CS480/680: Introduction to Machine Learning

Lecture 9: Multilayer Perceptron (MLP)

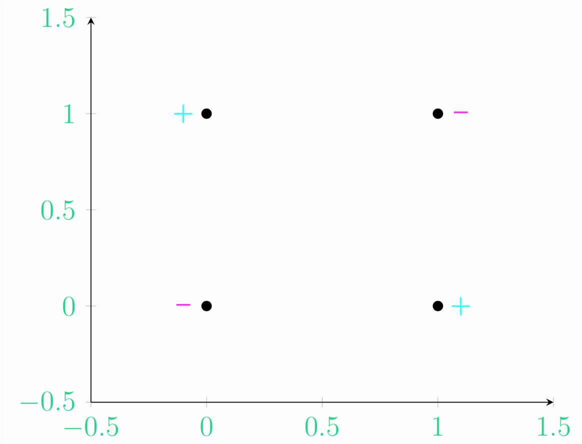
Hongyang Zhang

UNIVERSITY OF WATERLOO

Feb 8, 2024

XOR Recalled

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$
	0	1	0	1
	0	0	1	1
$\mathbf{y}$	$-$	$+$	$+$	$-$

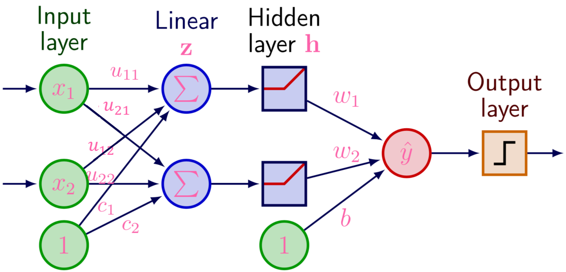


- We have shown no linear classifier can separate the data.

Fixing the Problem

- A linear classifier in the input space underfits the XOR data
- Can fix input representation but use a richer model (e.g., a quadratic classifier)
- Can fix the classifier as linear but use a richer input representation (the power of lifting)
- The two approaches are equivalent for certain classifiers, by a reproducing kernel
- Neural network: learn the feature map simultaneously with the linear classifier!

Multi-Layer Perception (MLP)



- 1st linear transformation: $\mathbf{z} = \mathbf{U}\mathbf{x} + \mathbf{c}$, $\quad \mathbf{U} \in \mathbb{R}^{2\times 2}$, $\mathbf{c} \in \mathbb{R}^2$
- Element-wise nonlinear activation: $\mathbf{h} = \sigma(\mathbf{z})$
- 2nd linear transformation: $\hat{y} = \langle \mathbf{h}, \mathbf{w} \rangle + b$, $\quad \mathbf{w} \in \mathbb{R}^2$, $b \in \mathbb{R}$
- Output layer: $\mathrm{sign}(\hat{y})$ or sigmoid$(\hat{y})$
- BLUE: parameters to be learned

Does It Work on XOR Dataset?

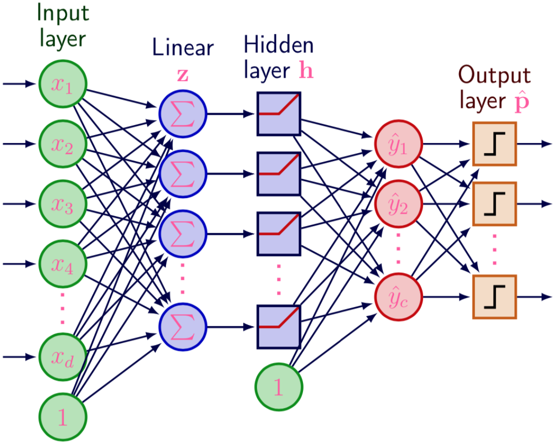
Consider a well-trained 2-layer NN:

- 1st linear transformation: $\mathbf{z} = \mathbf{U}\mathbf{x} + \mathbf{c}, \quad \mathbf{U} \in \mathbb{R}^{2\times 2},\ \mathbf{c} \in \mathbb{R}^2$
- Element-wise nonlinear activation: $\mathbf{h} = \sigma(\mathbf{z})$;
  - Rectified Linear Unit (ReLU): $\sigma(t) = \max\{t, 0\}$
- 2nd linear transformation: $\hat{y} = \langle \mathbf{h}, \mathbf{w} \rangle + b$

$$\mathbf{U} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \mathbf{c} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \mathbf{w} = \begin{bmatrix} 2 \\ -4 \end{bmatrix}, b = -1$$

- $\mathbf{x}_1 = (0,0) \implies \mathbf{z}_1 = (0,-1), \mathbf{h}_1 = (0,0) \implies \hat{y}_1 = -1$ ✓ $\mathrm{y}_1 = -$
- $\mathbf{x}_2 = (1,0) \implies \mathbf{z}_2 = (1,\ \ 0), \mathbf{h}_2 = (1,0) \implies \hat{y}_2 = +1$ ✓ $\mathrm{y}_2 = +$
- $\mathbf{x}_3 = (0,1) \implies \mathbf{z}_3 = (1,\ \ 0), \mathbf{h}_3 = (1,0) \implies \hat{y}_3 = +1$ ✓ $\mathrm{y}_3 = +$
- $\mathbf{x}_4 = (1,1) \implies \mathbf{z}_4 = (2,\ \ 1), \mathbf{h}_4 = (2,1) \implies \hat{y}_4 = -1$ ✓ $\mathrm{y}_4 = -$

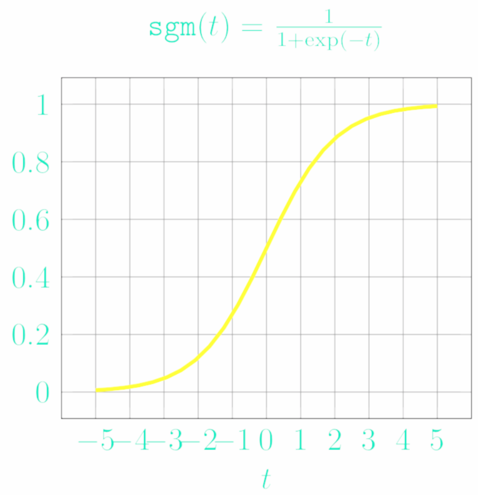
Multi-Class Classification

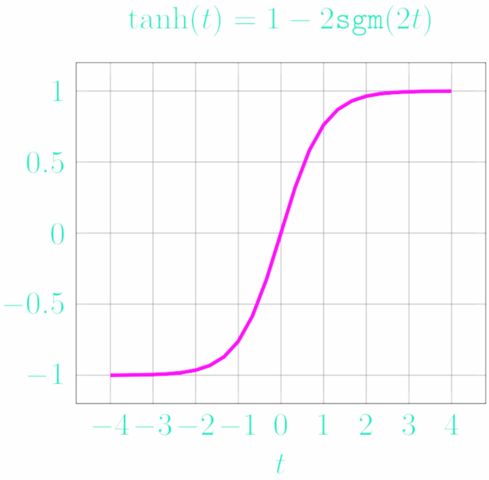


$$\underbrace{\mathbf{z} = \mathbf{U}\mathbf{x} + \mathbf{c}, \quad \mathbf{h} = \sigma(\mathbf{z}),}_{\text{learning feature } \mathbf{h}} \quad \underbrace{\hat{\mathbf{y}} = \mathbf{W}\mathbf{h} + \mathbf{b}, \quad \hat{\mathbf{p}} = \mathrm{softmax}(\hat{\mathbf{y}})}_{\text{learning linear classifier by logistic regression}}$$

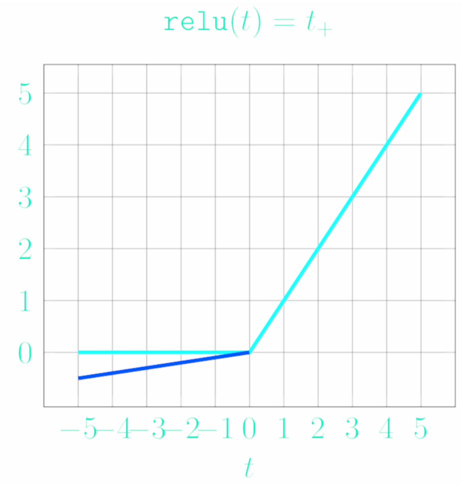
What if σ is linear? Say $\mathbf{h} = \sigma(\mathbf{z}) = \mathbf{V}\mathbf{z} + \mathbf{a}$

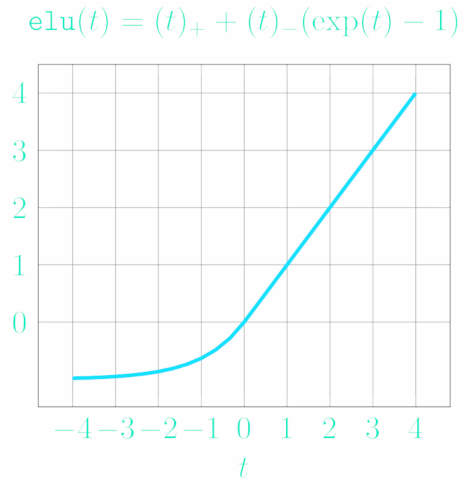
Activation Function



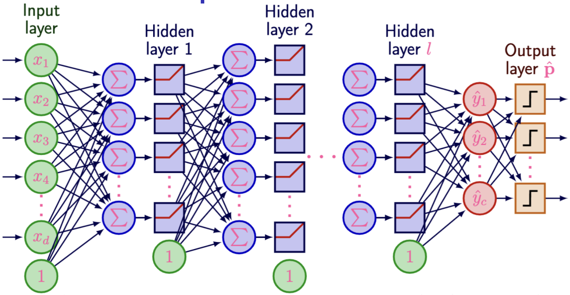


Activation Function — Cont'





MLP Training — Even Deeper



$$\hat{\mathbf{p}} = f(\mathbf{x}; \mathbf{w})$$

- Need a loss ℓ to measure difference between prediction $\hat{\mathbf{p}}$ and truth y
  - e.g., squared loss $\|\hat{\mathbf{p}} - \mathbf{y}\|_2^2$ (for regression, see Lecture 3) or log-loss $-\log \hat{p}_{\mathsf{y}}$ (for classification, see Lecture 4)
- Need a training set $\mathcal{D} = \{(\mathbf{x}_i, \mathsf{y}_i) : i = 1, \ldots, n\}$ to train weights $\mathbf{w}$

Stochastic Gradient Descent (SGD)

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} [\ell \circ f](\mathbf{x}_i, \mathsf{y}_i; \mathbf{w})$$

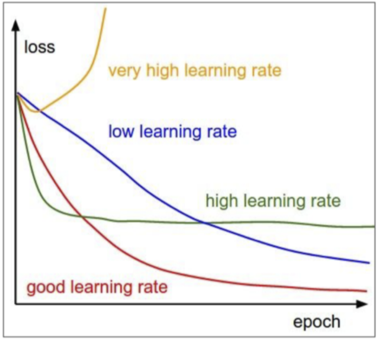
$$\boxed{\mathbf{w} \leftarrow \mathbf{w} - \eta \cdot \frac{1}{n} \sum_{i=1}^{n} \nabla [\ell \circ f](\mathbf{x}_i, \mathsf{y}_i; \mathbf{w})}$$

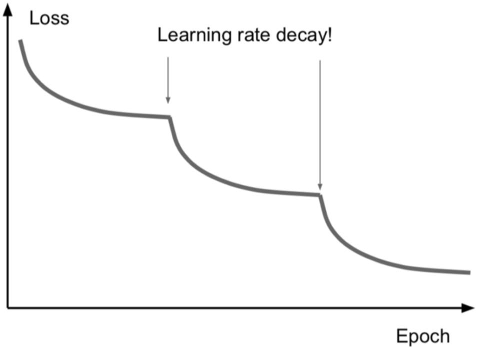
- $[\ell \circ f](\mathbf{x}_i, \mathsf{y}_i; \mathbf{w}) := \ell[f(\mathbf{x}_i; \mathbf{w}), \mathsf{y}_i]$
- Each iteration requires a full pass over the entire training set!
- A random, minibatch $B \subseteq \{1, \ldots, n\}$ suffices:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \cdot \frac{1}{|B|} \sum_{i \in B} \nabla [\ell \circ f](\mathbf{x}_i, \mathsf{y}_i; \mathbf{w})$$

- Trade-off between variance and computation

Learning Rate Decay





- Decrease every few epochs: $\eta_t = \begin{cases} \eta_0, & t \leq t_0; \\ \eta_0/10, & t_0 < t \leq t_1; \\ \eta_0/100, & t_1 < t. \end{cases}$
- Sublinear decay: $\eta_t = \eta_0/(1+ct)$ or $\eta_t = \eta_0/\sqrt{1+ct}$

How to Compute the Gradient for Gradient Descent?

- The forward pass of a 2-layer MLP (k is the NN width, c is the output dim):
  $\mathbf{x} =$ input ($\mathbf{x} \in \mathbb{R}^d$)
  $\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}_1$ ($\mathbf{W} \in \mathbb{R}^{k\times d}$ and $\mathbf{z}, \mathbf{b}_1 \in \mathbb{R}^k$)
  $\mathbf{h} = \text{ReLU}(\mathbf{z})$ ($\mathbf{h} \in \mathbb{R}^k$)
  $\boldsymbol{\theta} = \mathbf{U}\mathbf{h} + \mathbf{b}_2$ ($\mathbf{U} \in \mathbb{R}^{c\times k}$ and $\boldsymbol{\theta}, \mathbf{b}_2 \in \mathbb{R}^c$)
  $J = \frac{1}{2}\|\boldsymbol{\theta} - \mathbf{y}\|_2^2$ ($\mathbf{y} \in \mathbb{R}^c$)
- The parameters to be learned: $\mathbf{W}$, $\mathbf{b}_1$, $\mathbf{U}$, $\mathbf{b}_2$
- Network's gradient: $\frac{\partial J}{\partial \mathbf{W}}, \frac{\partial J}{\partial \mathbf{b}_1}, \frac{\partial J}{\partial \mathbf{U}}, \frac{\partial J}{\partial \mathbf{b}_2}$
- Recall that ReLU$(x) = \max(x, 0)$. So

$$\text{ReLU}'(x) = \begin{cases} 1, & \text{if } x > 0; \\ 0, & \text{otherwise.} \end{cases}$$

Matrix Calculus Basics

- Matrix calculus is complicated and cannot be taught clearly within 2-3 lectures (out of the scope of this course)!
- **Definition:** Let $y(\mathbf{X}) \in \mathbb{R}$ and $\mathbf{X} = [X_{ij}]_{i=1,j=1}^{m,n} \in \mathbb{R}^{m \times n}$. Then

$$\frac{\partial y}{\partial \mathbf{X}} = \begin{bmatrix} \frac{\partial y}{\partial X_{11}} & \frac{\partial y}{\partial X_{12}} & \cdots & \frac{\partial y}{\partial X_{1n}} \\ \frac{\partial y}{\partial X_{21}} & \frac{\partial y}{\partial X_{22}} & \cdots & \frac{\partial y}{\partial X_{2n}} \\ \vdots & \vdots & & \vdots \\ \frac{\partial y}{\partial X_{m1}} & \frac{\partial y}{\partial X_{m2}} & \cdots & \frac{\partial y}{\partial X_{mn}} \end{bmatrix} \in \mathbb{R}^{m \times n}.$$

- **Best way to calculate matrix calculus:** Analogous to your calculation of scalar calculus, you want to "guess" a solution with a matched dimensionality. Three steps:
  1. "Guess" a solution analogous to scalar calculus;
  2. Check if the dimension is right $\frac{\partial y}{\partial \mathbf{X}} \in \mathbb{R}^{m \times n}$ for $\mathbf{X} \in \mathbb{R}^{m \times n}$;
  3. Return to Step 1 or Finish.

How to Compute the Gradient? Chain Rule!

- The forward pass of a 2-layer MLP (k is the NN width, c is the output dim):
  $\mathbf{x} = \text{input } (\mathbf{x} \in \mathbb{R}^{d\times 1})$
  $\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}_1 \; (\mathbf{W} \in \mathbb{R}^{k\times d} \text{ and } \mathbf{z}, \mathbf{b}_1 \in \mathbb{R}^{k\times 1})$
  $\mathbf{h} = \text{ReLU}(\mathbf{z}) \; (\mathbf{h} \in \mathbb{R}^{k\times 1})$
  $\boldsymbol{\theta} = \mathbf{U}\mathbf{h} + \mathbf{b}_2 \; (\mathbf{U} \in \mathbb{R}^{c\times k} \text{ and } \boldsymbol{\theta}, \mathbf{b}_2 \in \mathbb{R}^{c\times 1})$
  $J = \frac{1}{2}\|\boldsymbol{\theta} - \mathbf{y}\|_2^2 \; (\mathbf{y} \in \mathbb{R}^{c\times 1})$
- The backward pass of the model ($\odot$ is the Hadamard product):

$$\frac{\partial J}{\partial \boldsymbol{\theta}} = \boldsymbol{\theta} - \mathbf{y}$$

$$\frac{\partial J}{\partial \mathbf{U}} = \frac{\partial J}{\partial \boldsymbol{\theta}} \circ \frac{\partial \boldsymbol{\theta}}{\partial \mathbf{U}} = (\boldsymbol{\theta} - \mathbf{y})\mathbf{h}^T$$

$$\frac{\partial J}{\partial \mathbf{b}_2} = \frac{\partial J}{\partial \boldsymbol{\theta}} \circ \frac{\partial \boldsymbol{\theta}}{\partial \mathbf{b}_2} = \boldsymbol{\theta} - \mathbf{y}$$

$$\frac{\partial J}{\partial \mathbf{h}} = \frac{\partial J}{\partial \boldsymbol{\theta}} \circ \frac{\partial \boldsymbol{\theta}}{\partial \mathbf{h}} = \mathbf{U}^T(\boldsymbol{\theta} - \mathbf{y})$$

$$\frac{\partial J}{\partial \mathbf{z}} = \frac{\partial J}{\partial \mathbf{h}} \circ \frac{\partial \mathbf{h}}{\partial \mathbf{z}} = \mathbf{U}^T(\boldsymbol{\theta} - \mathbf{y}) \odot \text{ReLU}'(\mathbf{z})$$

$$\frac{\partial J}{\partial \mathbf{W}} = \frac{\partial J}{\partial \mathbf{z}} \circ \frac{\partial \mathbf{z}}{\partial \mathbf{W}} = (\mathbf{U}^T(\boldsymbol{\theta} - \mathbf{y}) \odot \text{ReLU}'(\mathbf{z}))\mathbf{x}^T$$

$$\frac{\partial J}{\partial \mathbf{b}_1} = \frac{\partial J}{\partial \mathbf{z}} \circ \frac{\partial \mathbf{z}}{\partial \mathbf{b}_1} = \mathbf{U}^T(\boldsymbol{\theta} - \mathbf{y}) \odot \text{ReLU}'(\mathbf{z})$$

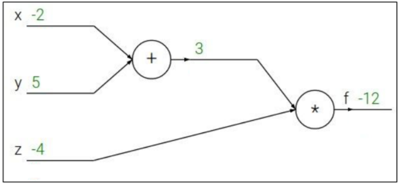
How to Compute the Gradient? Chain Rule!

Existing frameworks will memorize the computational graph in the forward process and calculate the back-propagation automatically for you!

A Simple Example

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



A Simple Example (Forward)

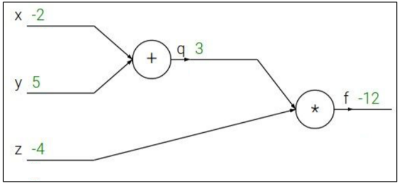
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$\boxed{q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1}$$

$$\boxed{f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



A Simple Example (Backward)

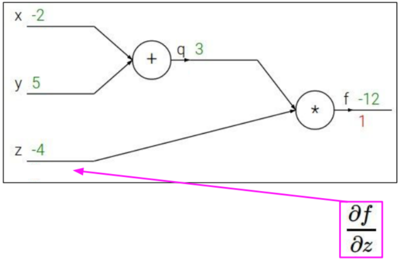
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



A Simple Example (Backward)

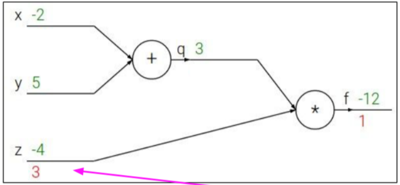
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$\boxed{q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1}$$

$$\boxed{f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\boxed{\frac{\partial f}{\partial z}}$$

Fei-Fei Li & Andrej Karpathy & Justin Johnson. Stanford University

A Simple Example (Backward)

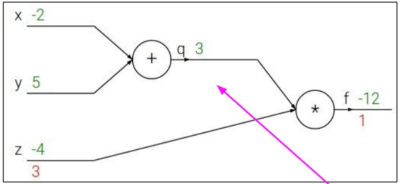
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial q}$$

A Simple Example (Backward)

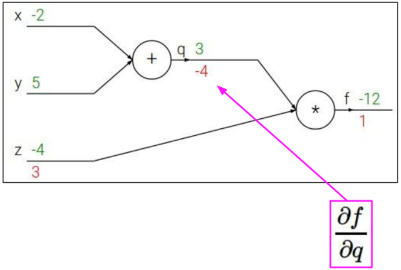
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



A Simple Example (Backward)

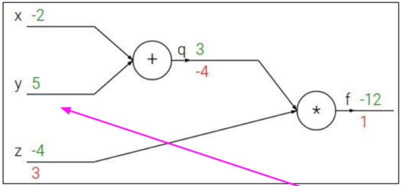
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial y}$$

A Simple Example (Backward)

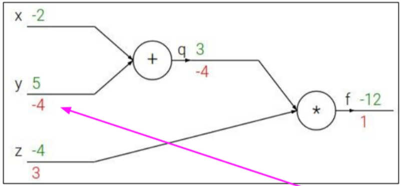
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$\frac{\partial f}{\partial y}$

Chain rule:

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial y}$$

A Simple Example (Backward)

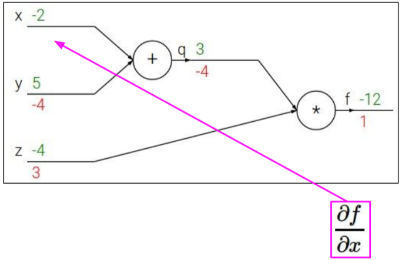
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\dfrac{\partial f}{\partial x}, \dfrac{\partial f}{\partial y}, \dfrac{\partial f}{\partial z}$



A Simple Example (Backward)

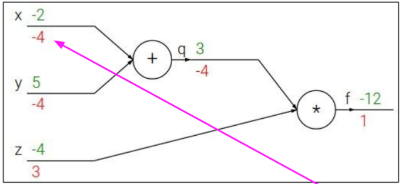
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial x}$$

Chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$

Universal Approximation Theorem

Theorem: Universal Approximation Theorem by 2-Layer NNs

For any continuous function $f : \mathbb{R}^d \to \mathbb{R}^c$ and any $\epsilon > 0$, there exists $k \in \mathbb{N}$, $\mathbf{W} \in \mathbb{R}^{k \times d}$, $\mathbf{b} \in \mathbb{R}^k$, $\mathbf{U} \in \mathbb{R}^{c \times k}$ such that

$$\sup_{\mathbf{x}} \|f(\mathbf{x}) - g(\mathbf{x})\|_2 < \epsilon,$$

where $g(\mathbf{x}) = \mathbf{U}(\sigma(\mathbf{W}\mathbf{x} + \mathbf{b}))$ and σ is the element-wise ReLU operation.

As long as 2-layer MLP is wide enough (a large k), it can approximate any continuous function arbitrarily closely.

J.-P. Kahane. "Sur le theoreme de superposition de Kolmogorov". Journal of Approximation Theory, vol. 13, no. 3 (1975), pp. 229–234, A. N. Kolmogorov. "On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition". Soviet Mathematics Doklady, vol. 114, no. 5 (1957), pp. 953–956, V. I. Arnol'd. "On Functions of Three Variables". Soviet Mathematics Doklady, vol. 114, no. 4 (1957), pp. 679–681.

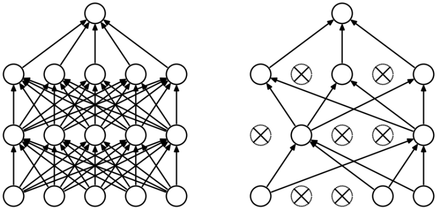
Then Why Deep Learning?

- There exists a function such that 2-layer MLP needs to be $k = \exp(1/\epsilon)$ wide to approximate the function, but 3-layer MLP only needs to be $k = \mathrm{poly}(1/\epsilon)$ wide.
- Deep NNs are more parameter-efficient.

"The Power of Depth for Feedforward Neural Networks" by Eldan and Shamir. 2016
"Benefit of Depth in Neural Networks" by Telgarsky. 2016

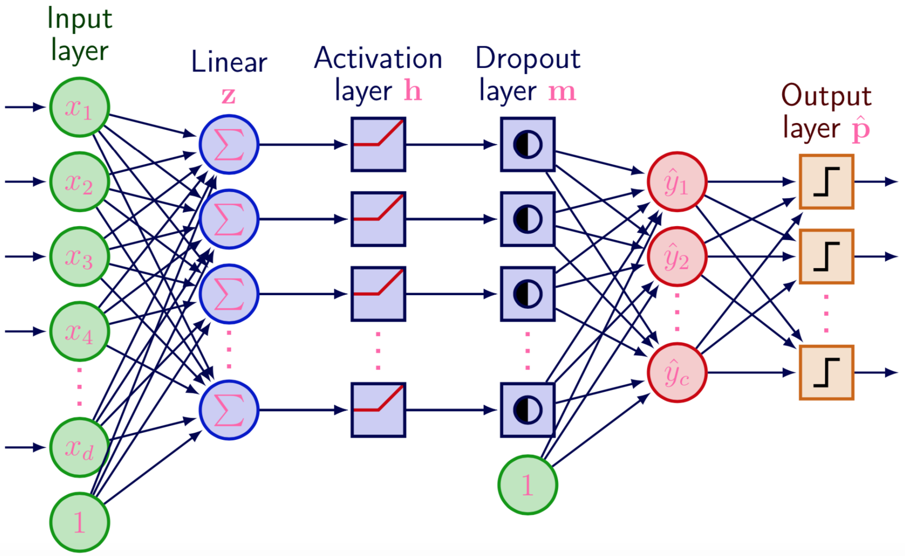
Dropout

- Training:
  - For each training minibatch, keep each hidden unit with probability q
  - A different and random network for each training minibatch
  - Hidden units are less likely to collude to overfit training data
  - Inverted: after the removal, multiply each $\mathbf{h}_k$ with a scaling factor $1/q$ to keep the same expectation
- Testing: Use the full network



(a) Standard Neural Net (b) After applying dropout.

N. Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". Journal of Machine Learning Research, vol. 15, no. 56 (2014), pp. 1929–1958.



Input layer
Linear $\mathbf{z}$
Activation layer $\mathbf{h}$
Dropout layer $\mathbf{m}$
Output layer $\hat{\mathbf{p}}$
x_1
x_2
x_3
x_4
x_d
1
1
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_c$

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

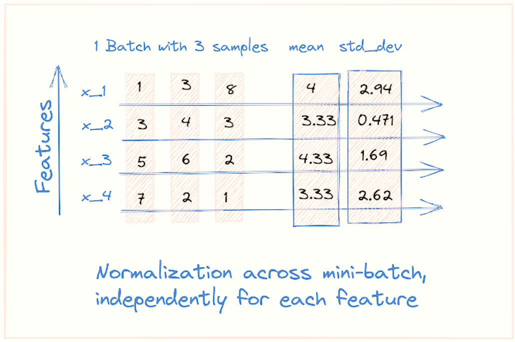
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

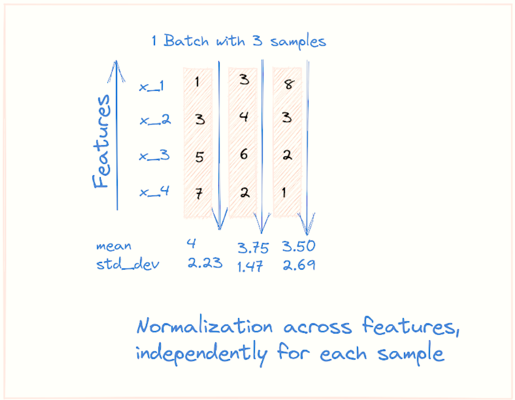
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

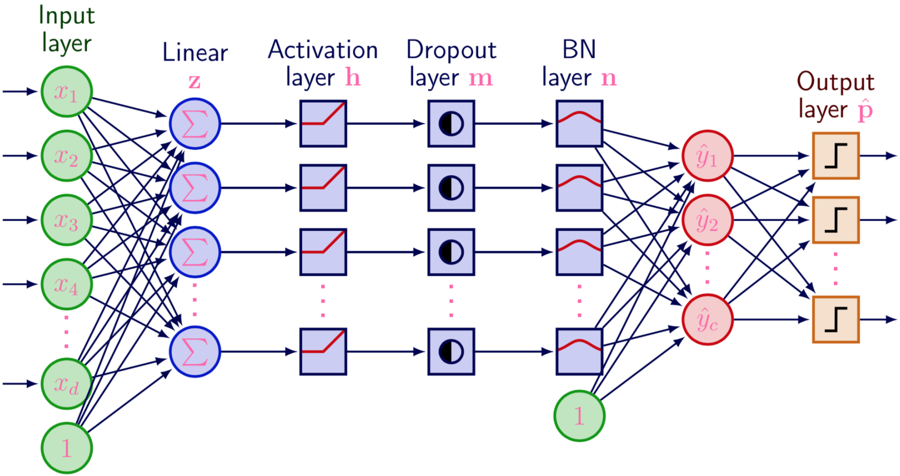
S. Ioffe and C. Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: Proceedings of the 32nd International Conference on Machine Learning ICML. vol. 37. 2015, pp. 448–456.

Batch Normalization vs. Layer Normalization





A Complete MLP Architecture




Questions
?
?
Answers
?