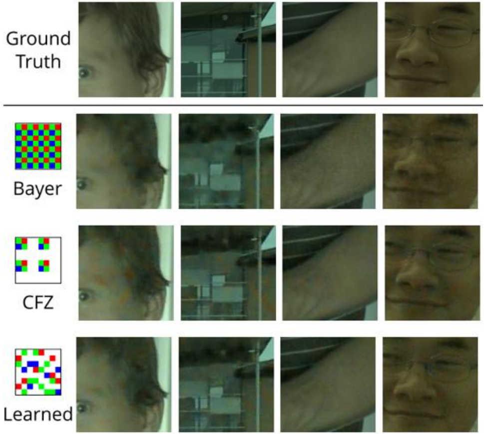
Axel Levy

Stanford University

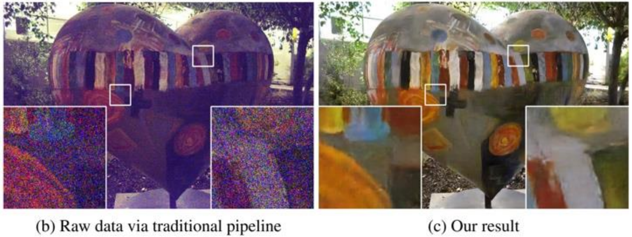
courtesy of David Lindell, adapted from Stanford CS231N

Neural Networks in Computational Imaging

- Now: learned pipelines for computational imaging



Learned demosaicking



Learning end-to-end image processing

Neural Networks in Computational Imaging

- Now: learned pipelines for computational imaging

Learned denoising

Learned deblurring

HDR Imaging

Today

- What is a neural network?
- How do we train neural networks?

Today

- What is a neural network?
- How do we train neural networks?

Wed.

- Convolutional neural networks
- Making networks deep
- Applications in denoising and deblurring

What is a neural network?

- Image classification example

Image Classification

- Image classification example

Images

MNIST Dataset

Image Classification

- Image classification example

Images

Class

“zero”
“one”

…

“nine”

Image Classification

- Image classification example

Images

What the computer “sees”

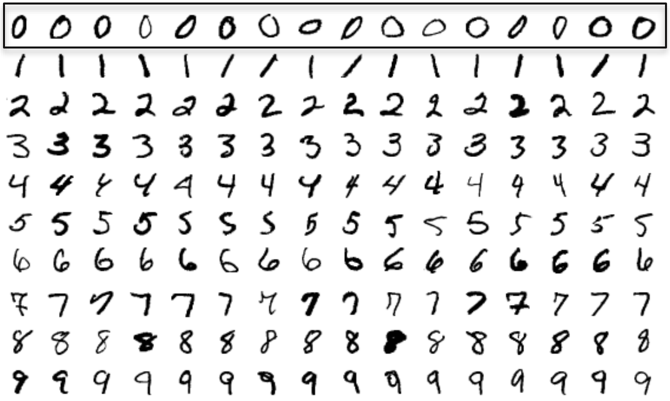
000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
000	000	000	000	000	000	000	000	019	146	146	018	000	000	000	000
000	000	000	000	000	000	000	023	188	248	236	115	003	000	000	000
000	000	000	000	000	000	059	192	249	246	112	236	086	000	000	000
000	000	000	000	000	017	214	243	145	223	064	180	113	000	000	000
000	000	000	000	014	161	232	137	006	014	000	156	209	003	000	000
000	000	000	000	108	230	042	009	000	000	000	155	227	004	000	000
000	000	000	020	223	138	000	000	000	000	000	156	220	003	000	000
000	000	000	054	237	023	000	000	000	000	024	206	111	000	000	000
000	000	000	055	210	003	000	000	001	057	216	088	006	000	000	000
000	000	000	055	242	085	022	089	160	219	108	001	000	000	000	000
000	000	000	040	233	246	226	242	175	070	001	000	000	000	000	000
000	000	000	000	058	211	212	069	001	000	000	000	000	000	000	000
000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000

000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
000	000	000	000	000	000	000	000	000	001	003	002	000	000	000	000
000	000	000	000	000	001	002	035	101	171	239	229	052	000	000	000
000	000	000	000	026	195	223	242	202	219	243	250	134	000	000	000
000	000	000	002	195	249	239	096	005	024	231	229	028	000	000	000
000	000	000	086	245	200	037	000	004	079	243	138	001	000	000	000
000	000	000	033	120	016	000	000	082	245	176	007	000	000	000	000
000	000	000	000	000	000	000	015	245	227	015	000	000	000	000	000
000	000	000	000	000	000	000	112	248	110	001	000	000	000	000	000
000	000	000	000	000	001	095	225	124	006	000	000	000	000	000	000
000	000	000	000	000	026	224	177	002	000	000	000	000	000	000	000
000	000	000	000	003	172	234	038	000	000	000	000	000	000	000	000
000	000	000	000	046	234	076	002	000	000	000	000	000	000	000	000
000	000	000	000	010	087	005	000	000	000	000	000	000	000	000	000

Image Classification

- Image classification example

Images



Challenges

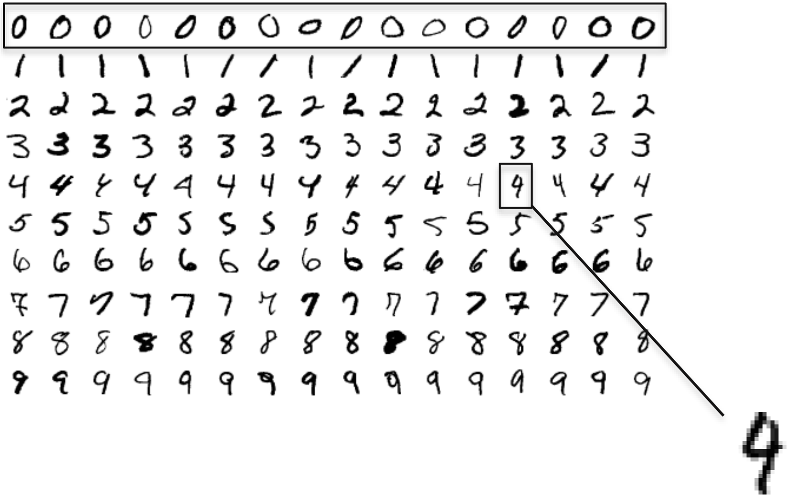
Intra-class variation

- stroke widths
- alignment
- writing styles

Image Classification

- Image classification example

Images



Challenges

Intra-class variation

- stroke widths
- alignment
- writing styles

Inter-class similarities

- “four” or “nine”?

Image Classification

- Image classification example

Images

Implementation?

```python
def classify_digit(image):
    # ???
    return image_class
```

Can't hardcode solution!

Supervised Learning

1. **Collect** training images and labels $\{x_i^{\text{tr}}\}, \{y_i^{\text{tr}}\}$

2. Define a **model** = parametric function with discretized outputs $f(x, \theta) = \hat{y}$

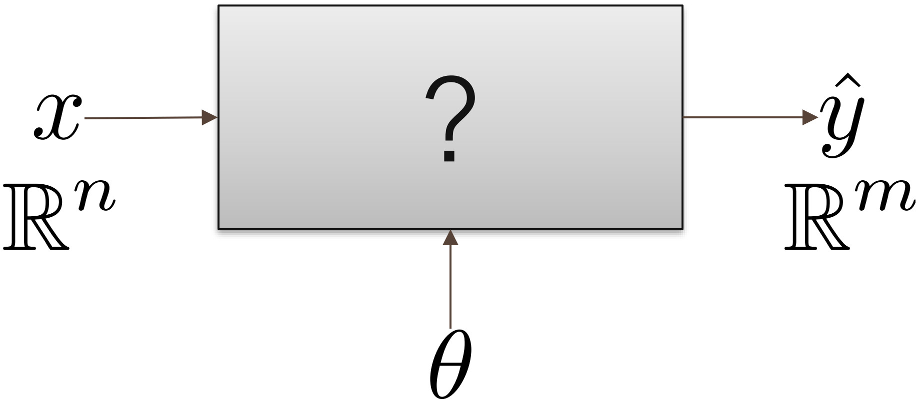
3. Define a **loss** = score function $\mathcal{L}(\{\hat{y}_i\}, \{y_i\})$

4. **Train** the model (i.e., optimize the weights) $\min_{\theta} \mathcal{L}(\{f(x_i^{\text{tr}}, \theta)\}, \{y_i^{\text{tr}}\})$

5. **Evaluate** the model on unseen images $\mathcal{L}(\{f(x_i^{\text{test}}, \theta^*)\}, \{y_i^{\text{test}}\})$

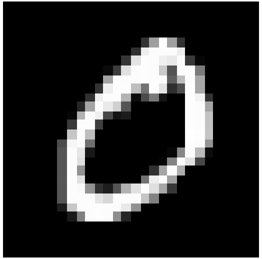
Step 2: Defining a model

If x and y are **vectors**, what is the simplest (non-trivial) “model” you can think of?



Step 2: Defining a model

- Linear Model $$f(x, W) = Wx$$

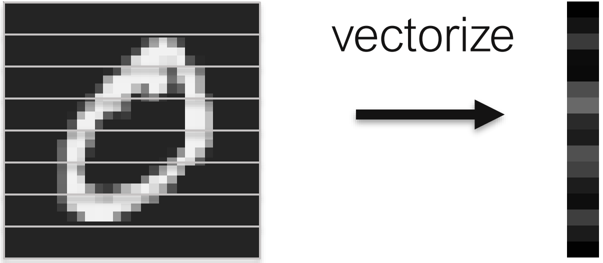


vectorize

x

Linear Model

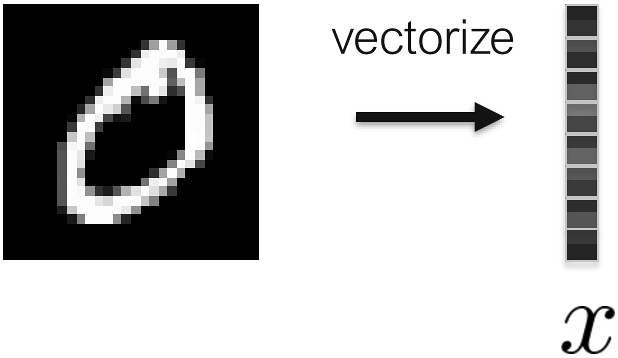
- Linear Model $$f(x, W) = Wx$$



x

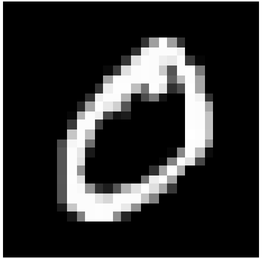
Linear Model

- Linear Model $$f(x, W) = Wx$$



Linear Model

- Linear Model $f(x, W) = Wx$



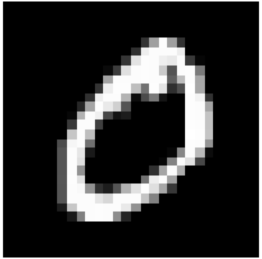
vectorize

$$x \in \mathbb{R}^N$$

Length (dimension) of this vector = number of pixels

Linear Model

- Linear Model $$f(x, W) = Wx$$



vectorize

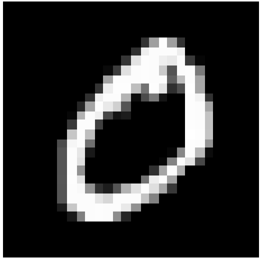
$$Wx$$

$$x \in \mathbb{R}^N \qquad W \in \mathbb{R}^{10 \times N}$$

$$\boxed{\text{In general: } Wx + b}$$

Linear Model

- Linear Model $$f(x, W) = Wx$$



vectorize

$$Wx$$

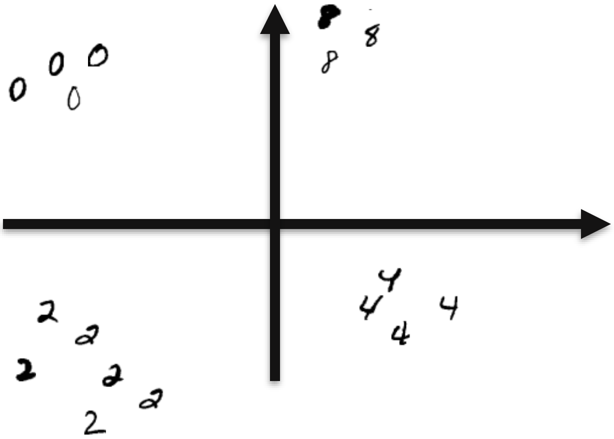
10 numbers with class scores

$$x \in \mathbb{R}^N \qquad W \in \mathbb{R}^{10 \times N}$$

Output: entry with the highest score

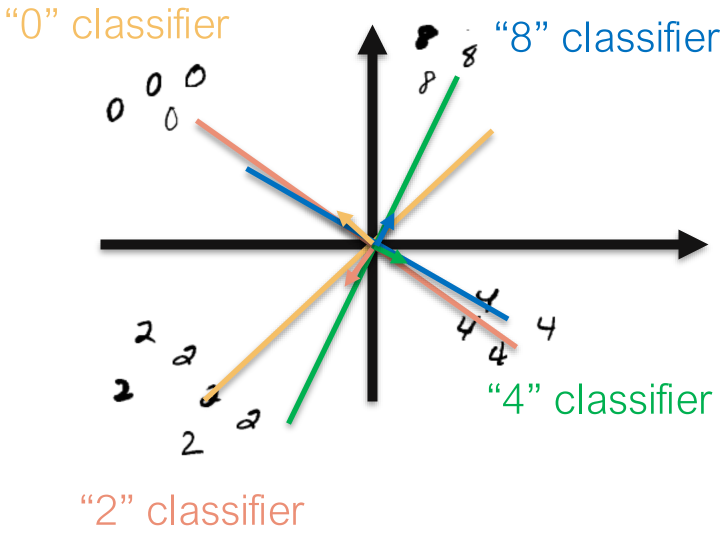
Linear Model

- Linear model: geometric interpretation



Linear Model

- Linear model: geometric interpretation



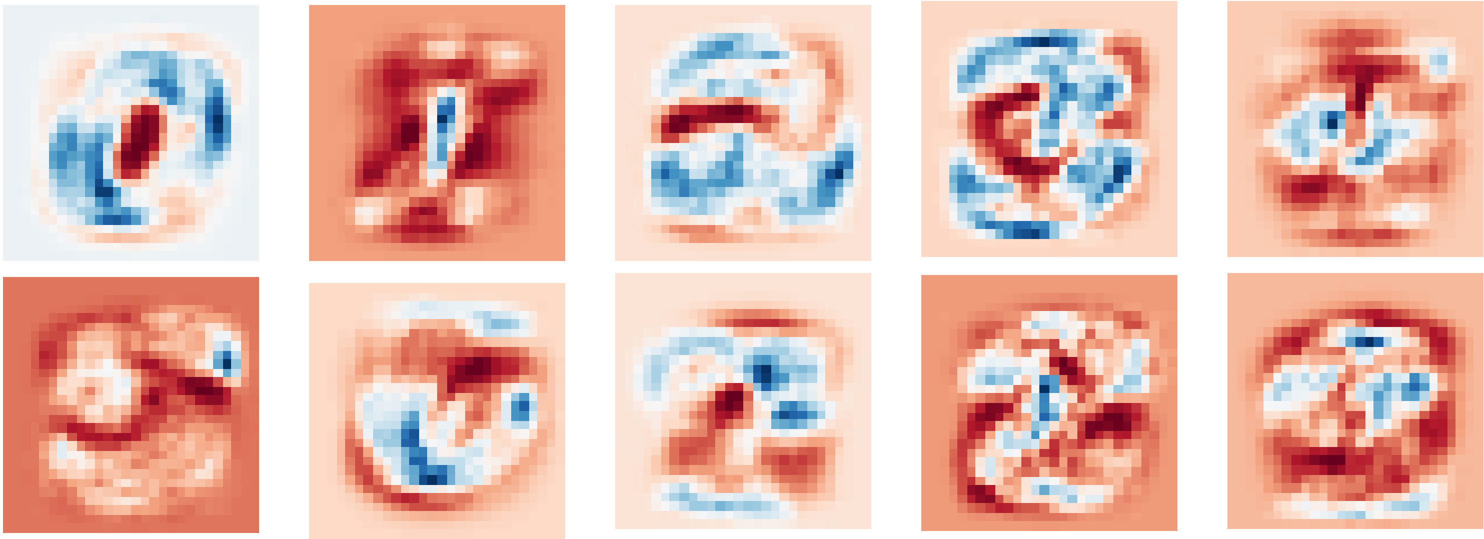
$$Wx = \begin{bmatrix} w_0 \cdot x \\ \vdots \\ w_9 \cdot x \end{bmatrix}$$

Can be seen as 10 inner products.

Linear Model

- Linear model (visual interpretation)

Learned filters (rows of W)

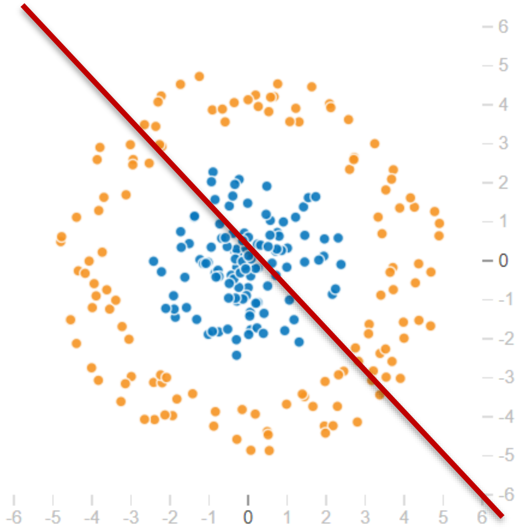


Linear Model

- Limits of linear classifiers

Linear classifiers learn linear decision planes

What if dataset is not linearly separable?



Multilayer Perceptrons (MLPs)

- Linear Model $f = Wx$
- 2-layer MLP $f = W_2 \max(0, W_1 x)$

Multilayer Perceptrons (MLPs)

- Linear Model $f = Wx$
- 2-layer MLP $f = W_2 \max(0, W_1 x)$
- 3-layer MLP $f = W_3 \max(0, W_2 \max(0, W_1 x))$

Multilayer Perceptrons (MLPs)

- Linear Model $$f = Wx$$
- 2-layer MLP $$f = W_2 \max(0, W_1 x)$$
- 3-layer MLP $$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

Non-linearity/activation function between linear layers

Multilayer Perceptrons (MLPs)

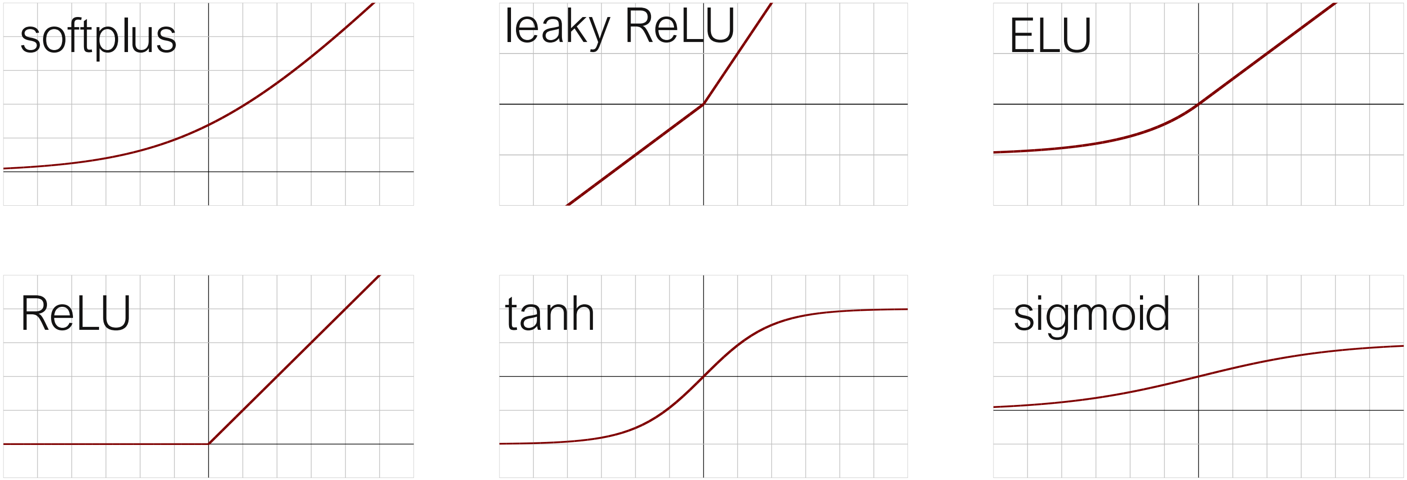
- Linear Model $f = Wx$
- 2-layer MLP $f = W_2 \max(0, W_1 x)$
- 3-layer MLP $f = W_3 \max(0, W_2 \max(0, W_1 x))$

Otherwise we have:

$$f = W_3 W_2 W_1 x$$

Activation Functions

…many to choose from

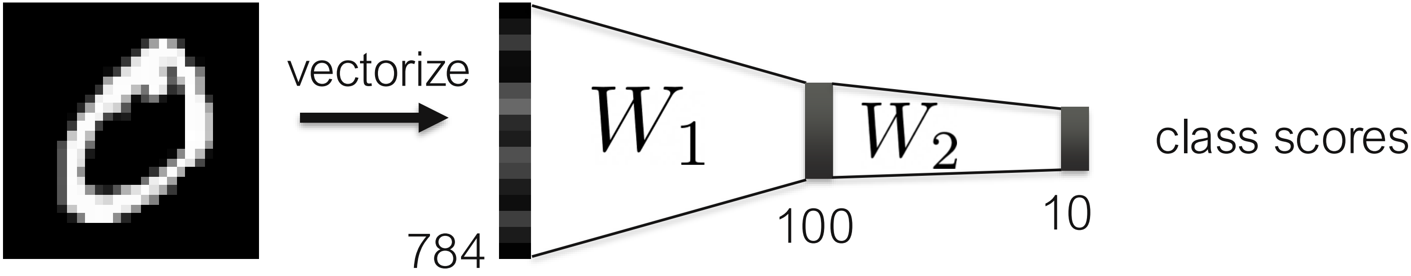


… ReLU is a good general-purpose choice: ReLU(x) = max(0, x)

Multilayer Perceptrons (MLPs)

- Linear Model $$f = Wx$$
- 2-layer MLP $$f = W_2 \max(0, W_1 x)$$

Back to our classification example…

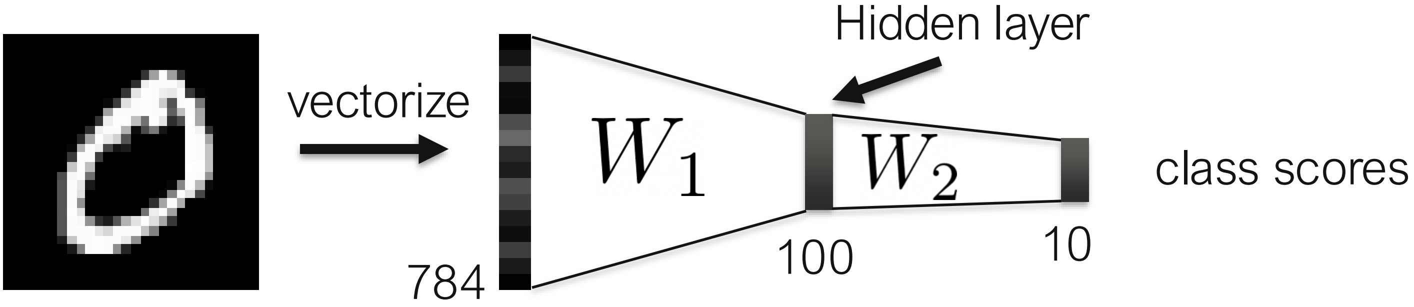


$$x \in \mathbb{R}^D, W_1 \in \mathbb{R}^{H \times D}, W_2 \in \mathbb{R}^{C \times H}$$

Multilayer Perceptrons (MLPs)

- Linear Model $$f = Wx$$
- 2-layer MLP $$f = W_2 \max(0, W_1 x)$$

Back to our classification example…

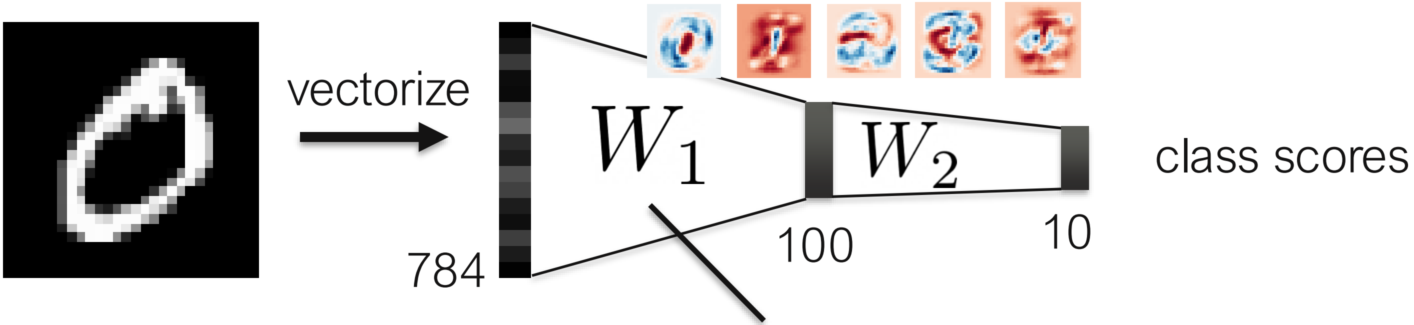


$$x \in \mathbb{R}^D, W_1 \in \mathbb{R}^{H \times D}, W_2 \in \mathbb{R}^{C \times H}$$

Multilayer Perceptrons (MLPs)

- Linear Model $f = Wx$
- 2-layer MLP $f = W_2 \max(0, W_1 x)$

Back to our classification example…



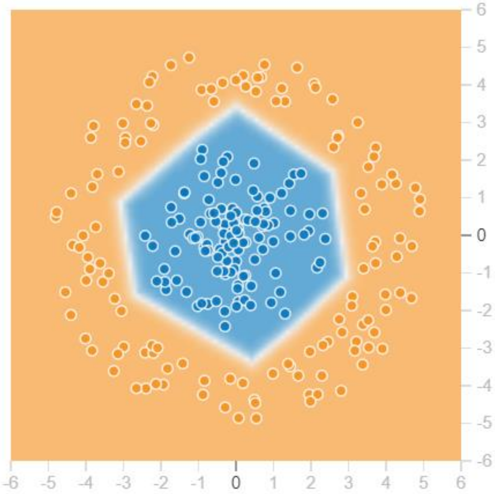
Now we have 100 shape templates, shared between classes

Multilayer Perceptrons (MLPs)

- Overcomes limits of linear classifiers

  - Can learn non-linear decision boundaries

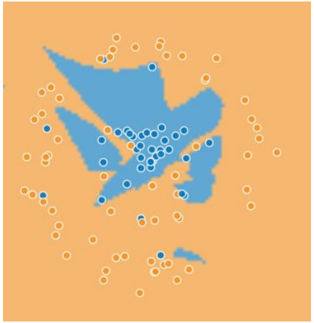
  - Complexity scales with the number of neurons/hidden layers



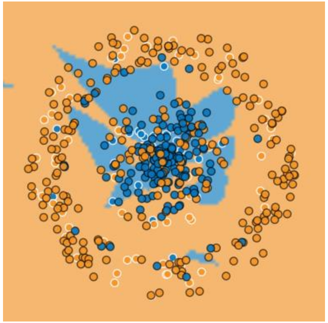
Multilayer Perceptrons (MLPs)

- More parameters is not always better!
  - Can lead to overfitting the training data
  - Performance on test data is worse

train

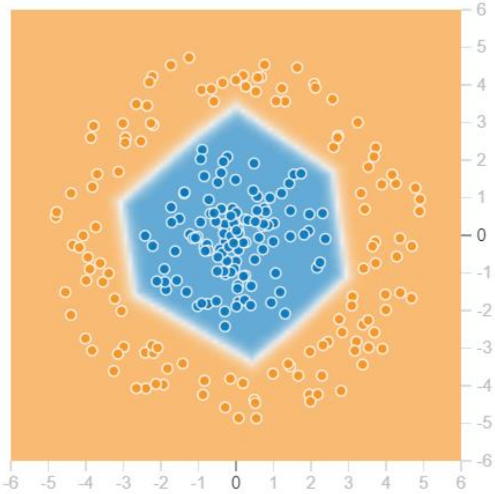


test



Multilayer Perceptrons (MLPs)

- More on classification…
  - CS231N (Deep Learning for Computer Vision)
  - CS229 (Machine Learning)



Supervised Learning

1. **Collect** training images and labels $\{x_i^{\mathrm{tr}}\}, \{y_i^{\mathrm{tr}}\}$

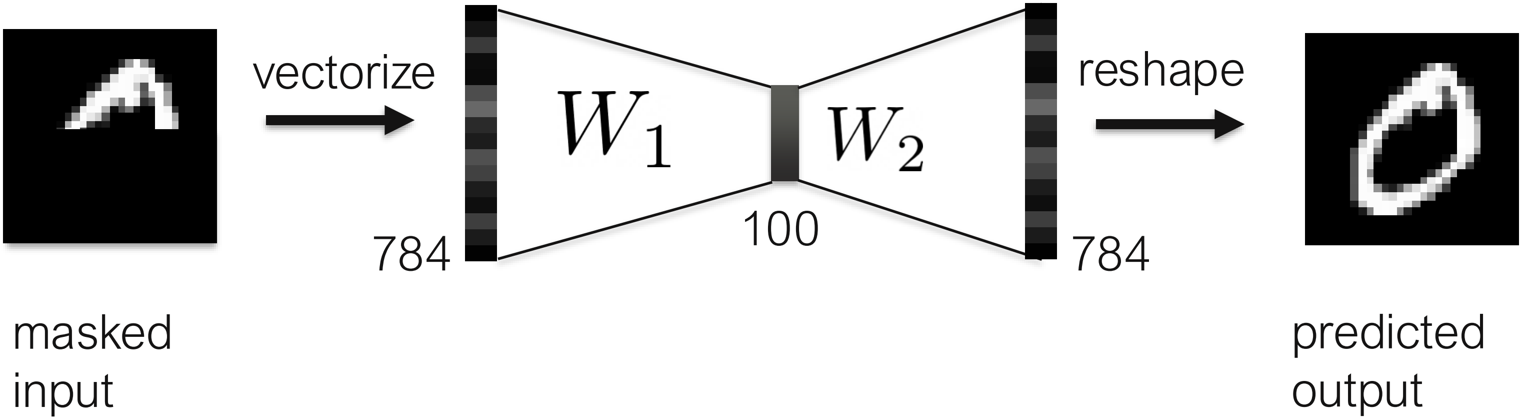
2. Define a **model** = parametric function with discretized outputs $f(x, \theta) = \hat{y}$

3. Define a **loss** = score function $\mathcal{L}(\{\hat{y}_i\}, \{y_i\})$

4. **Train** the model (i.e., optimize the weights) $\min_{\theta} \mathcal{L}(\{f(x_i^{\mathrm{tr}}, \theta)\}, \{y_i^{\mathrm{tr}}\})$

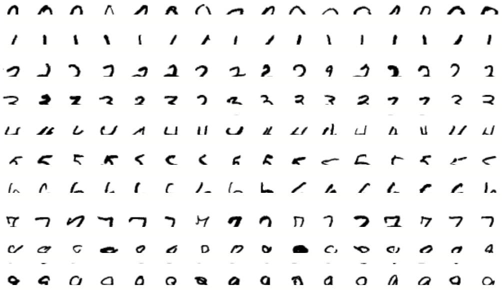
5. **Evaluate** the model on unseen images $\mathcal{L}(\{f(x_i^{\mathrm{test}}, \theta^*)\}, \{y_i^{\mathrm{test}}\})$

Image Inpainting



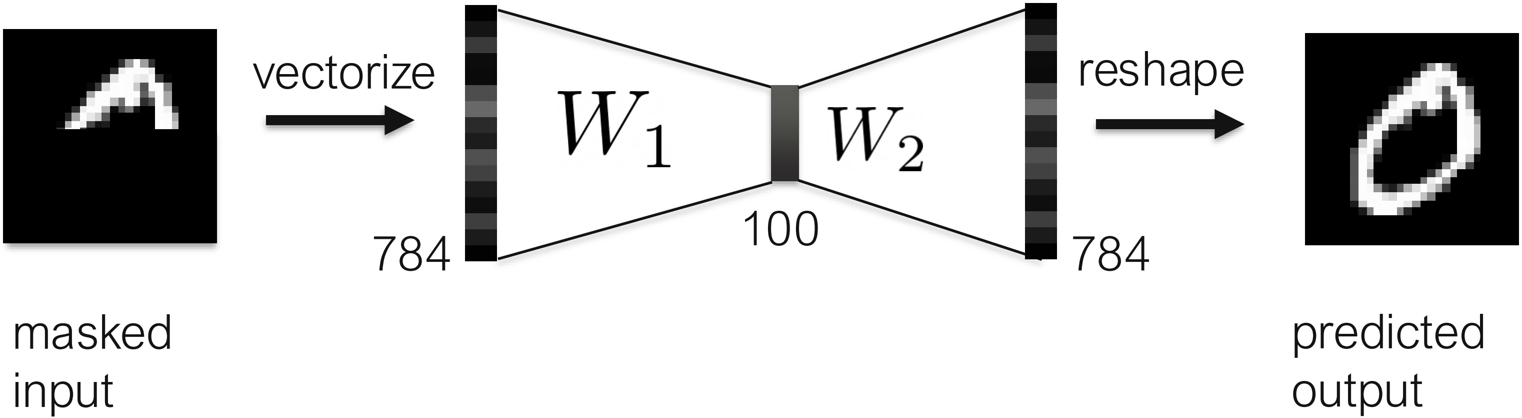
Step 1: Collect training inputs and outputs

masked images



ground truth

Step 2: Define a model



Step 3: Defining a loss

$$\mathcal{L}_\theta = \frac{1}{2}||y - \hat{y}||_2^2$$

network
parameters
$\theta = \{W_1, W_2\}$

Step 3: Defining a loss

$$\mathcal{L}_\theta = \frac{1}{2}||y - \hat{y}||_2^2$$

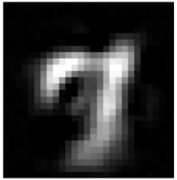
network parameters

$\theta = \{W_1, W_2\}$

ground truth image

network prediction

input

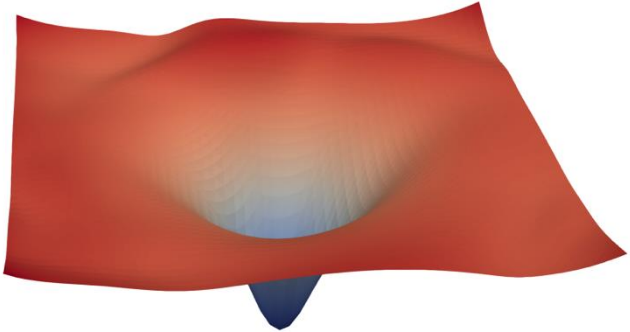


Step 4: Training the model

Gradient-based optimization

$$\nabla_\theta \mathcal{L}$$

$$\theta^{(k+1)} = \theta^{(k)} - \alpha \nabla_\theta \mathcal{L}(\theta^{(k)})$$



Loss Landscape
[Li et al. '18]

Step 4: Training the model

Need to calculate the partial derivative with respect to each parameter

$$\frac{\partial}{\partial W_1}\mathcal{L}_\theta = \frac{\partial}{\partial W_1}\frac{1}{2}\|y - \hat{y}\|_2^2$$

$$\frac{\partial}{\partial W_2}\mathcal{L}_\theta = \frac{\partial}{\partial W_2}\frac{1}{2}\|y - \hat{y}\|_2^2$$

Computing Gradients

Level 1 - “Numerical” differentiation

Level 2 - “Symbolic” differentiation

Level 3 -“Automatic” differentiation

Numerical Differentiation

$$\frac{\partial f(x)}{\partial x} \approx \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Compute this with a "small" h (e.g. 10^{-6})

Easy to implement!

Not very accurate....

Symbolic Differentiation

$$\frac{\partial \mathcal{L}_\theta}{\partial W_1} = \frac{\partial}{\partial W_1}\frac{1}{2}\|y - \hat{y}\|_2^2$$

$$= \frac{\partial}{\partial W_1}\frac{1}{2}(y - W_2\sigma(W_1 x))^T \cdot (y - W_2\sigma(W_1 x))$$

chain rule, product rule…

Accurate

Tedious (must be manually calculated for each term)

Automatic Differentiation

Think about the problem as a “computational graph”

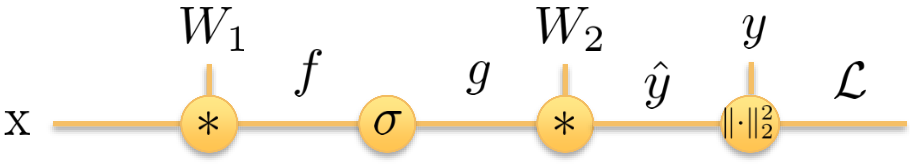
“Divide and conquer” using the chain rule

Enables “backpropagation” – an efficient way to take derivatives of the loss w.r.t. all the parameters in the graph

Automatic Differentiation

Think about the problem as a "computational graph"

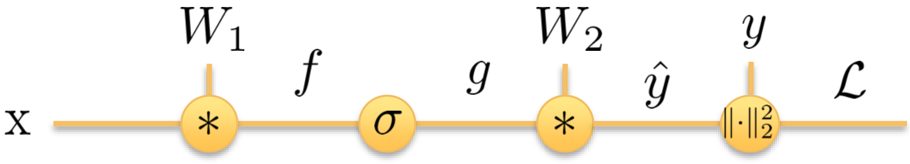
Divide and conquer using the chain rule



Automatic Differentiation

Think about the problem as a "computational graph"

Divide and conquer using the chain rule

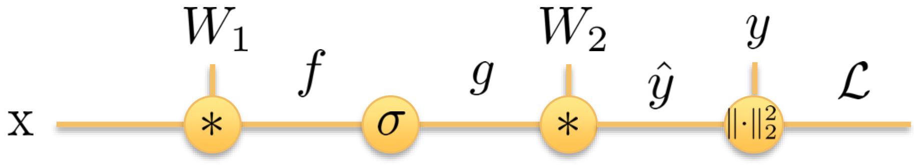


$$\frac{\partial \mathcal{L}}{\partial W_2} = \frac{\partial \hat{y}}{\partial W_2} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Automatic Differentiation

Think about the problem as a "computational graph"

Divide and conquer using the chain rule

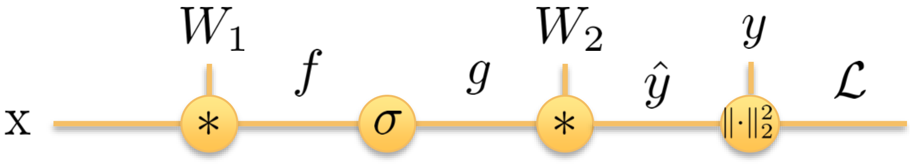


$$\frac{\partial \mathcal{L}}{\partial W_1} = \frac{\partial f}{\partial W_1} \frac{\partial g}{\partial f} \frac{\partial \hat{y}}{\partial g} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Automatic Differentiation

Think about the problem as a "computational graph"

Divide and conquer using the chain rule

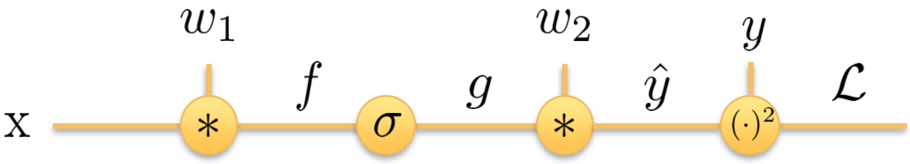


$$\frac{\partial \mathcal{L}}{\partial W_1} = \frac{\partial f}{\partial W_1} \frac{\partial g}{\partial f} \frac{\partial \hat{y}}{\partial g} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

We can calculate analytical expressions for each of these terms and then plug in our values

Autodiff Example

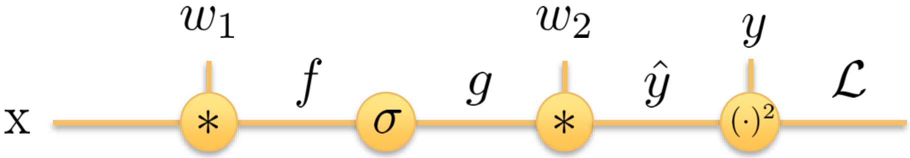
(assume scalar values for now)



$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial f}{\partial w_1} \frac{\partial g}{\partial f} \frac{\partial \hat{y}}{\partial g} \boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}}}$$

Autodiff Example

(assume scalar values for now)

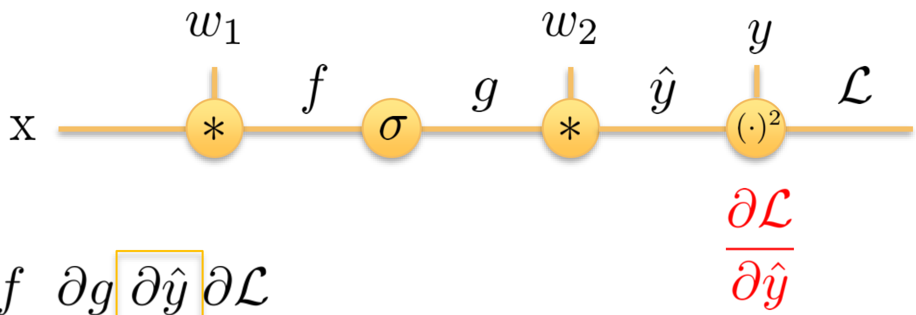


$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial f}{\partial w_1}\frac{\partial g}{\partial f}\frac{\partial \hat{y}}{\partial g}\boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}}}$$

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}}\frac{1}{2}(\hat{y}-y)^2 = \hat{y}-y$$

Autodiff Example

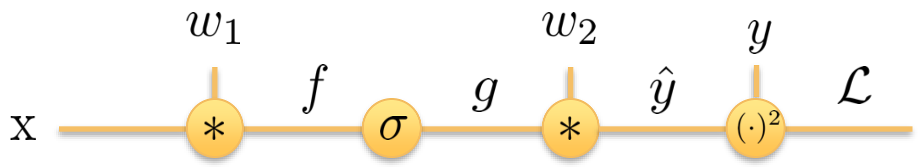
(assume scalar values for now)



$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial f}{\partial w_1} \frac{\partial g}{\partial f} \boxed{\frac{\partial \hat{y}}{\partial g}} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Autodiff Example

(assume scalar values for now)



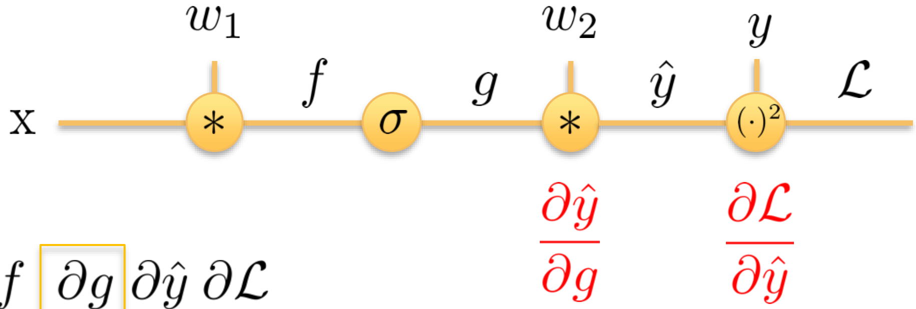
$$\frac{\partial \mathcal{L}}{\partial \hat{y}}$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial f}{\partial w_1} \frac{\partial g}{\partial f} \boxed{\frac{\partial \hat{y}}{\partial g}} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

$$\frac{\partial \hat{y}}{\partial g} = \frac{\partial}{\partial g} w_2 \cdot g = w_2$$

Autodiff Example

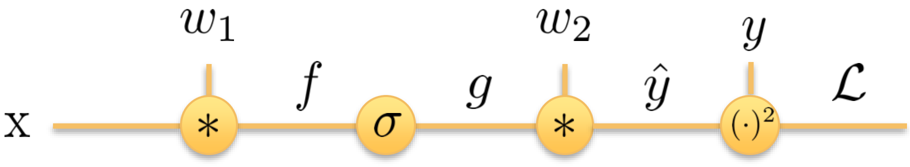
(assume scalar values for now)



$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial f}{\partial w_1} \boxed{\frac{\partial g}{\partial f}} \frac{\partial \hat{y}}{\partial g} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Autodiff Example

(assume scalar values for now)



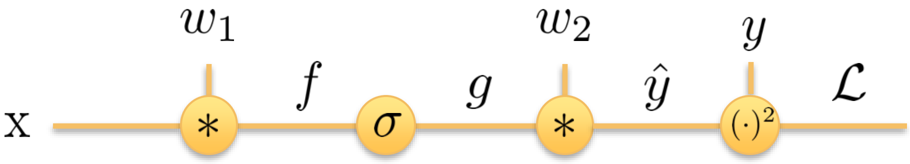
$$\frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial f}{\partial w_1} \boxed{\frac{\partial g}{\partial f}} \frac{\partial \hat{y}}{\partial g} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

$$\frac{\partial g}{\partial f} = \frac{\partial}{\partial f}\sigma(f) = \frac{\partial}{\partial f}\max(0, f) = \begin{cases} 0, & f < 0 \\ 1 & \text{else} \end{cases}$$

Autodiff Example

(assume scalar values for now)

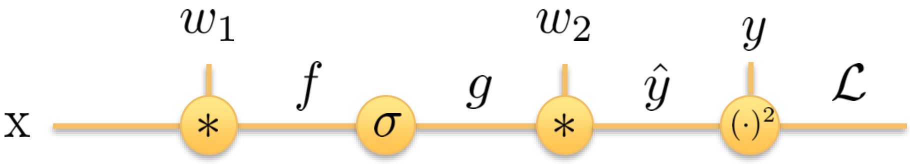


$$\frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \boxed{\frac{\partial f}{\partial w_1}} \frac{\partial g}{\partial f} \frac{\partial \hat{y}}{\partial g} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Autodiff Example

(assume scalar values for now)



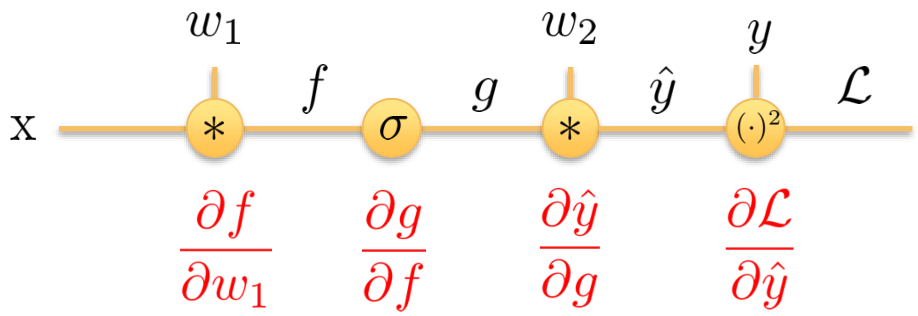
$$\frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \boxed{\frac{\partial f}{\partial w_1}} \frac{\partial g}{\partial f} \frac{\partial \hat{y}}{\partial g} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

$$\frac{\partial f}{\partial w_1} = \frac{\partial}{\partial w_1} w_1 \cdot x = x$$

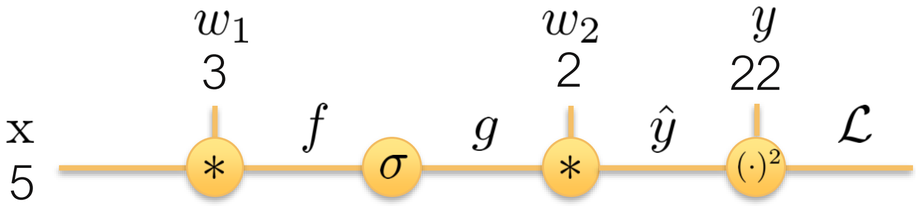
Autodiff Example

(assume scalar values for now)



Autodiff Example

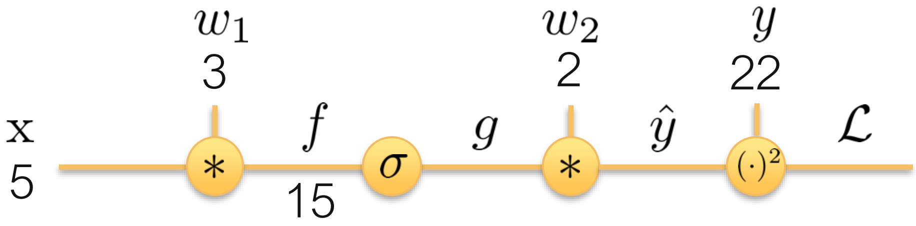
Let's plug in the values now…



$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Autodiff Example

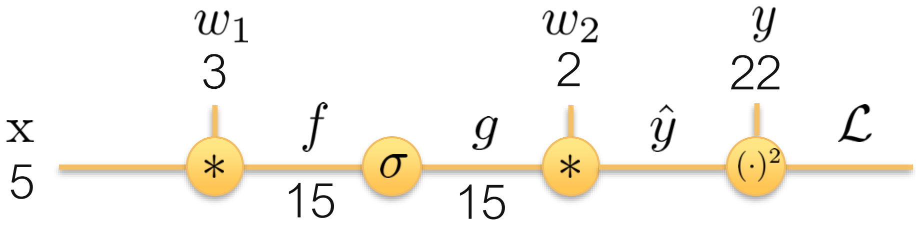
Let’s plug in the values now…



$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Autodiff Example

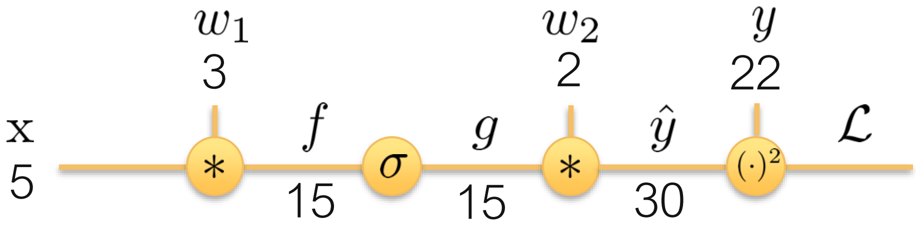
Let’s plug in the values now…



$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Autodiff Example

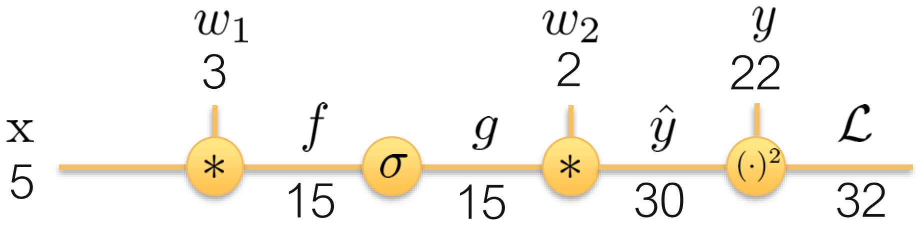
Let's plug in the values now…



$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Autodiff Example

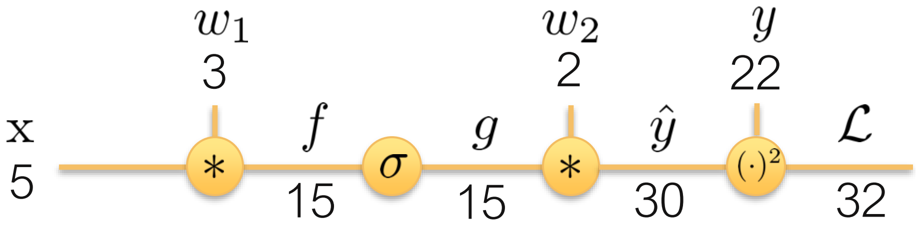
Let's plug in the values now…



$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Autodiff Example

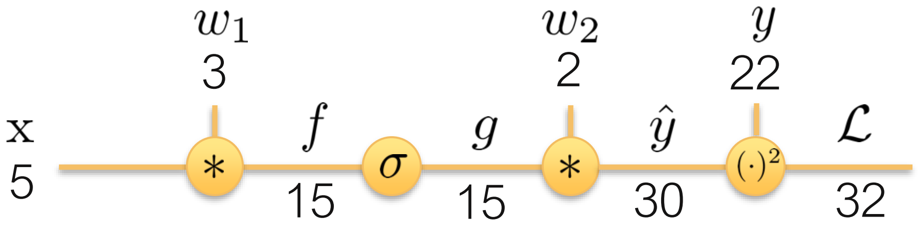
Let's plug in the values now…



$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$

Autodiff Example

Let's plug in the values now…

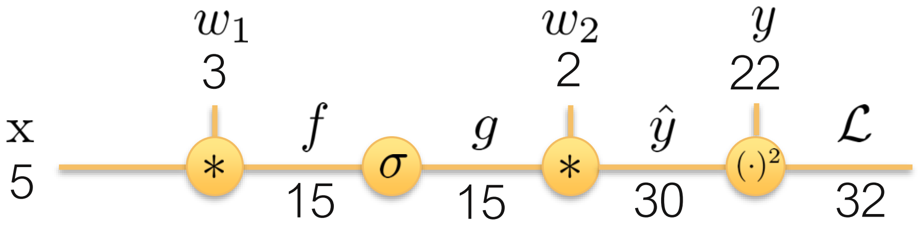


$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$

8

Autodiff Example

Let's plug in the values now…



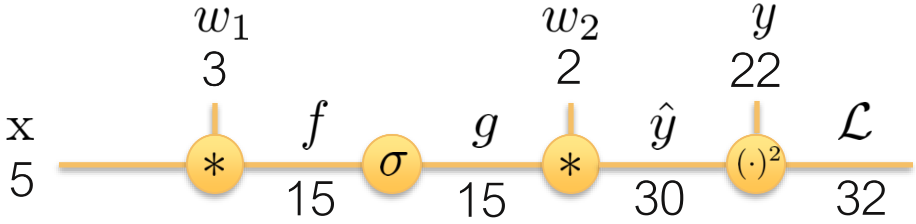
$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$

$\frac{\partial \hat{y}}{\partial g}$: 2

$\frac{\partial \mathcal{L}}{\partial \hat{y}}$: 8

Autodiff Example

Let's plug in the values now…

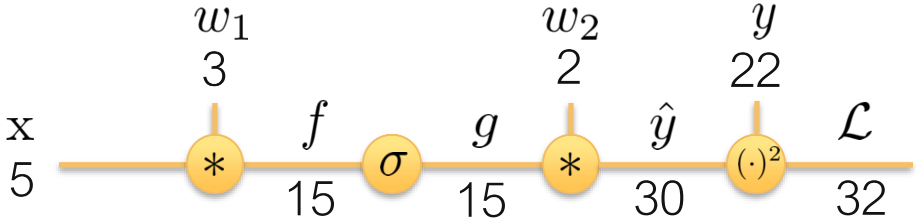


$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$

1   2   8

Autodiff Example

Let’s plug in the values now…

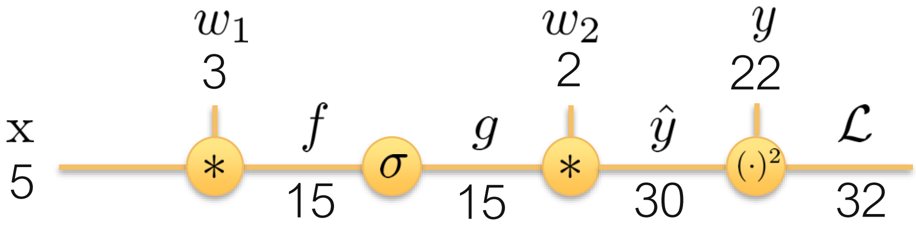


$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$

5 1 2 8

Autodiff Example

Let’s plug in the values now…

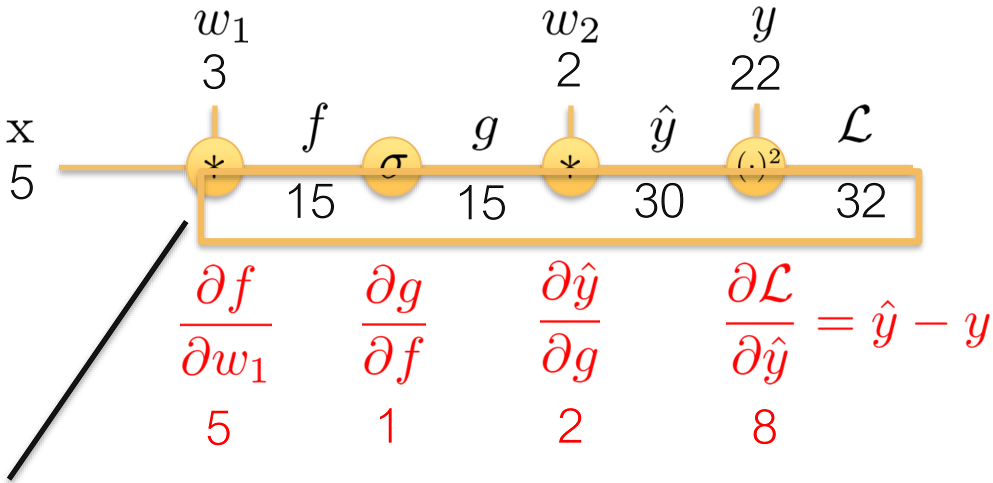


$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$

5 1 2 8

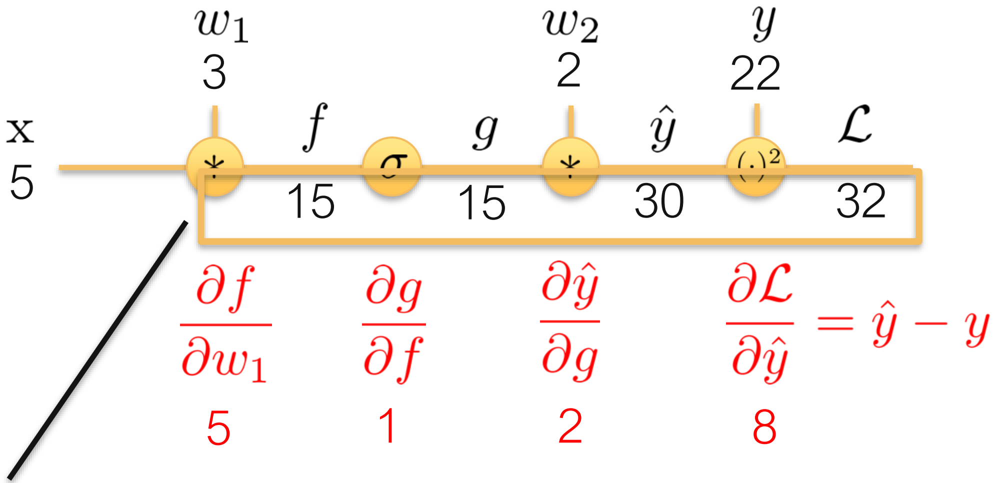
$$\frac{\partial \mathcal{L}}{\partial w_1} = 80$$

Autodiff Example



Save these intermediate values during forward computation

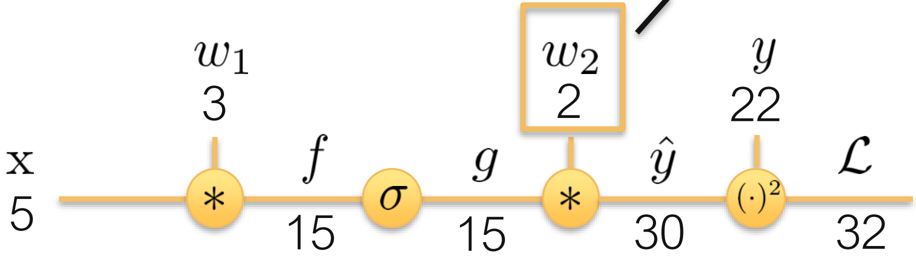
Autodiff Example



Then we perform a "backward pass"

Autodiff Example

What about $\frac{\partial \mathcal{L}}{\partial w_2}$?



$\frac{\partial f}{\partial w_1}$ 5

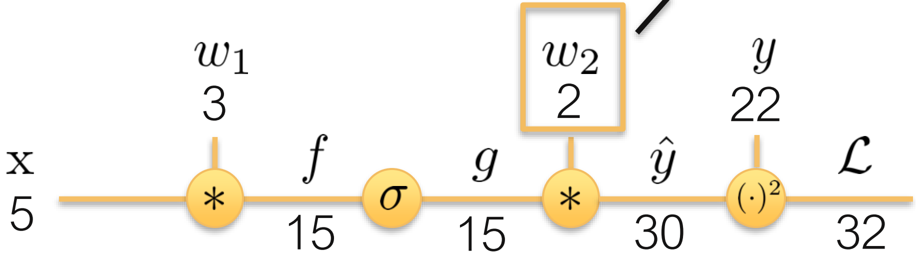
$\frac{\partial g}{\partial f}$ 1

$\frac{\partial \hat{y}}{\partial g}$ 2

$\frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$ 8

Autodiff Example

What about $\frac{\partial \mathcal{L}}{\partial w_2}$?

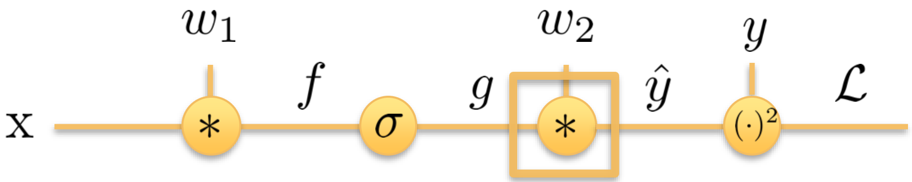


$\frac{\partial f}{\partial w_1}$ 5 $\quad$ $\frac{\partial g}{\partial f}$ 1 $\quad$ $\frac{\partial \hat{y}}{\partial g}$ 2 $\quad$ $\boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}}} = \hat{y} - y$

We can re-use computation!

$$\frac{\partial \mathcal{L}}{\partial w_2} = \frac{\partial \hat{y}}{\partial w_2} \boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}}}$$

Autodiff Example



PyTorch Code:

```
class Multiply(torch.autograd.Function):
  @staticmethod
  def forward(ctx, x, y):
    ctx.save_for_backward(x, y)
    z = x * y
    return z
  @staticmethod
  def backward(ctx, grad_z):
    x, y = ctx.saved_tensors
    grad_x = y * grad_z    # dz/dx * dL/dz
    grad_y = x * grad_z    # dz/dy * dL/dz
    return grad_x, grad_y
```

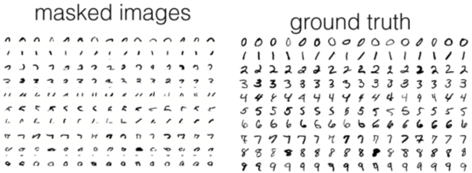
Need to stash some values for use in backward

Upstream gradient

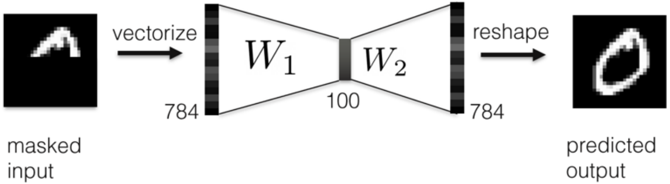
Multiply upstream and local gradients

Image Inpainting Training Loop

1. Sample batch of images from dataset



2. Run forward pass to calculate network output for each image



3. Run backward pass to calculate gradients with backpropagation

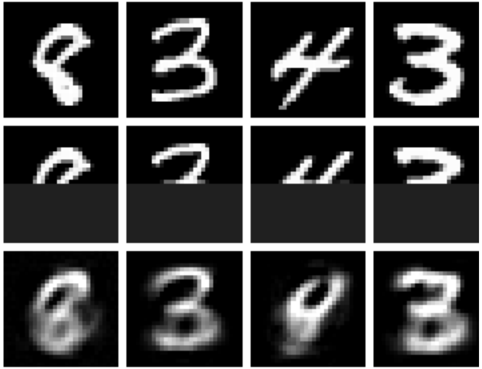
4. Update parameters with stochastic gradient descent

4. Update parameters with stochastic gradient descent

$$\nabla_\theta \mathcal{L} = \left(\frac{\partial \mathcal{L}}{\partial W_1}, \frac{\partial \mathcal{L}}{\partial W_2} \right)$$

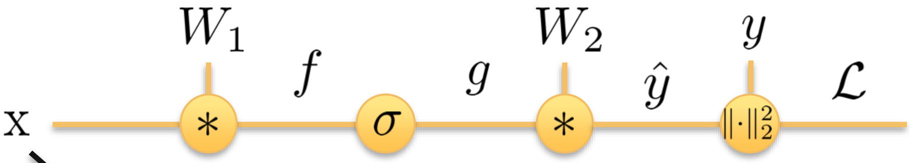
$$W_2^{(k+1)} = W_2^{(k)} - \alpha \frac{\partial \mathcal{L}}{\partial W_2}$$

$$W_1^{(k+1)} = W_1^{(k)} - \alpha \frac{\partial \mathcal{L}}{\partial W_1}$$



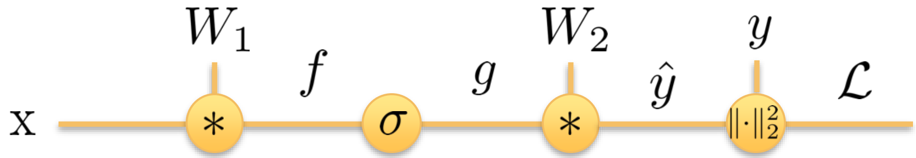
“stochastic” refers to the fact that inputs are processed in batches

Vector Differentiation



But wait, aren't these vectors?

Vector Differentiation



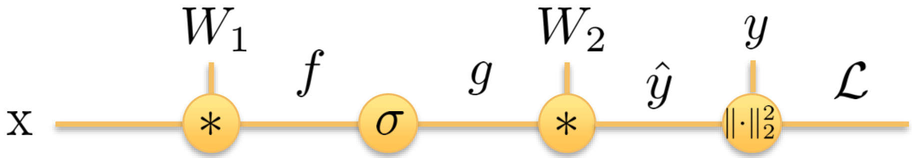
Recap: vector differentiation

Scalar wrt Scalar

$$x \in \mathbb{R} \; y \in \mathbb{R}$$

$$\frac{\partial y}{\partial x} \in \mathbb{R}$$

Vector Differentiation



Recap: vector differentiation

Scalar wrt Scalar

$$x \in \mathbb{R} \; y \in \mathbb{R}$$

$$\frac{\partial y}{\partial x} \in \mathbb{R}$$

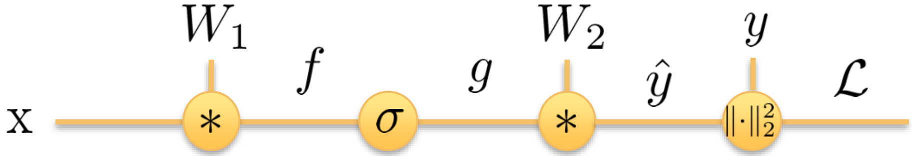
Vector wrt Vector

$$x \in \mathbb{R}^N \quad y \in \mathbb{R}^M$$

$$\frac{\partial y}{\partial x} \in \mathbb{R}^{M \times N}$$

“output x input”

Vector Differentiation



Recap: vector differentiation

$$\frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_N} \\ & \vdots & \\ \frac{\partial y_M}{\partial x_1} & \cdots & \frac{\partial y_M}{\partial x_N} \end{bmatrix}$$

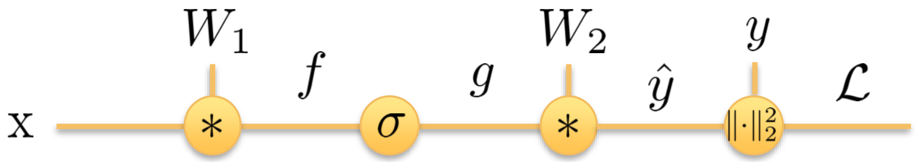
Vector wrt Vector

$x \in \mathbb{R}^N \quad y \in \mathbb{R}^M$

$$\frac{\partial y}{\partial x} \in \mathbb{R}^{M \times N}$$

"output x input"

Recap: vector differentiation



Example 1: matrix multiply

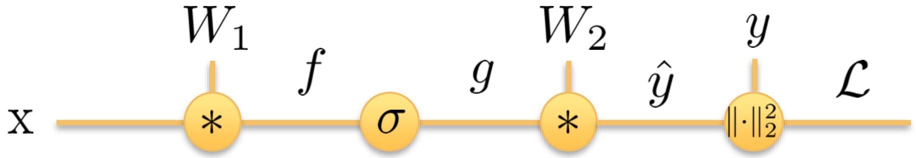
$$\frac{\partial \hat{y}}{\partial g} = \frac{\partial}{\partial g} W_2 g$$

$$g \in \mathbb{R}^N$$

$$\hat{y} \in \mathbb{R}^M$$

$$W_2 \in \mathbb{R}^{M \times N}$$

Recap: vector differentiation



Example 1: matrix multiply

$$\frac{\partial \hat{y}}{\partial g} = \frac{\partial}{\partial g} W_2 g$$

$$g \in \mathbb{R}^N$$

$$\hat{y} \in \mathbb{R}^M$$

$$W_2 \in \mathbb{R}^{M \times N}$$

$$\frac{\partial \hat{y}}{\partial g} = W_2$$

Recap: vector differentiation

Example 2: elementwise functions

$$h = f \odot g$$

$$f \in \mathbb{R}^N$$

$$h \in \mathbb{R}^N$$

$$\frac{\partial h}{\partial f} \in \mathbb{R}^{N \times N}$$

Recap: vector differentiation

Example 2: elementwise functions

$$h = f \odot g$$

$$f \in \mathbb{R}^N$$

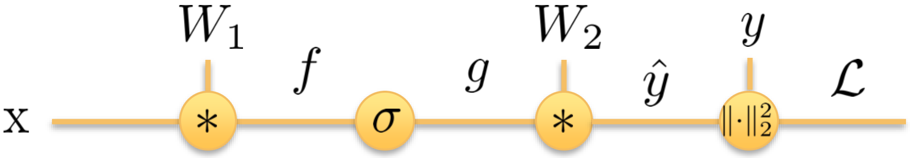
$$h \in \mathbb{R}^N$$

$$\frac{\partial h}{\partial f} \in \mathbb{R}^{N \times N}$$

$$\frac{\partial h}{\partial f} = \begin{bmatrix} g_1 & & 0 \\ & \ddots & \\ 0 & & g_n \end{bmatrix} = \mathrm{diag}(g)$$

Recap: vector differentiation

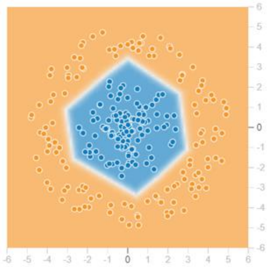
Final hint: dimensions should always match up!



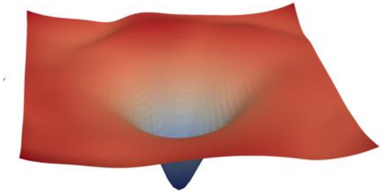
$$\frac{\partial \mathcal{L}}{\partial W_1} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial f} \cdot \frac{\partial f}{\partial W_1}$$

⚠ The product would be flipped with the “input x output” convention

Summary



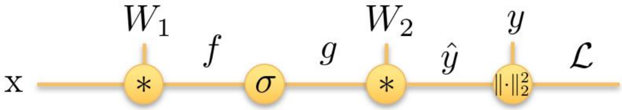
Linear models and MLPs



Gradient descent

Automatic differentiation, backpropagation

Computational graphs

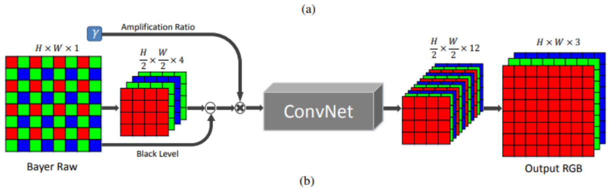


Next Time

Convolutional neural networks

Building blocks of deep networks

Image processing with deep networks



(b) Raw data via traditional pipeline

(c) Our result

References and Further Reading

slides adapted from Stanford CS231N: http://cs231n.stanford.edu/slides/

CS229/CS231n notes on linear classifiers

https://cs231n.github.io/linear-classify/

https://cs229.stanford.edu/notes2021fall/cs229-notes1.pdf

CS231n Notes on backprop

http://cs231n.stanford.edu/handouts/linear-backprop.pdf

https://cs231n.github.io/optimization-2/

Intro to pytorch autograd

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Extending pytorch autograd functions

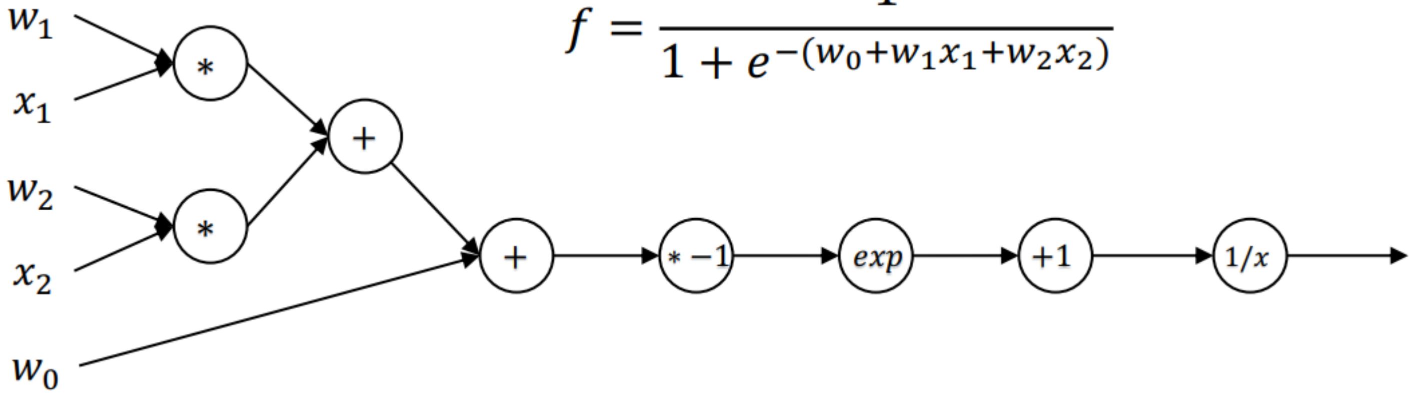
https://pytorch.org/docs/stable/notes/extending.html

Extra backpropagation example

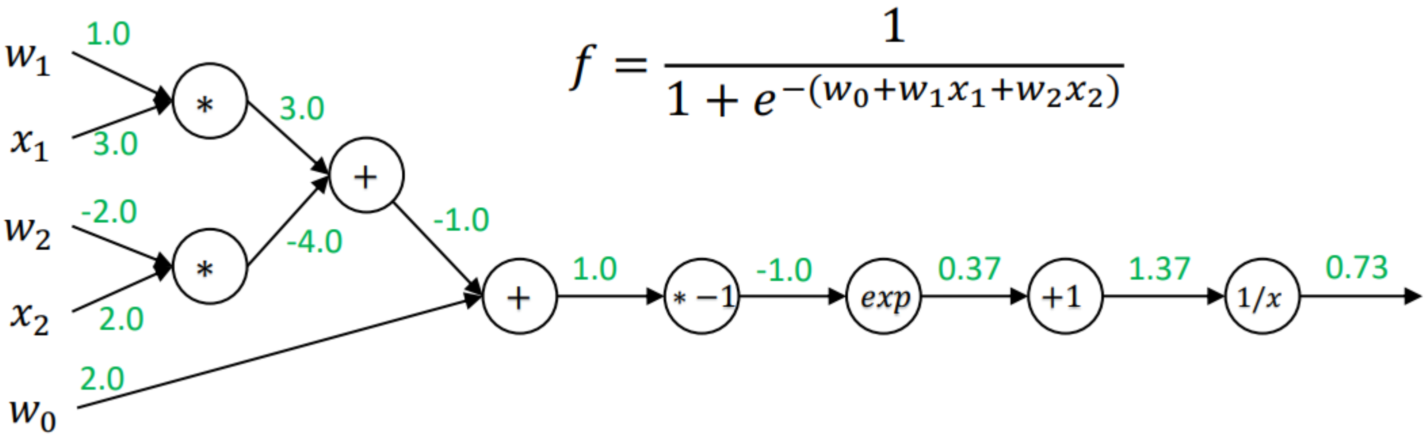
$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$

Extra backpropagation example

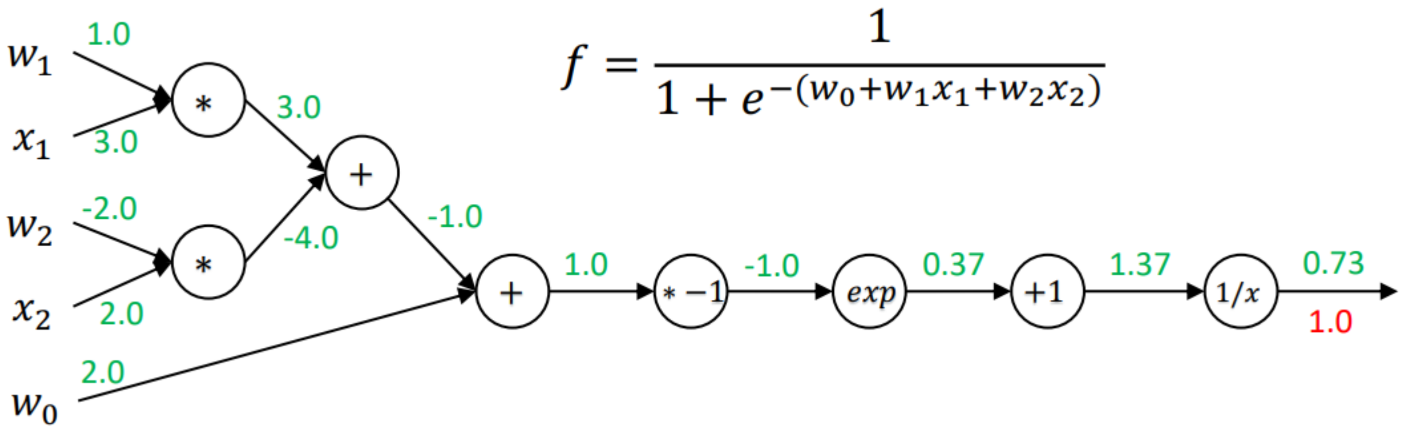
$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$



Extra backpropagation example

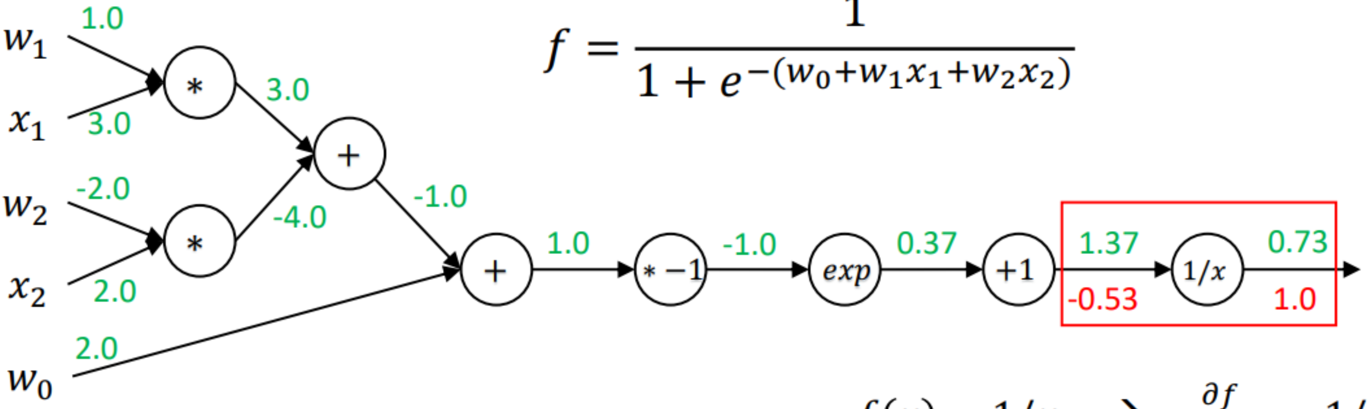


Extra backpropagation example



Extra backpropagation example

$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$

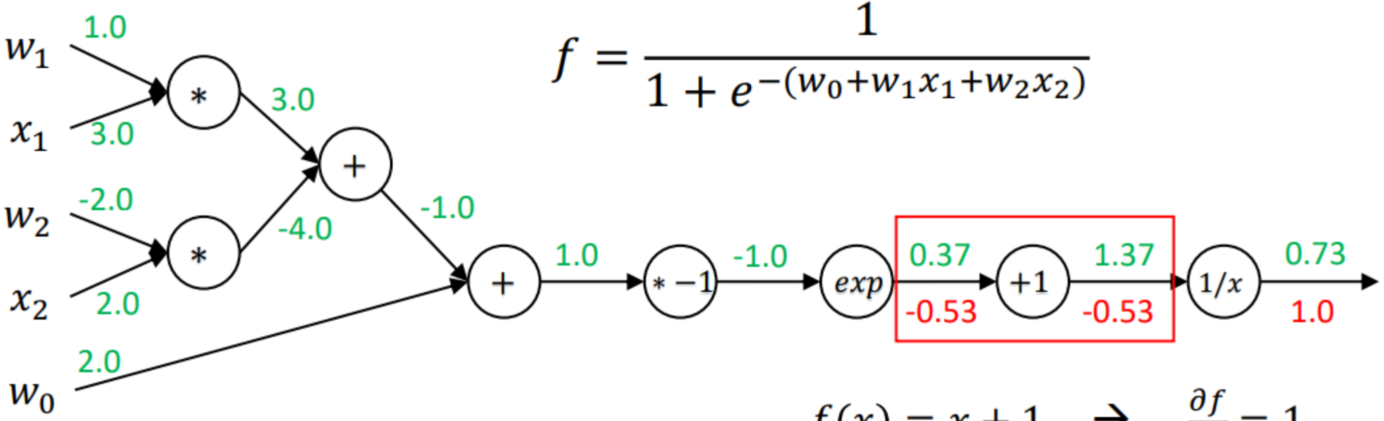


$$f(x) = 1/x \quad \rightarrow \quad \frac{\partial f}{\partial x} = -1/x^2$$

$$\frac{\partial J}{\partial x} = \frac{\partial J}{\partial f}\frac{\partial f}{\partial x} = -1/x^2$$

Extra backpropagation example

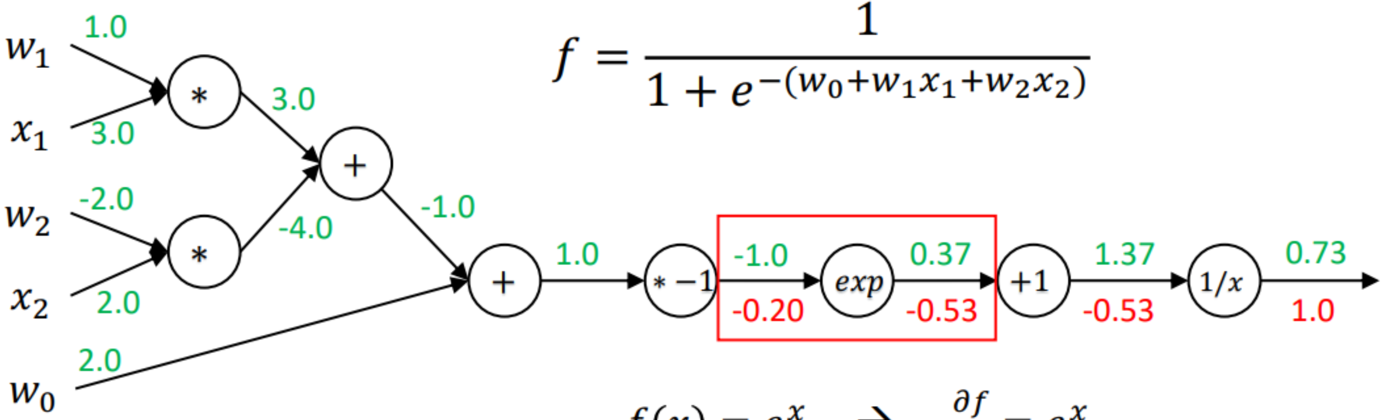
$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$



$$f(x) = x + 1 \quad \rightarrow \quad \frac{\partial f}{\partial x} = 1$$

Extra backpropagation example

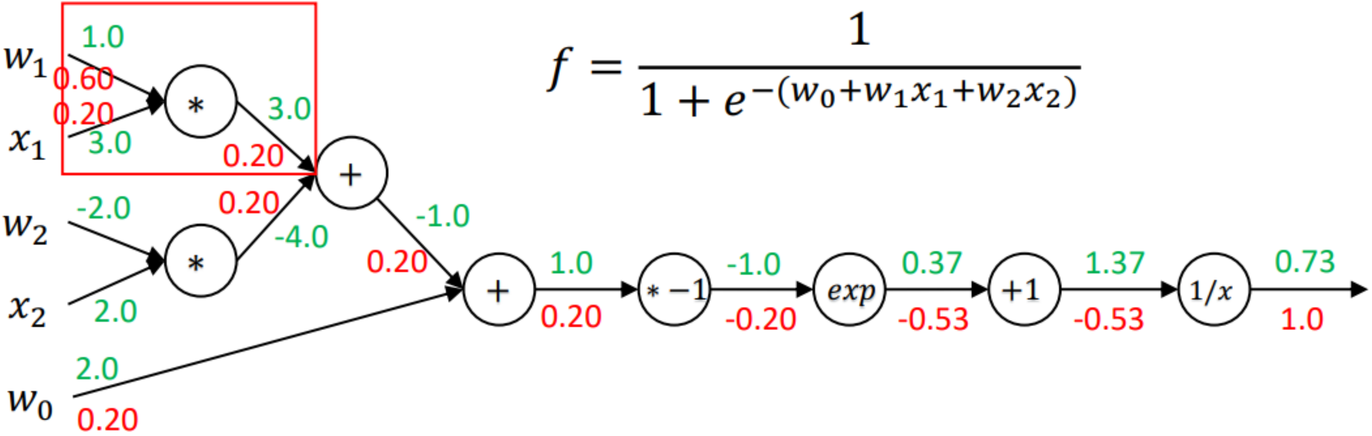
$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$



$$f(x) = e^x \quad \rightarrow \quad \frac{\partial f}{\partial x} = e^x$$

$$\frac{\partial J}{\partial x} = \frac{\partial J}{\partial f}\frac{\partial f}{\partial x} = \frac{\partial J}{\partial f} \cdot e^x$$

Extra backpropagation example



$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$

$$f(x, w) = xw \quad \rightarrow \quad \frac{\partial f}{\partial x} = w, \; \frac{\partial f}{\partial w} = x$$

Extra backpropagation example

$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$