

Machine Learning Using Tensorflow

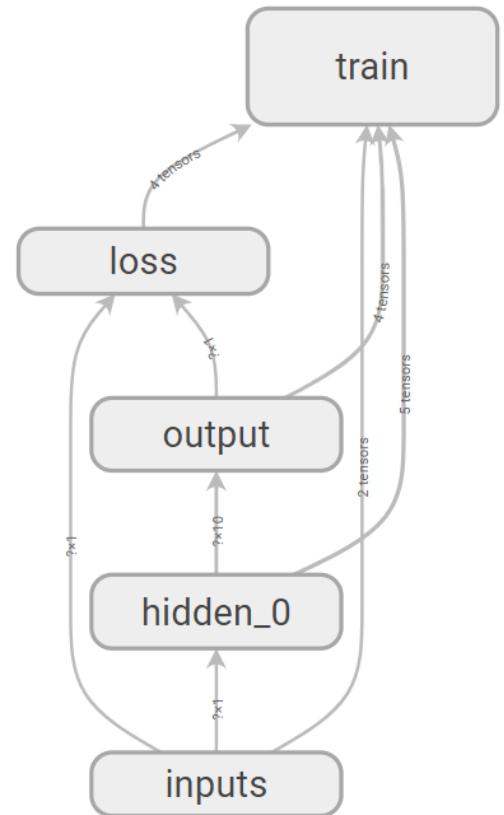
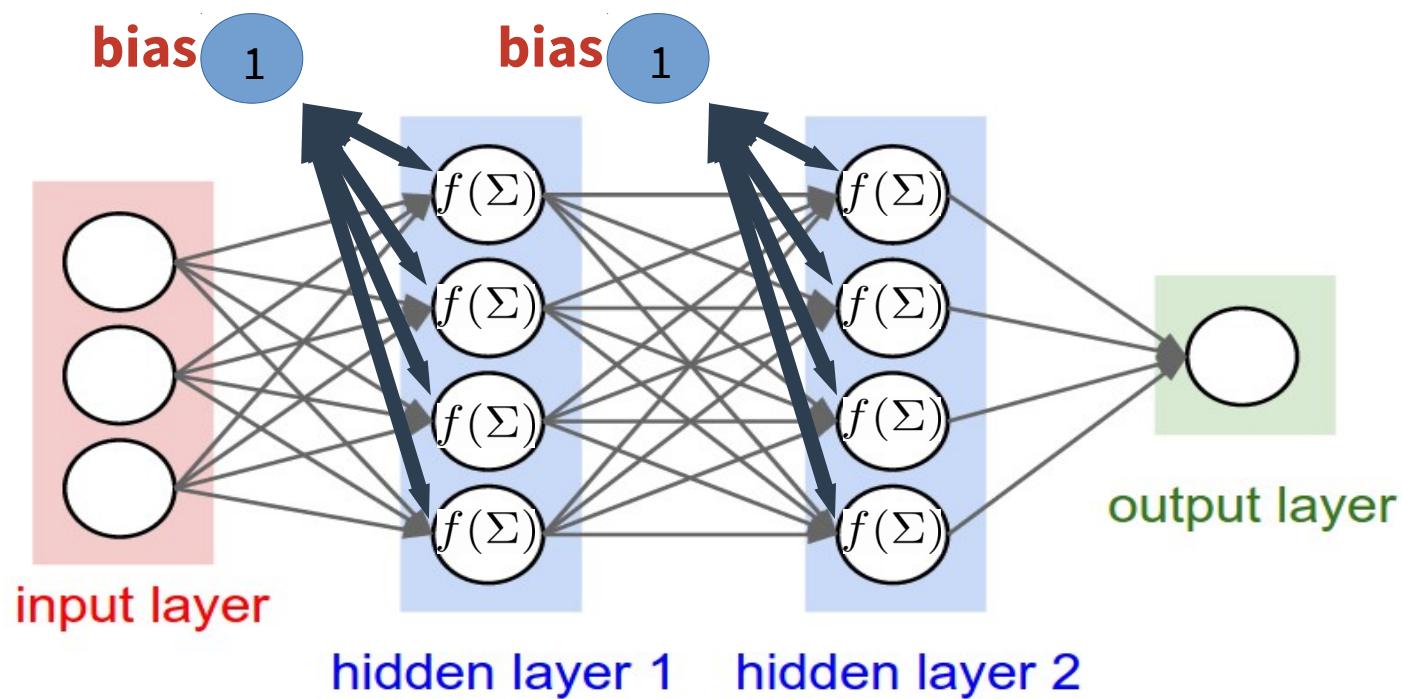
Week4:

Basis of Deep Learning : rethink deep learning

Shu-Ting Pi, PhD
UC Davis



Really understand everything?



Questions

Overfitting

- In general, we can fit “any” data if parameters are many

Dimensionality Curse

- How many data are consider “enough”?

Gradient vanishing and explosion

- How deep can a neural network be?
- Why Relu is better?

Information Theory

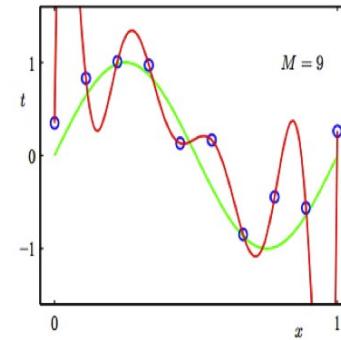
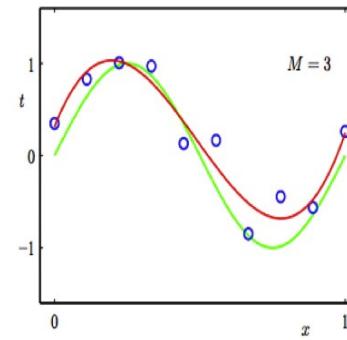
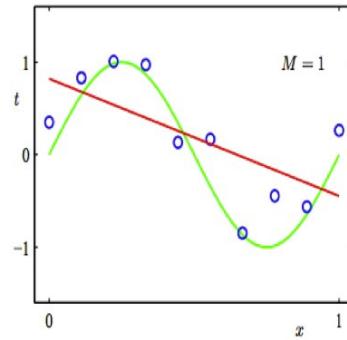
- Ask again: what is deep learning?

Overfitting

Overfitting

Under- and Over-fitting examples

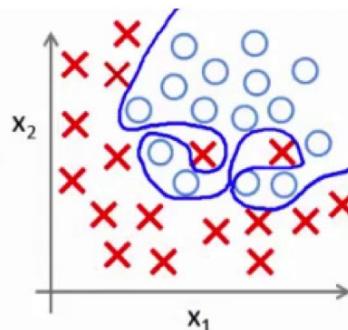
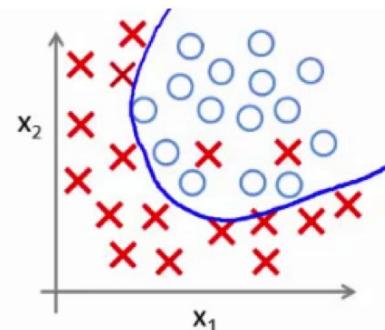
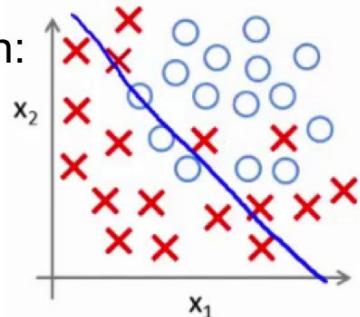
Regression:



predictor too inflexible:
cannot capture pattern

predictor too flexible:
fits noise in the data

Classification:



Copyright © 2014 Victor Lavrenko

Regularization

Regularization

When number of parameters in model is high, overfitting is very probable

Solution: add a *regularization term* to the loss function:

$$\mathcal{L} = \frac{1}{N} \sum_i L(x_i, y_i) + \mathcal{L}_{\text{reg}} \rightarrow \min$$

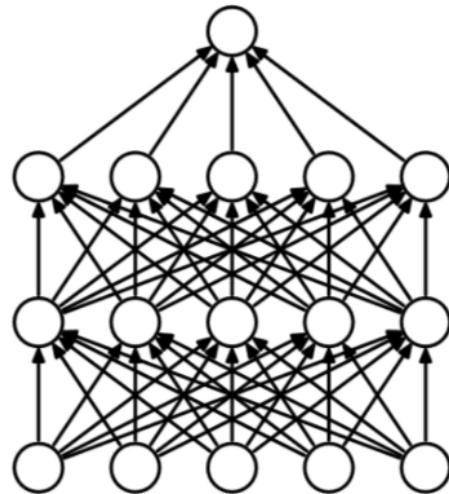
- L_2 regularization : $\mathcal{L}_{\text{reg}} = \alpha \sum_j |w_j|^2$ Hint: Lagrangian constraints
- L_1 regularization: $\mathcal{L}_{\text{reg}} = \beta \sum_j |w_j|$
- $L_1 + L_2$ regularization: $\mathcal{L}_{\text{reg}} = \alpha \sum_j |w_j|^2 + \beta \sum_j |w_j|$

L1: make more W zero, non-derivative

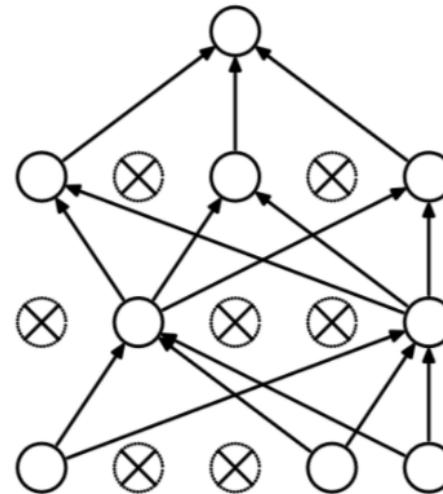
L2: won't make more W zero, derivative

40 / 99

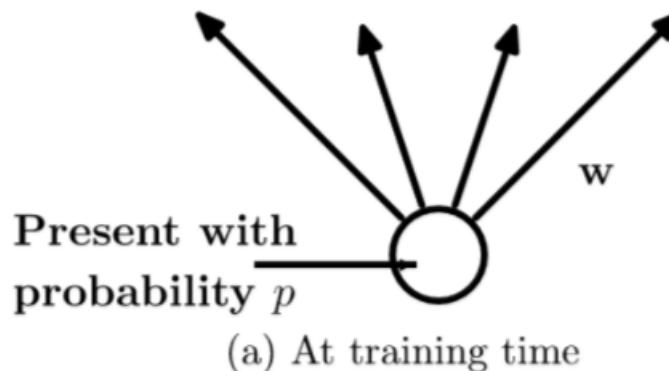
Dropout



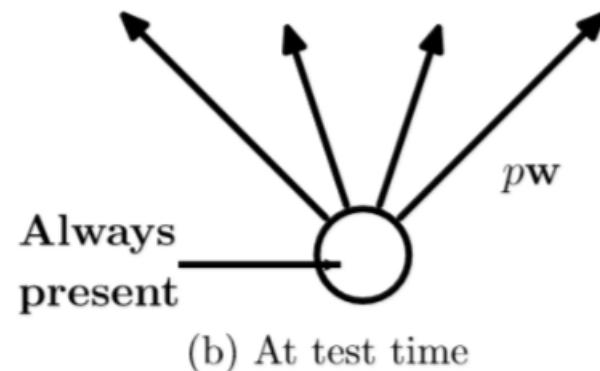
(a) Standard Neural Net



(b) After applying dropout.

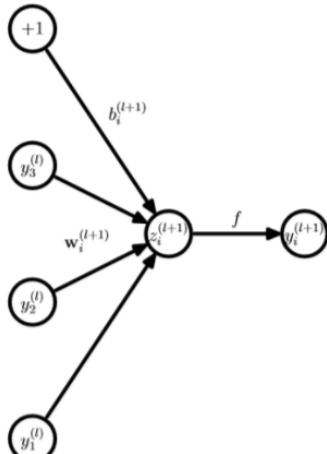


(a) At training time



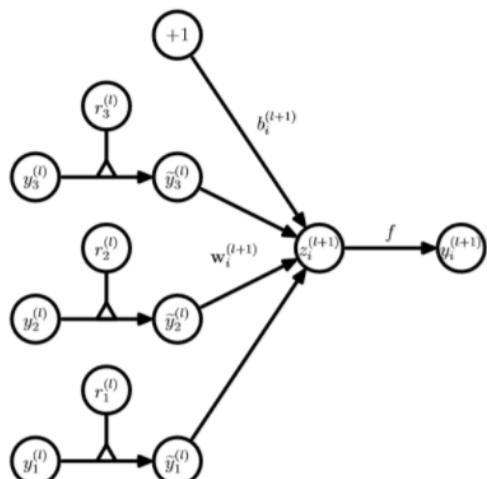
(b) At test time

Dropout



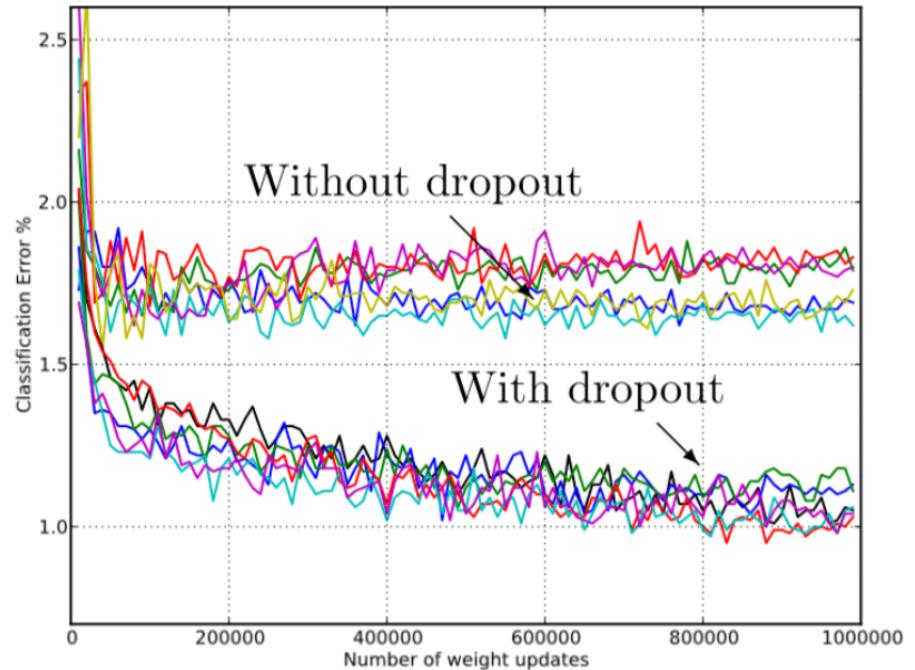
(a) Standard network

$$\begin{aligned} z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \mathbf{y}^l + b_i^{(l+1)}, \\ y_i^{(l+1)} &= f(z_i^{(l+1)}), \end{aligned}$$



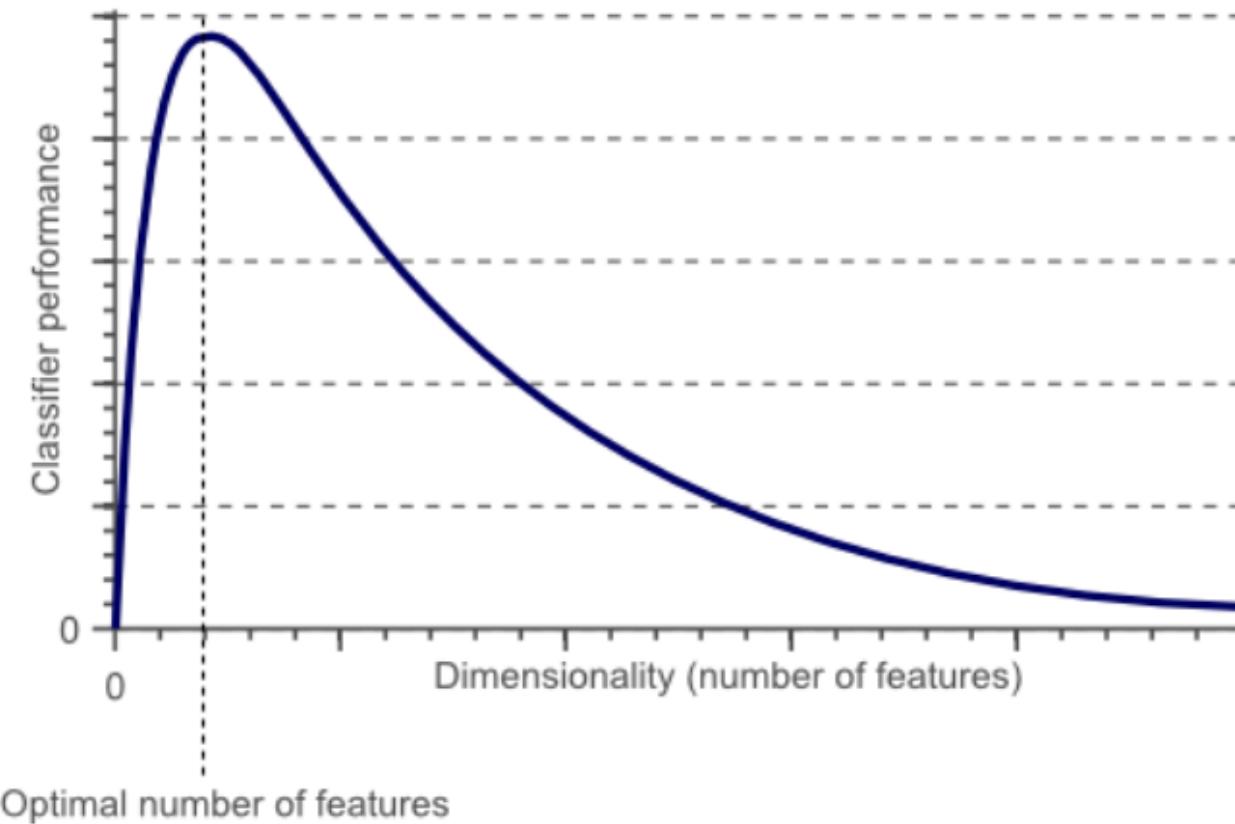
(b) Dropout network

$$\begin{aligned} r_j^{(l)} &\sim \text{Bernoulli}(p), \\ \tilde{\mathbf{y}}^{(l)} &= \mathbf{r}^{(l)} * \mathbf{y}^{(l)}, \\ z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \tilde{\mathbf{y}}^l + b_i^{(l+1)}, \\ y_i^{(l+1)} &= f(z_i^{(l+1)}). \end{aligned}$$



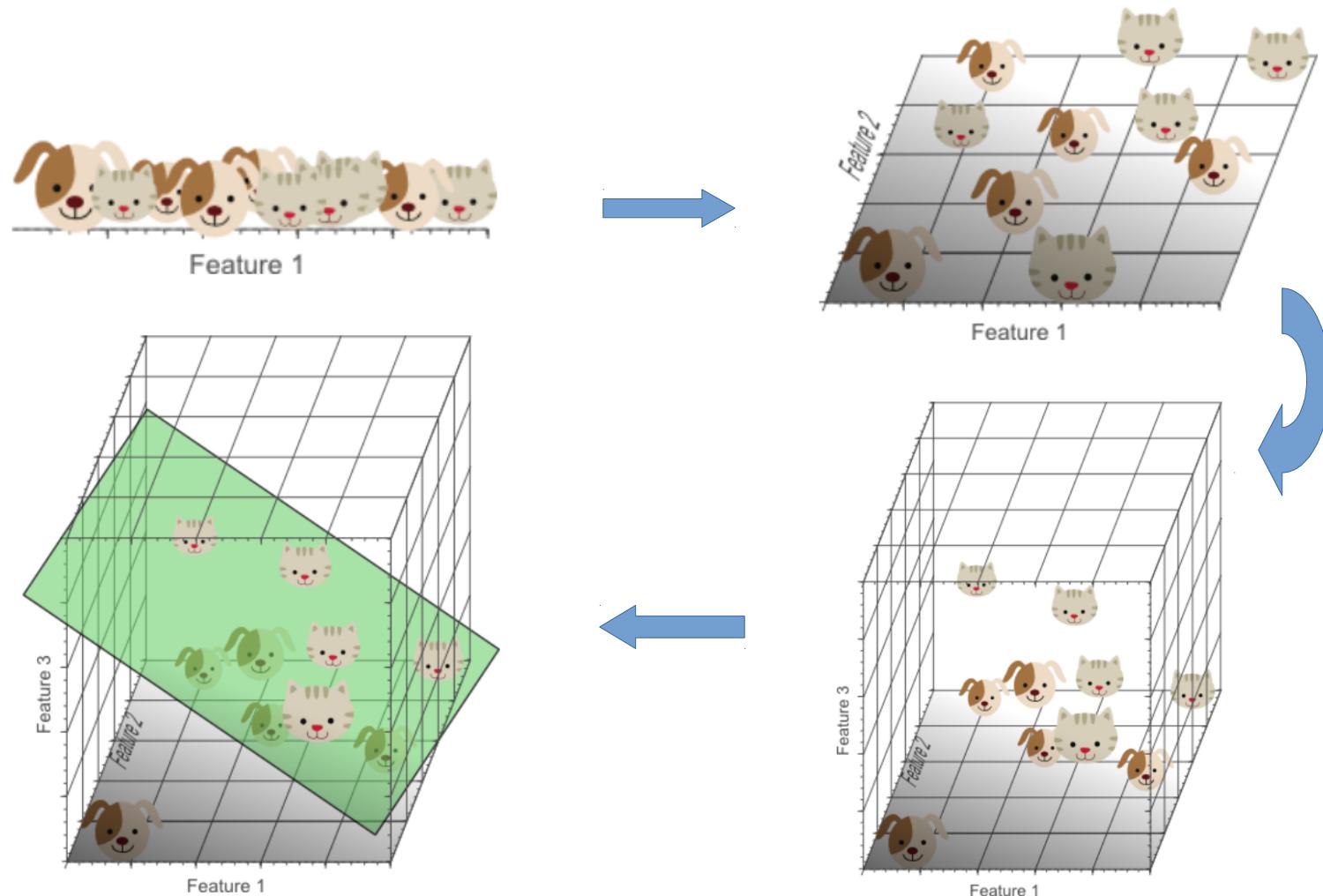
Dimensionality Curse

What is dimensionality curse?

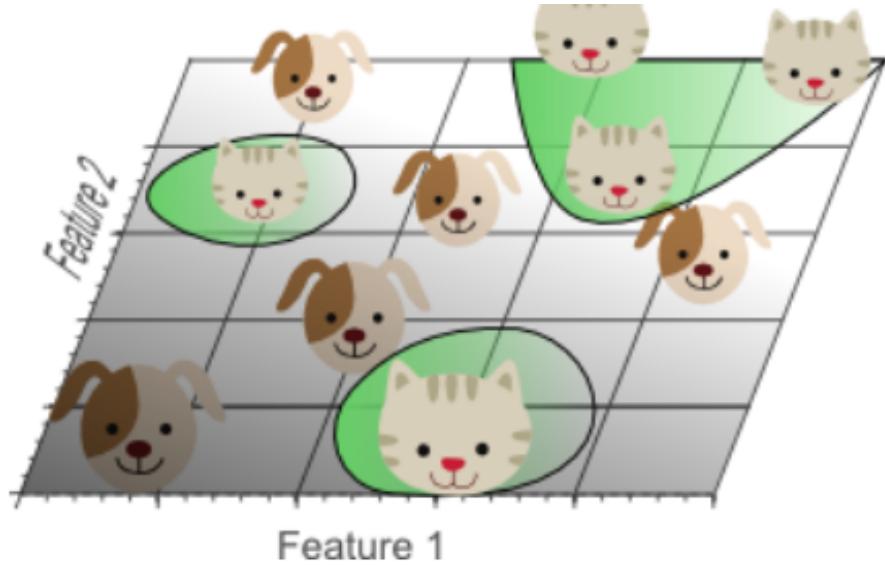


Performance decrease beyond a critical number of features

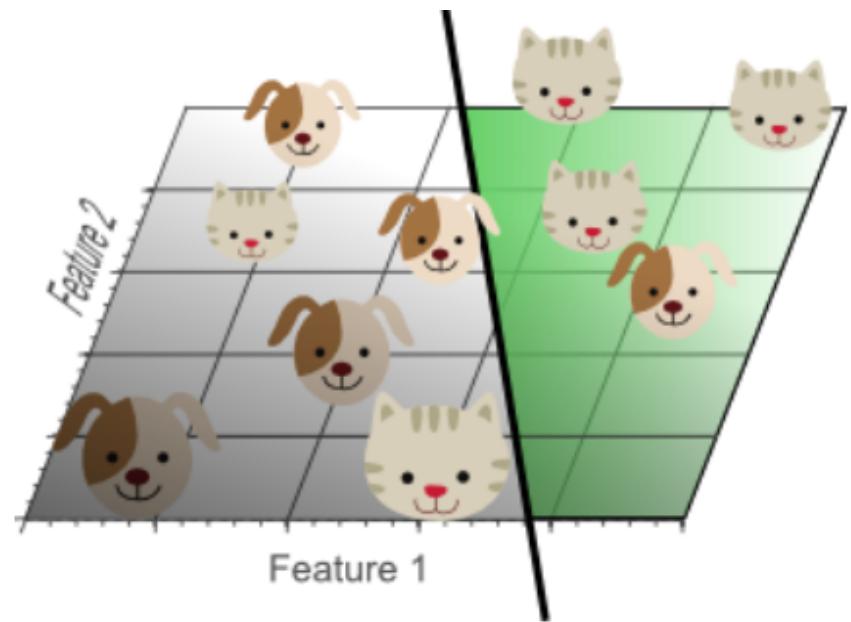
Increase dimension



Which one is better?

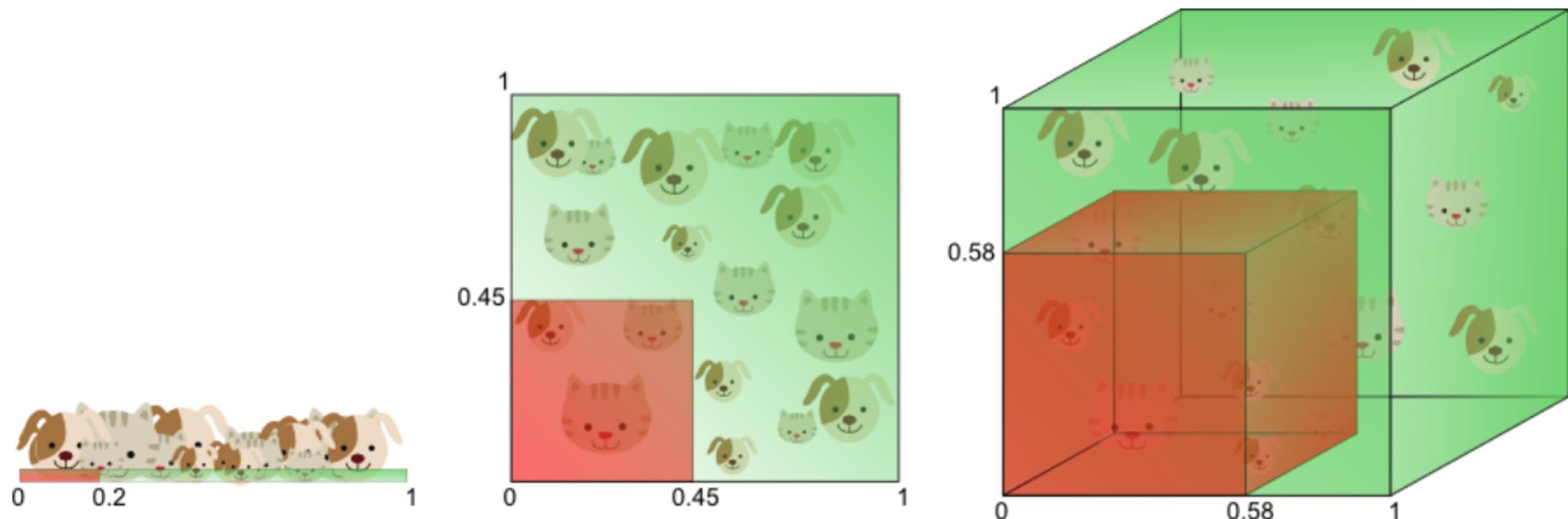


Project 3D linear separation to 2D



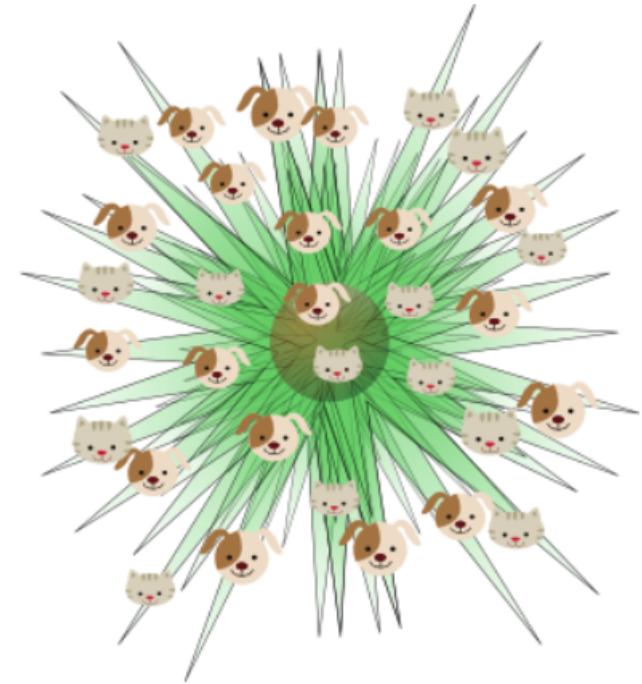
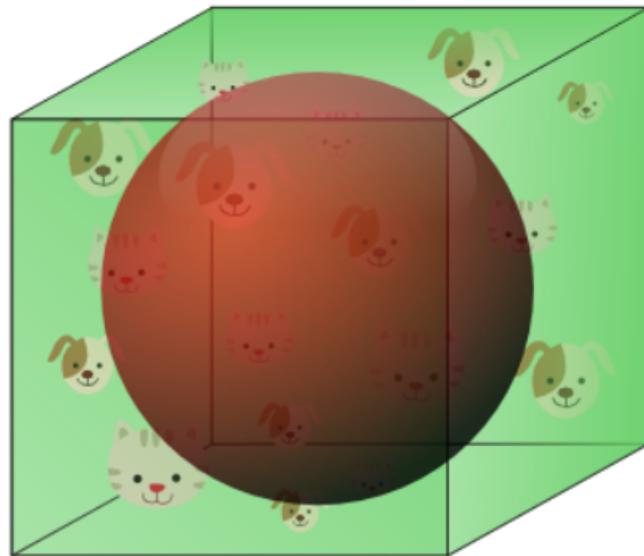
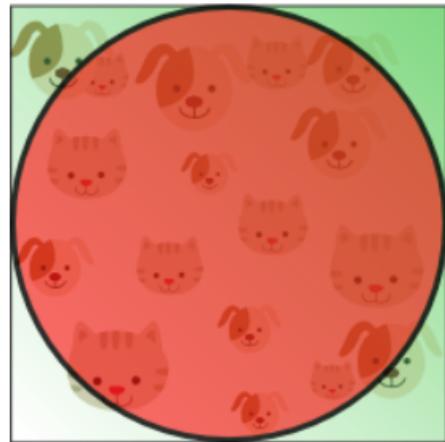
Simple 2D linear separation

The cubic approach



**Want 20% coverage on the value domain:
1D = 0.2 , 2D= 0.45, 3D= 0.58**

The corner approach

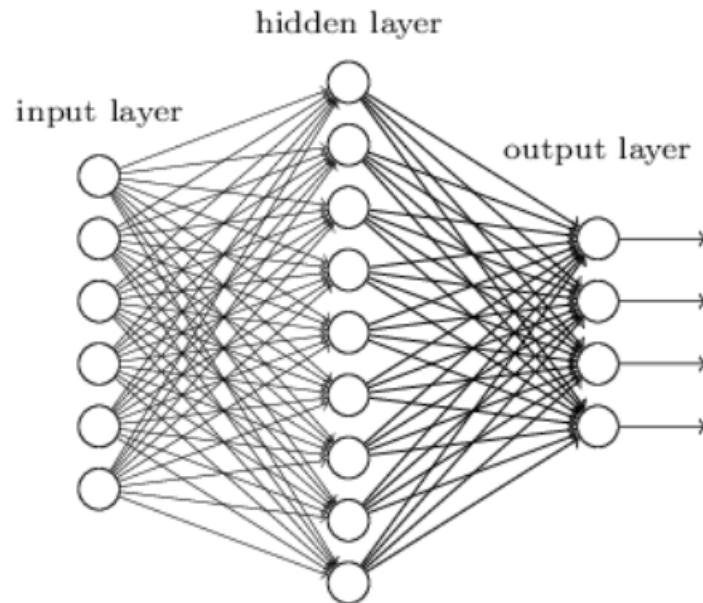


Data around the corner are “sparse” !

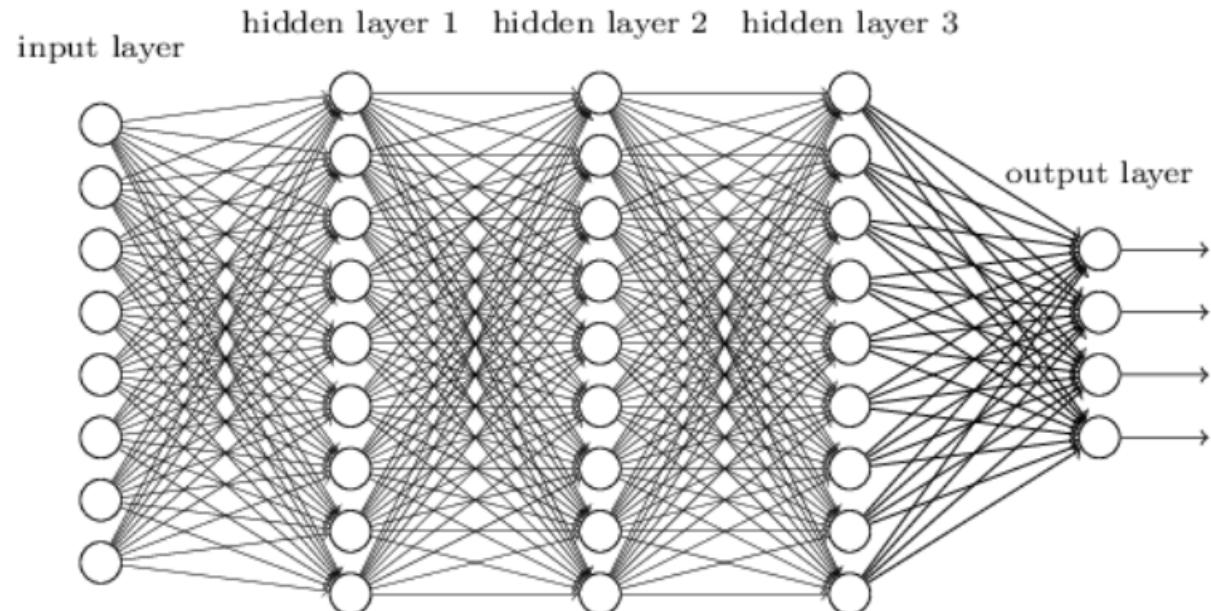
$$V(d) = \frac{\pi^{d/2}}{\Gamma(d/2 + 1)} \left(\frac{1}{2}\right)^d \rightarrow \boxed{\frac{V_s}{V_c} = 0.02} \rightarrow \lim_{d \rightarrow \infty} \frac{d_{max} - d_{min}}{d_{min}} = 0$$

Gradient vanishing & explosion

the deeper the better?



Few-layer NN

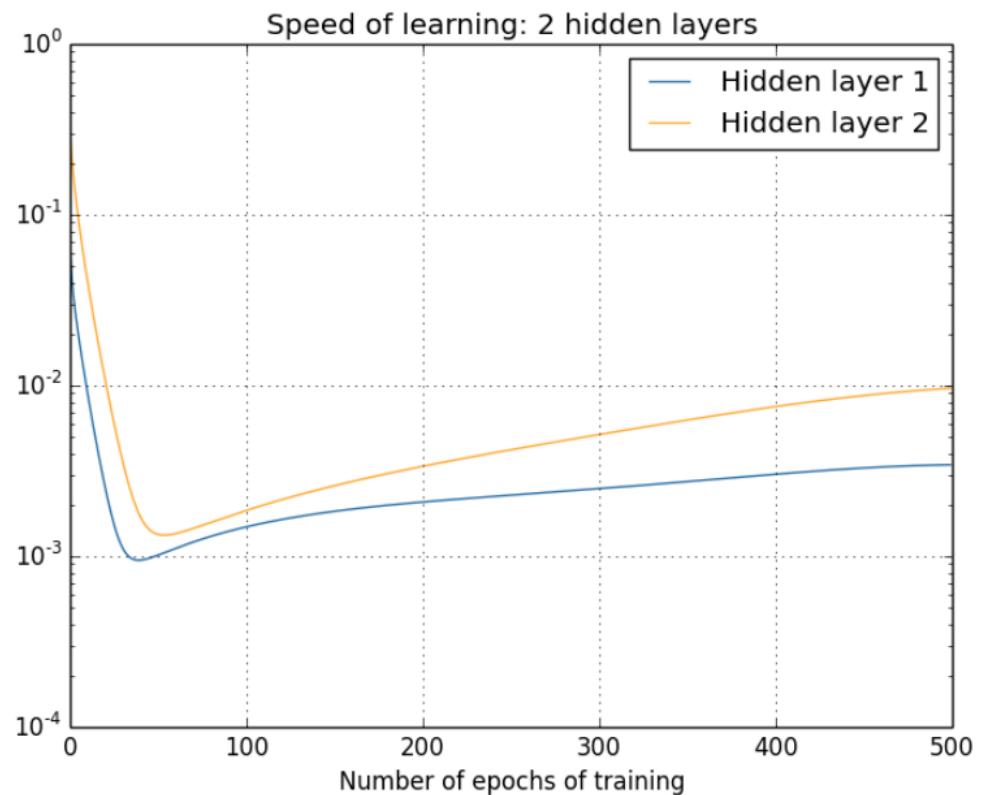
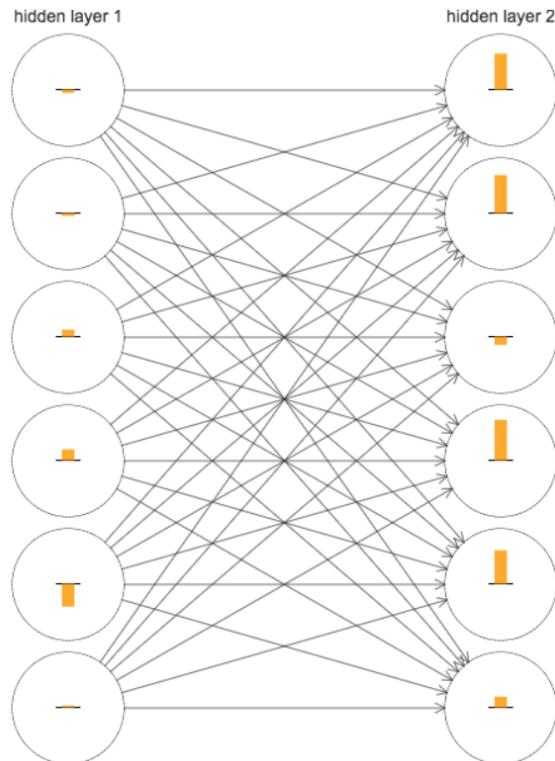


Many-layer NN

More parameters in many-layer NN, so more features can be extract?
Okay, maybe you are right, but how to train it?

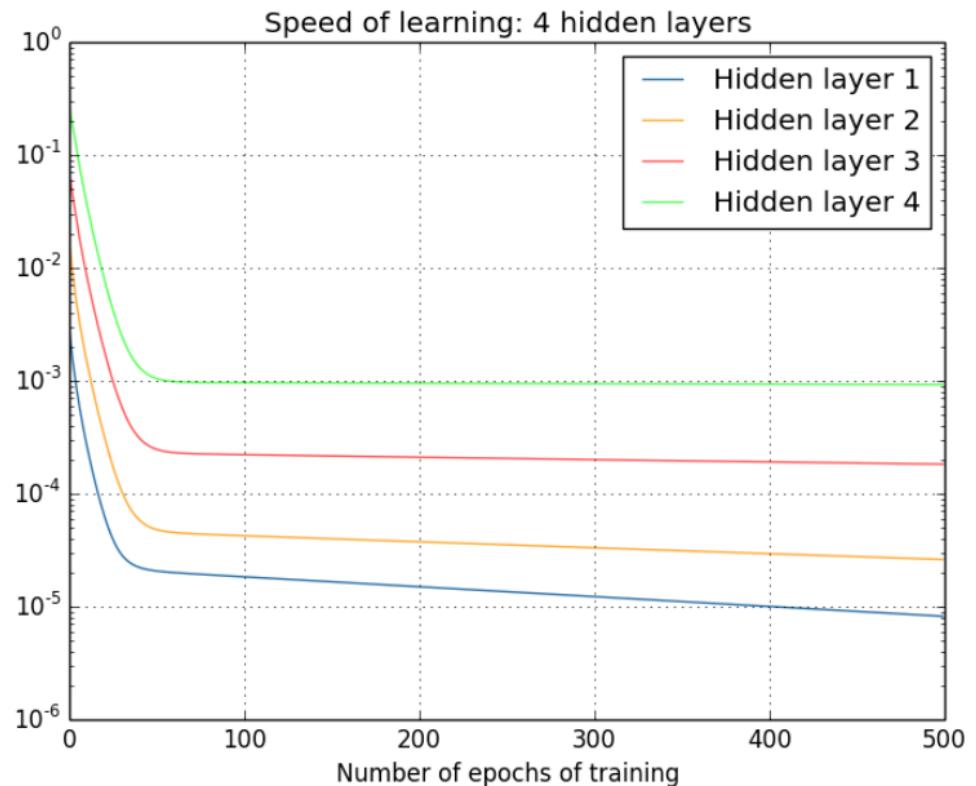
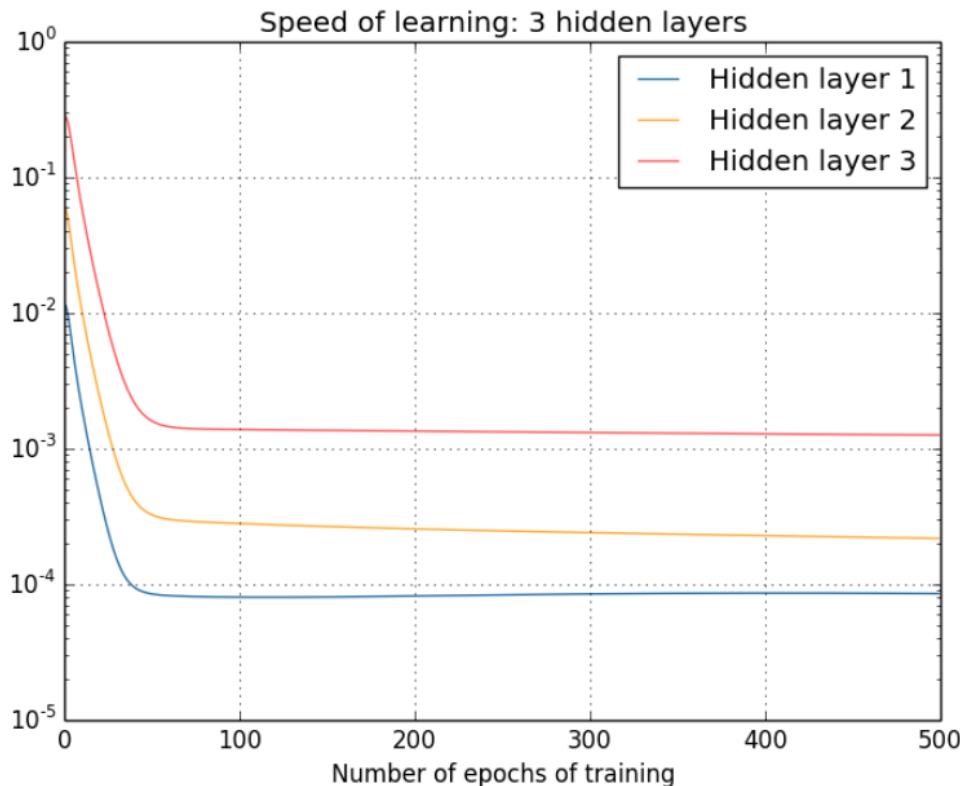
What is Gradient Vanishing?

MNIST, sigmoid activation function



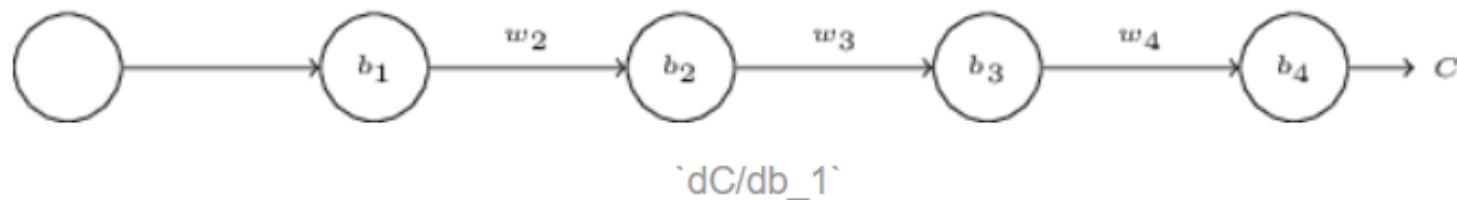
Learning speed: $\frac{\partial S}{\partial b_i}$

Problem becomes worse in multilayer



Why gradient vanishing?

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$



$$\frac{\partial C}{\partial b_1} \approx \frac{\Delta C}{\Delta b_1}. \quad (114)$$

$$\Delta C \approx \sigma'(z_1)w_2\sigma'(z_2) \dots \sigma'(z_4) \frac{\partial C}{\partial a_4} \Delta b_1. \quad (120)$$

$$\Delta a_1 \approx \frac{\partial \sigma(w_1 a_0 + b_1)}{\partial b_1} \Delta b_1 \quad (115)$$

$$= \sigma'(z_1) \Delta b_1. \quad (116)$$

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1)w_2\sigma'(z_2) \dots \sigma'(z_4) \frac{\partial C}{\partial a_4}. \quad (121)$$

$$\Delta z_2 \approx \frac{\partial z_2}{\partial a_1} \Delta a_1 \quad (117)$$

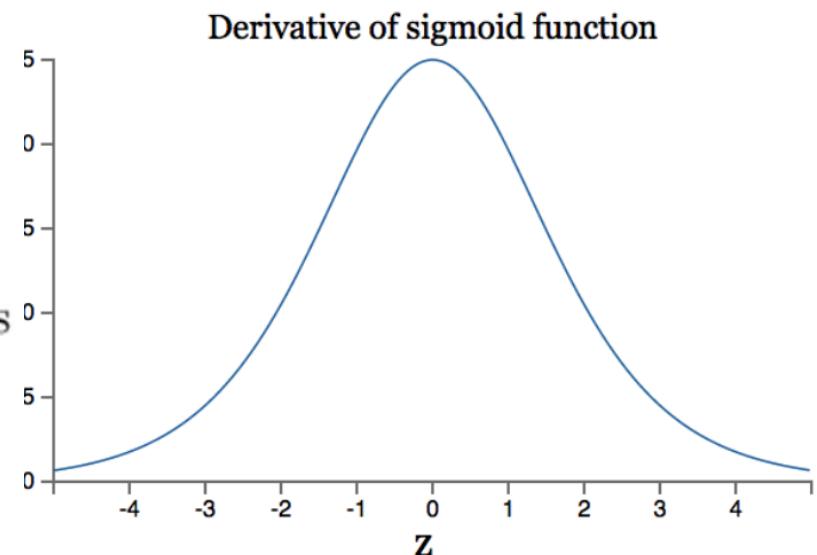
$$= w_2 \Delta a_1. \quad (118)$$

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) w_2 \sigma'(z_2) w_3 \sigma'(z_3) w_4 \sigma'(z_4) \frac{\partial C}{\partial a_4}. \quad (122)$$

Gradient Vanishing

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) w_2 \sigma'(z_2) w_3 \sigma'(z_3) w_4 \sigma'(z_4) \frac{\partial C}{\partial a_4}.$$

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \overbrace{w_2 \sigma'(z_2)}^{< \frac{1}{4}} \overbrace{w_3 \sigma'(z_3)}^{< \frac{1}{4}} \underbrace{w_4 \sigma'(z_4)}_{\text{common terms}} \frac{\partial C}{\partial a_4}$$
$$\frac{\partial C}{\partial b_3} = \sigma'(z_3) w_4 \sigma'(z_4) \frac{\partial C}{\partial a_4}$$



If all W's < 1, all layers contribute a value < 1/4, exponential decay. Gradient Vanishing !

If all W's are larger, say, > 1/4, exponential increase. Gradient explosion !

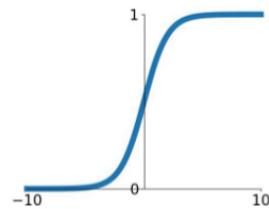
Only 25%, at most, errors can propagate backward !

Deep Learning is Gradient Unstable

Activation Functions

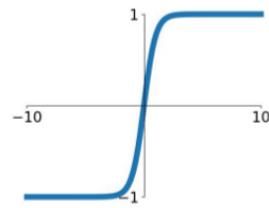
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



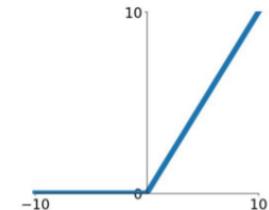
tanh

$$\tanh(x)$$



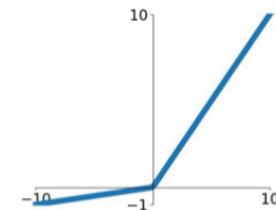
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

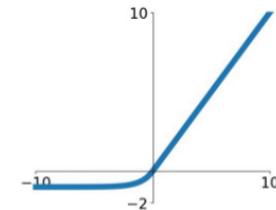


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



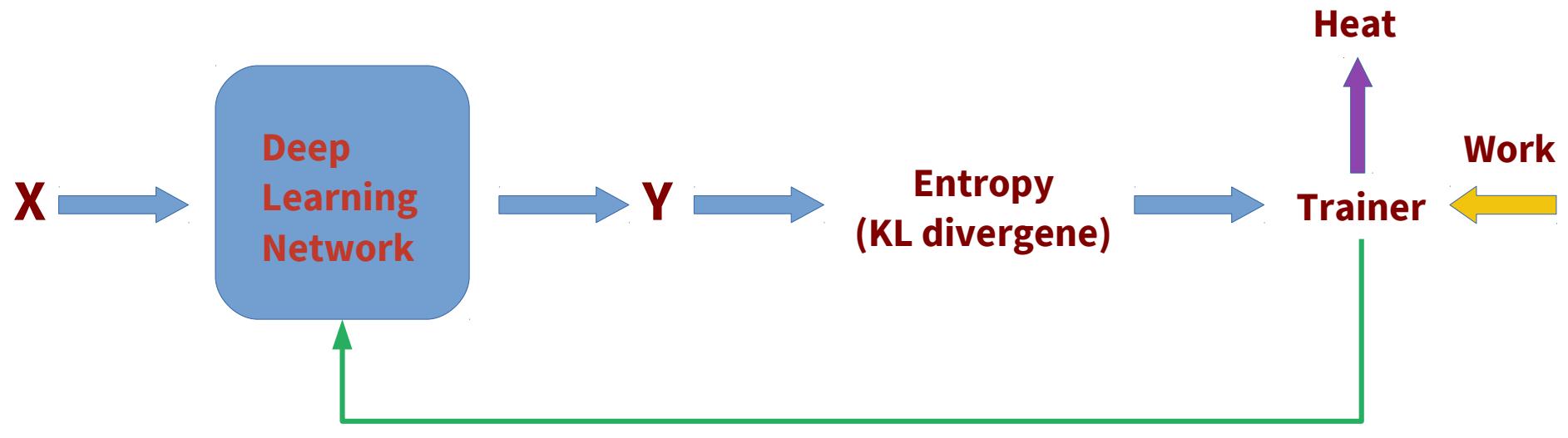
Remember: Deep learning neural network is always “gradient unstable”!

Information Theory

What a neural network really does?

Information

- Any object or event that can reduce “uncertainty”



Deep Learning Network is essential a “Carnot Refrigerator”

Opening the black box of Deep Neural Networks via Information

Ravid Schwartz-Ziv

*Edmond and Lilly Safra Center for Brain Sciences
The Hebrew University of Jerusalem
Jerusalem, 91904, Israel*

RAVID.ZIV@MAIL.HUJI.AC.IL

Naftali Tishby*

*School of Engineering and Computer Science
and Edmond and Lilly Safra Center for Brain Sciences
The Hebrew University of Jerusalem
Jerusalem, 91904, Israel*

TISHBY@CS.HUJI.AC.IL



Naftali Tishby

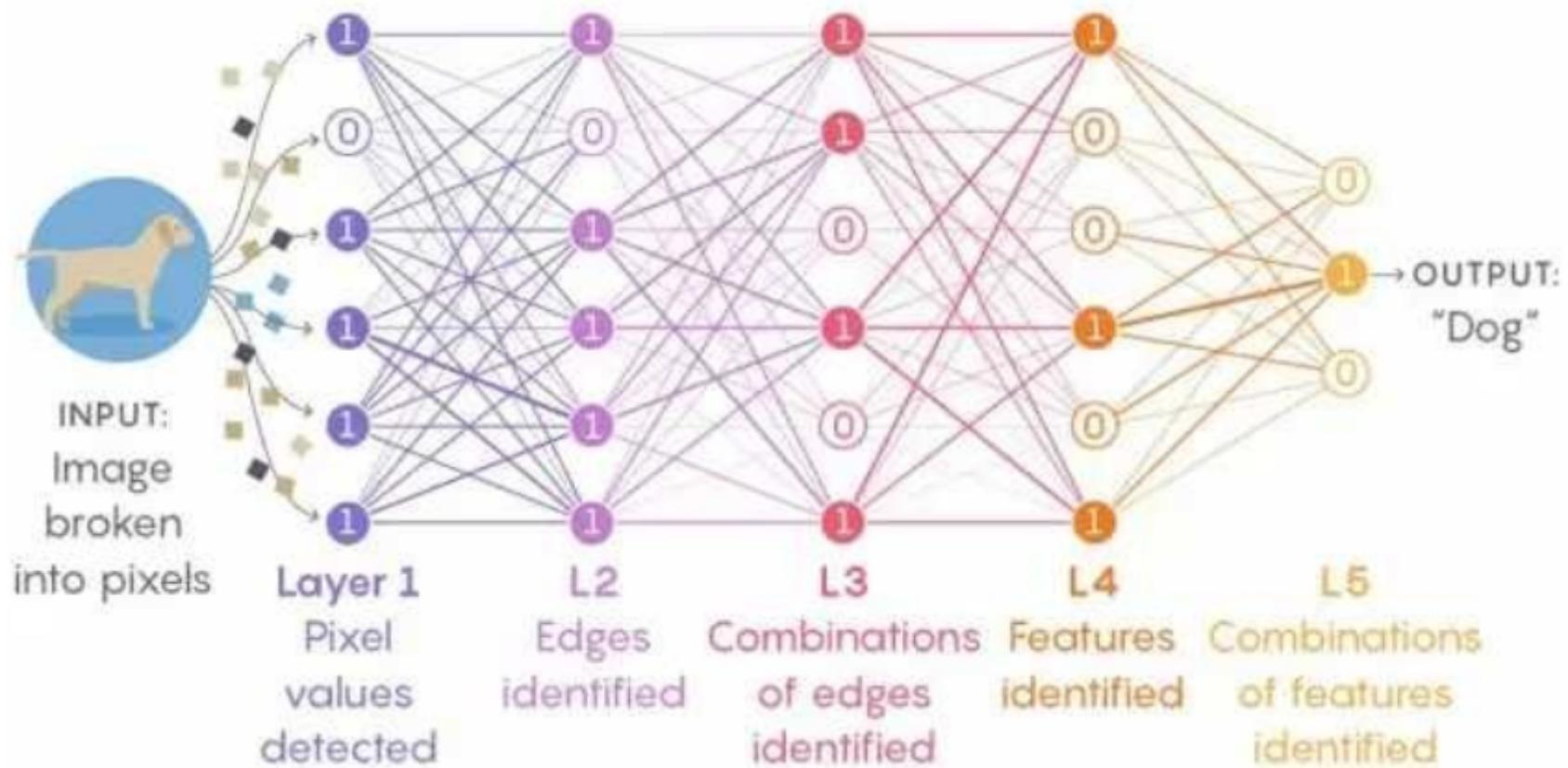
Computer Science & Engineering



How to reduce uncertainty?



Information Extraction

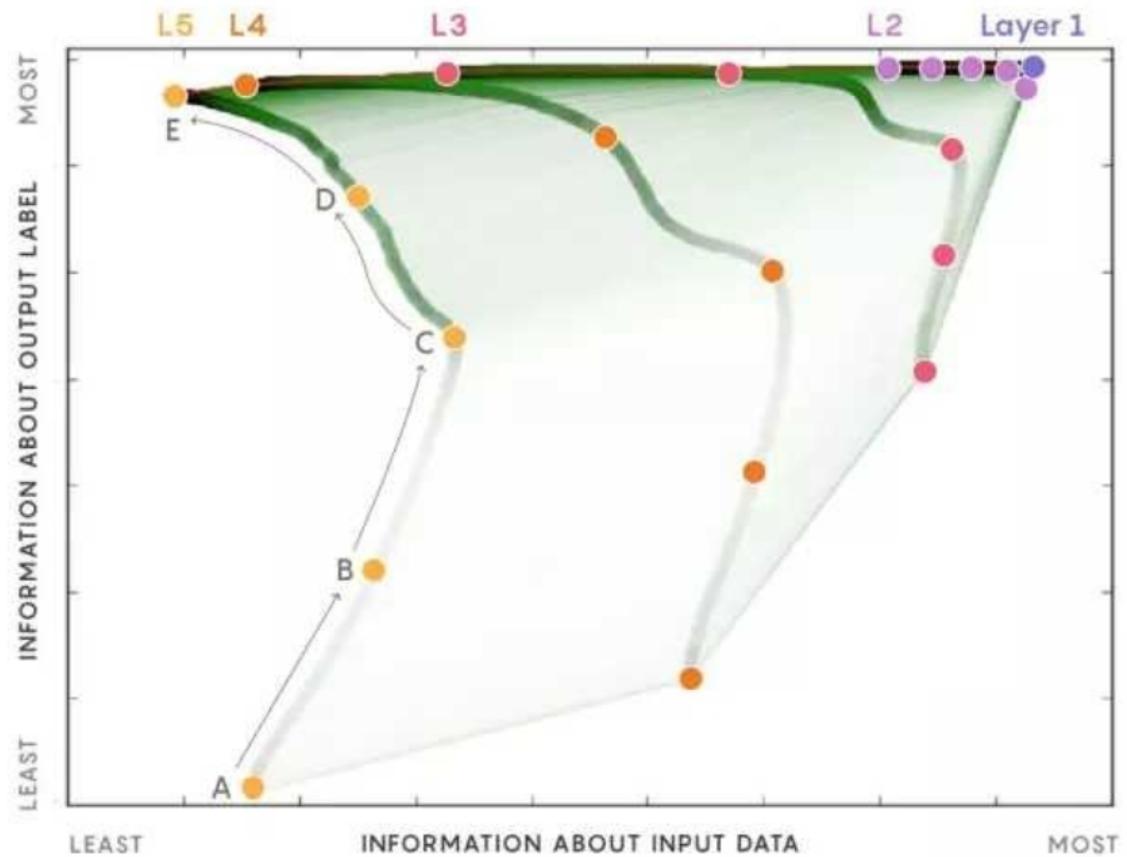


Inside Deep Learning

Inside Deep Learning

New experiments reveal how deep neural networks evolve as they learn.

- A **INITIAL STATE:** Neurons in Layer 1 encode everything about the input data, including all information about its label. Neurons in the highest layers are in a nearly random state bearing little to no relationship to the data or its label.
- B **FITTING PHASE:** As deep learning begins, neurons in higher layers gain information about the input and get better at fitting labels to it.
- C **PHASE CHANGE:** The layers suddenly shift gears and start to "forget" information about the input.
- D **COMPRESSION PHASE:** Higher layers compress their representation of the input data, keeping what is most relevant to the output label. They get better at predicting the label.
- E **FINAL STATE:** The last layer achieves an optimal balance of accuracy and compression, retaining only what is needed to predict the label.



Why multilayer matters?

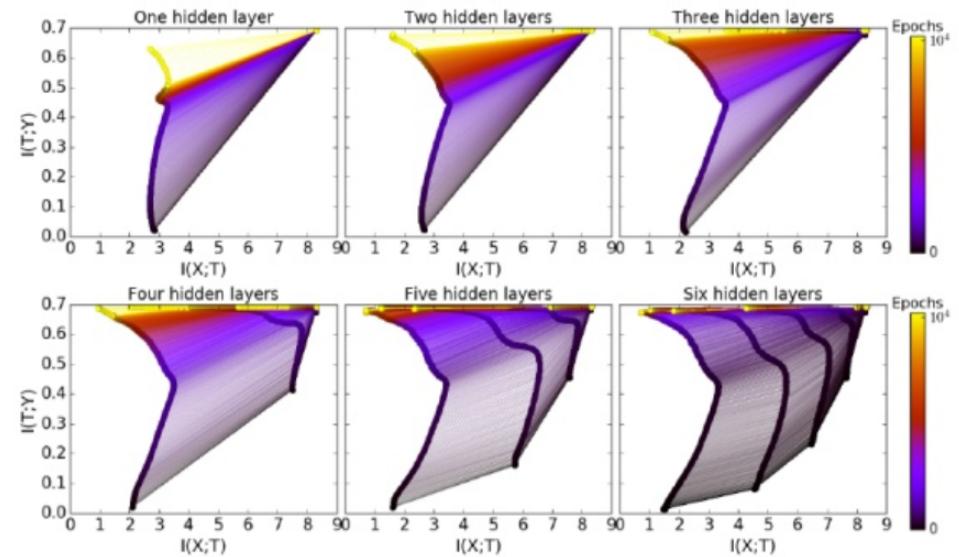
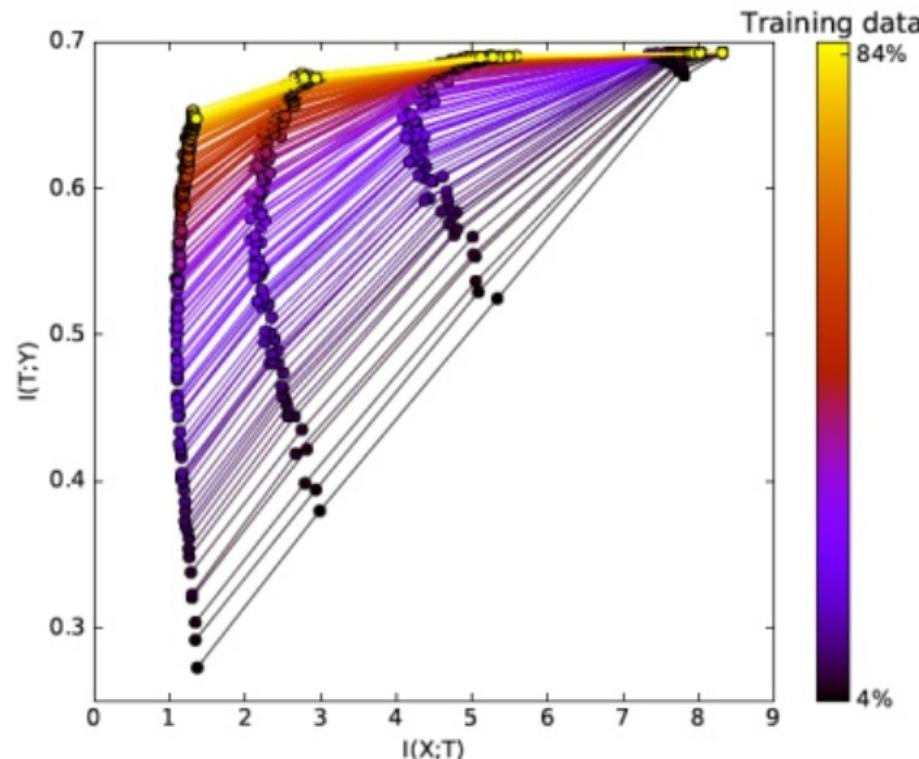


Figure 5: The layers information paths during the SGD optimization for different architectures. Each panel is the *information plane* for a network with a different number of hidden layers. The width of the hidden layers start with 12, and each additional layer has 2 fewer neurons. The final layer with 2 neurons is shown in all panels. The line colors correspond to the number of training epochs.

data size can increase generalization

Multilayer can reduce epochs ! (why?)