

CSE 417T

Introduction to Machine Learning

Lecture 22

Instructor: Chien-Ju (CJ) Ho

Logistics

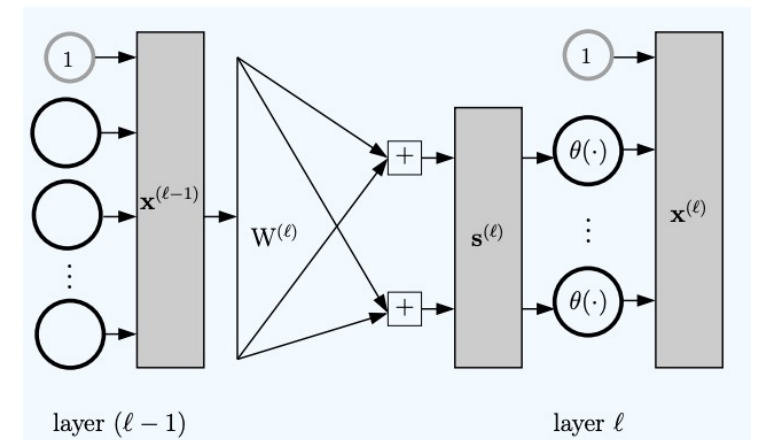
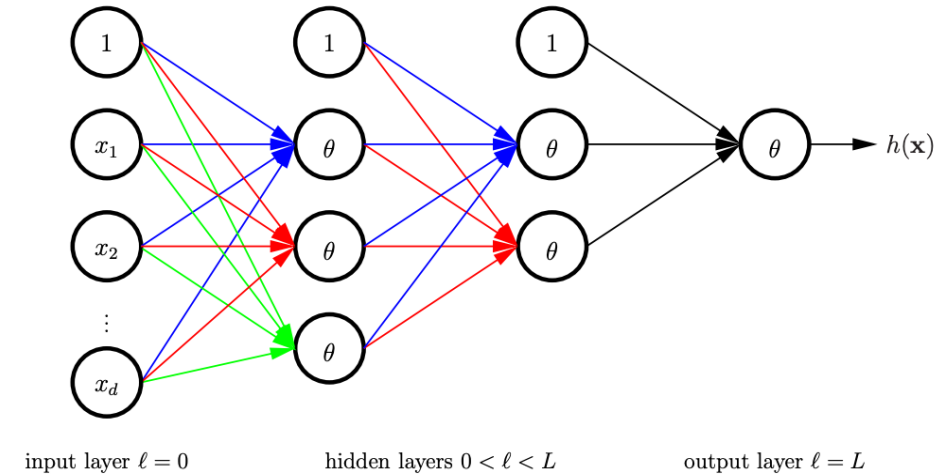
- Homework 5 is due Apr 19 (Tuesday)
- Exam 2 will be on April 28 (Thursday)
 - Will focus on the topics in the second half of the semester
 - Format / logistics will be similar with what we have in Exam 1
 - Timed exam (75 min) during lecture time in the classroom
 - Closed-book exam with 2 letter-size cheat sheets allowed (4 pages in total)
 - No format limitations (it can be typed, written, or a combination)
 - April 26 (Tuesday) will be a review lecture

Recap

Neural Networks (NN)

- Notations:
 - $\ell = 0$ to L : layer
 - $d^{(\ell)}$: dimension of layer ℓ
 - $\vec{x}^{(\ell)}$: the nodes in layer ℓ
 - $w_{i,j}^{(\ell)}$: weights; characterize hypothesis in NN
 - $s_j^{(\ell)} = \sum_{i=0}^{d^{(\ell-1)}} w_{i,j}^{(\ell)} x_i^{(\ell-1)}$: linear signals
 - θ : activation function
 - $x_j^{(\ell)} = \theta(s_j^{(\ell)})$

Feed-forward network

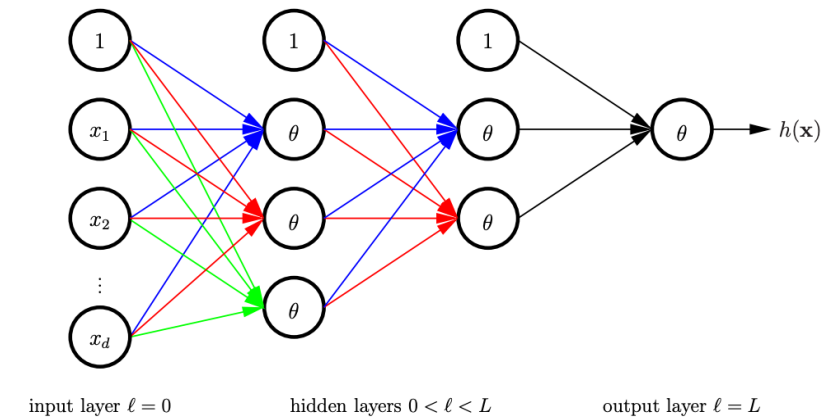


Forward Propagation and Backpropagation

- Evaluate $h(\vec{x})$ given h (characterized by $\{w_{i,j}^{(\ell)}\}$)
 - Forward propagation

$$\mathbf{x} = \mathbf{x}^{(0)} \xrightarrow{w^{(1)}} \mathbf{s}^{(1)} \xrightarrow{\theta} \mathbf{x}^{(1)} \xrightarrow{w^{(2)}} \mathbf{s}^{(2)} \xrightarrow{\theta} \mathbf{x}^{(2)} \dots \xrightarrow{w^{(L)}} \mathbf{s}^{(L)} \xrightarrow{\theta} \mathbf{x}^{(L)} = h(\mathbf{x}).$$

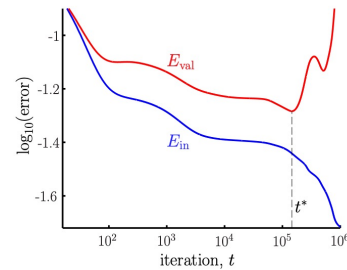
- Given D , learn the weights $W = \{w_{i,j}^{(\ell)}\}$
 - Backpropagation
 - Initialize $w_{i,j}^{(\ell)}$ randomly
 - For $t = 1$ to T
 - Randomly pick a point from D (for stochastic gradient descent)
 - Forward propagation: Calculate all $x_i^{(\ell)}$ and $s_i^{(\ell)}$
 - Backward propagation: Calculate all $\delta_j^{(\ell)}$
 - Update the weights $w_{i,j}^{(\ell)} \leftarrow w_{i,j}^{(\ell)} - \eta \delta_j^{(\ell)} x_i^{(\ell-1)}$
 - Return the weights



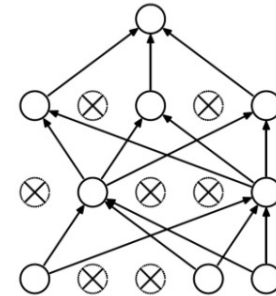
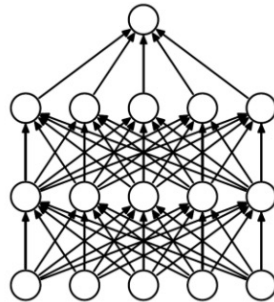
Regularizations in Neural Networks

- Weight-based regularization

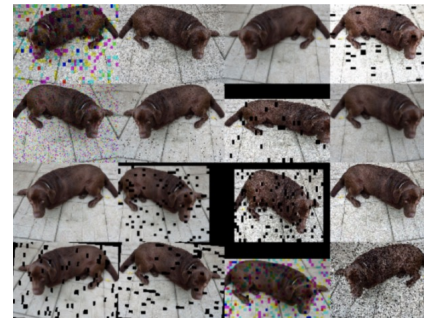
- Early stopping



- Dropout



- Adding noises



Today's Lecture

The notes are not intended to be comprehensive. They should be accompanied by lectures and/or textbook.
Let me know if you spot errors.

Deep Learning

Neural networks with many layers

Single Hidden-Layer Neural Network

- How do we write a hypothesis in a single-hidden layer NN mathematically?

- $$h(\vec{x}) = \theta \left(w_{0,1}^{(2)} + \sum_{j=1}^{d^{(1)}} w_{j,1}^{(2)} x_j^{(1)} \right)$$
$$= \theta \left(w_{0,1}^{(2)} + \sum_{j=1}^{d^{(1)}} w_{j,1}^{(2)} \theta \left(\sum_{i=0}^{d^{(0)}} w_{i,j}^{(1)} x_i^{(0)} \right) \right)$$

- How do we write a linear model with nonlinear transform

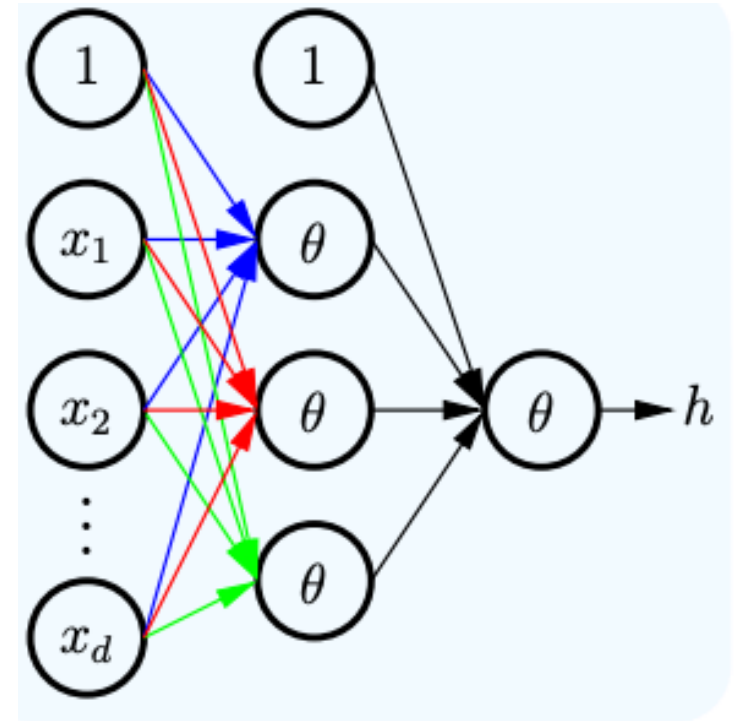
- $$h(\vec{x}) = \theta(w_0 + \sum w_i \phi_i(\vec{x}))$$

- How do we write a Kernel SVM hypothesis

- $$g(\vec{x}) = \theta \left(b^* + \sum_{\alpha_n^* > 0} \alpha_n^* y_n K(\vec{x}_n, \vec{x}) \right)$$

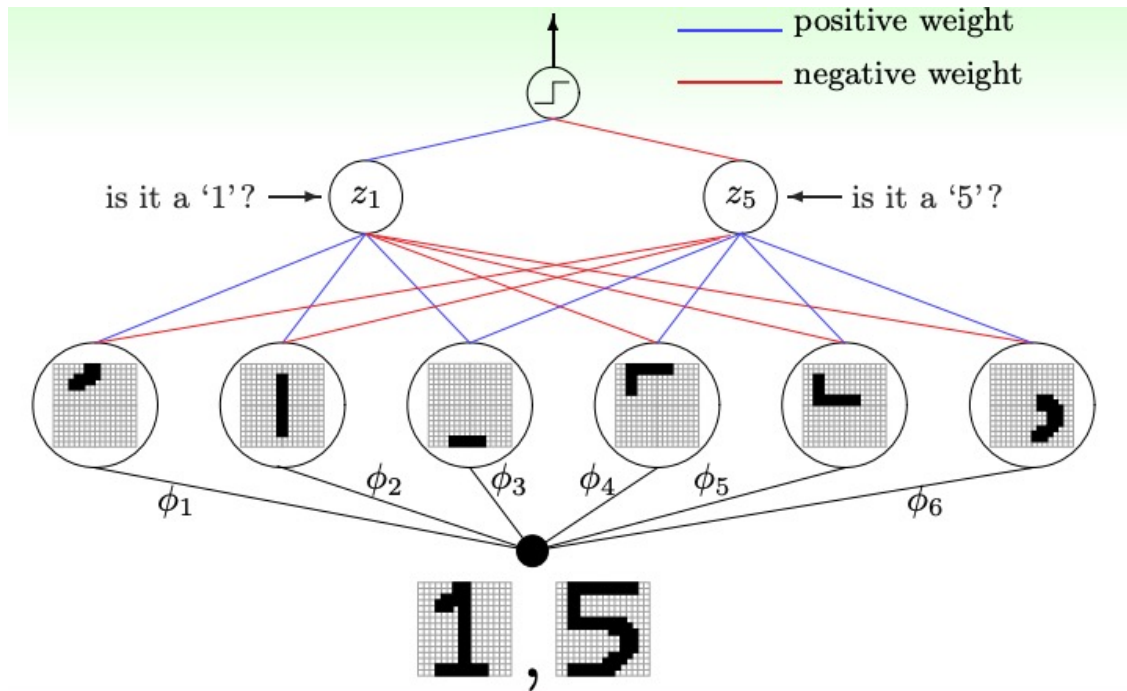
- Interpretation:

- The hidden layer is like **feature transform**
 - Shallow learning vs. deep learning



Deep Neural Network

- “Shallow” neural network is powerful (universal approximation theorem holds with a single hidden layer). Why “deep” neural networks?



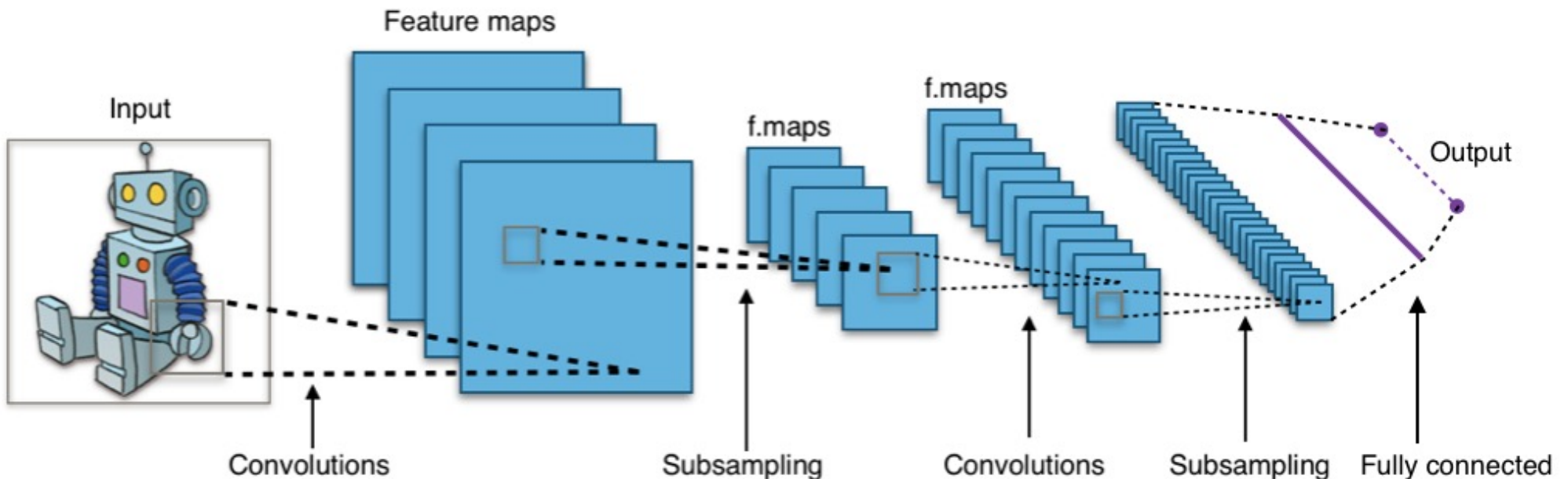
Each layer captures **features** of the previous layers.

We can use “raw data” (e.g., pixels of an image) as input. The hidden layer are extracting the **features**.

Design different **network architectures** to incorporate domain knowledge.

Convolutional Neural Networks (CNN)

- Captures the localized properties of features hierarchically



Convolutional Filters

- A convolutional filter is like a matrix version of a dot product.

| | | | | | |
|---|---|---|---|---|---|
| 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

- A convolutional filter is like a matrix version of a dot product.

| | | | | | |
|---|---|---|---|---|---|
| 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 | | | |
| | | | |
| | | | |
| | | | |

$$\begin{aligned} & (0 * 0) + (0 * 1) + (0 * 0) + \\ & (0 * 1) + (1 * -4) + (2 * 1) + \\ & (0 * 0) + (2 * 1) + (4 * 0) \\ & = 0 \end{aligned}$$

Convolutional Filters

- A convolutional filter is like a matrix version of a dot product.

| | | | | | |
|---|---|---|---|---|---|
| 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 | | | |
| | | | |
| | | | |
| | | | |

Convolutional Filters

- A convolutional filter is like a matrix version of a dot product.

| | | | | | |
|---|---|---|---|---|---|
| 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 | | |
| | | | |
| | | | |
| | | | |

$$\begin{aligned} & (0 * 0) + (0 * 1) + (0 * 0) + \\ & (1 * 1) + (2 * -4) + (2 * 1) + \\ & (2 * 0) + (4 * 1) + (4 * 0) \\ & = -1 \end{aligned}$$

Convolutional Filters

- A convolutional filter is like a matrix version of a dot product.

| | | | | | |
|---|---|---|---|---|---|
| 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 |

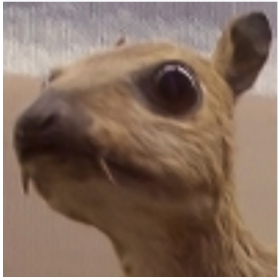
*

| | | |
|---|----|---|
| 0 | 1 | 0 |
| 1 | -4 | 1 |
| 0 | 1 | 0 |

=

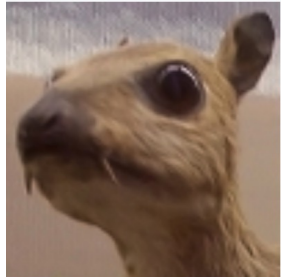
| | | | |
|----|----|----|----|
| 0 | -1 | -1 | 0 |
| -2 | -5 | -5 | -2 |
| 2 | -2 | -1 | 3 |
| -1 | 0 | -5 | 0 |





Convolutional Filters



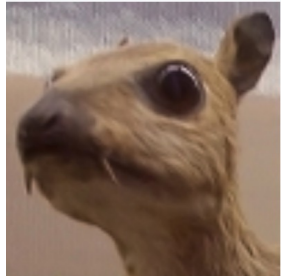
| Operation | Kernel ω | Image result $g(x,y)$ |
|-----------|---|-----------------------|
| | $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ | |
| | $\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$ | |
| | $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 &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

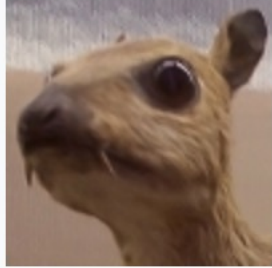
Convolutional Filters



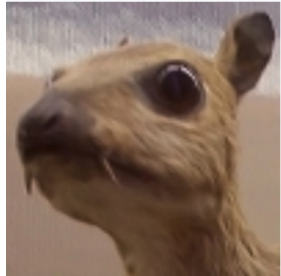
| Operation | Kernel ω | Image result $g(x,y)$ |
|----------------|---|---|
| Identity | $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ |  |
| Edge detection | $\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$ |  |
| | $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1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

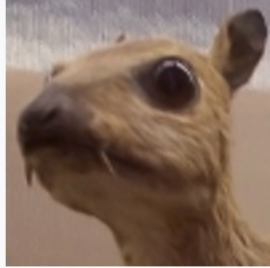
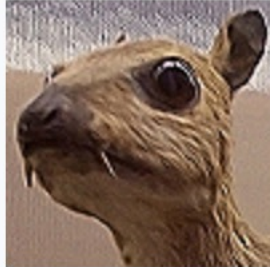
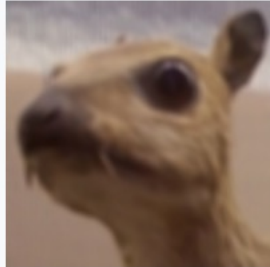
Convolutional Filters



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

Convolutional 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) | $\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ |  |

Connection to Neural Networks

- Convolutions can be represented by a network structure
 - Nodes in the previous layer are only connected to “adjacent” nodes in the next layer.
 - Many of the weights have the same value.

| | | | | | |
|---|---|---|---|---|---|
| 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 |

Pooling Layers

- Commonly used in convolutional neural networks.

A subsampling / down-sampling process:

- Combines multiple adjacent nodes into a single node

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

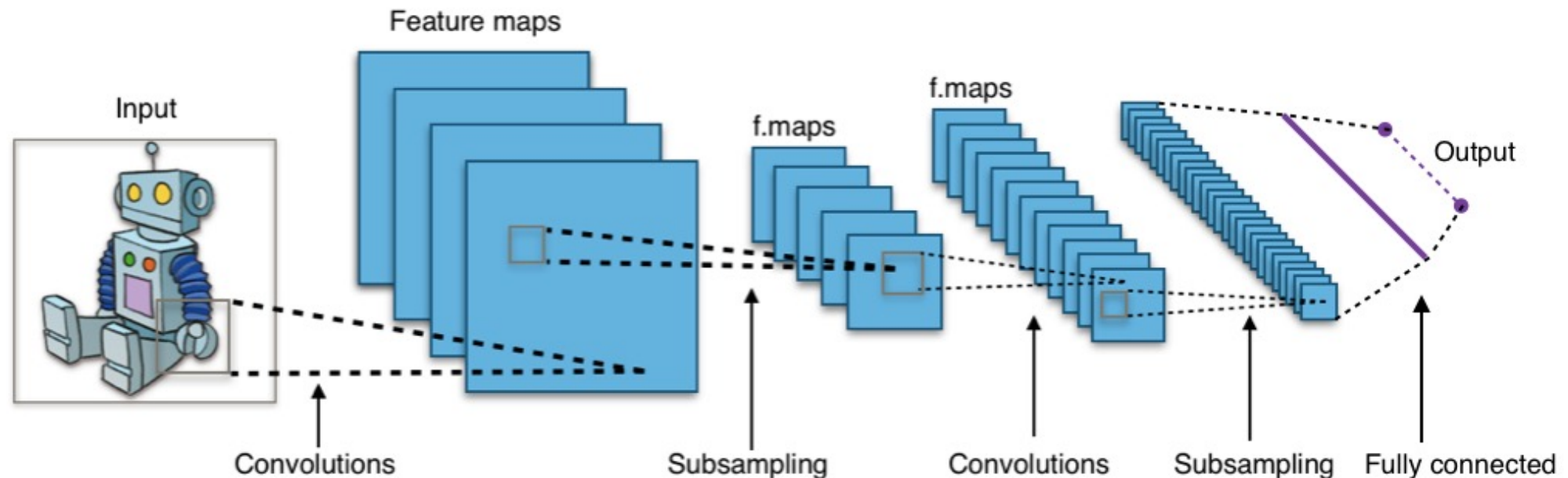
*average
pooling*

| | |
|----|----|
| -2 | -2 |
| 0 | 0 |

- Reduce the dimensionality of input. More robust to noise.

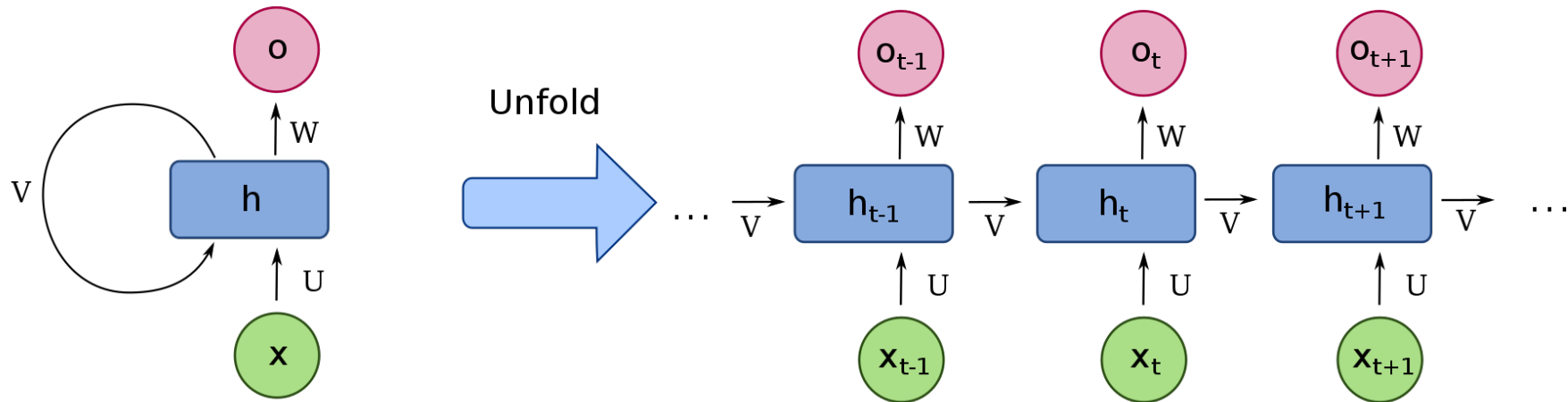
Convolutional Neural Networks (CNN)

- Captures the localized properties of features
 - Particularly suitable for computer vision (images)
 - Go (AlphaGo) is another famous application of CNN



Another Example Network Structure [Safe to Skip for the Exam]

- Recurrent Neural Network (RNN)
 - Aim to deal with time-series data, such as natural language processing
 - Using hidden layers to store temporal information
 - Allow previous outputs to be used as inputs and keep hidden states



Some Techniques in Improving Deep Learning

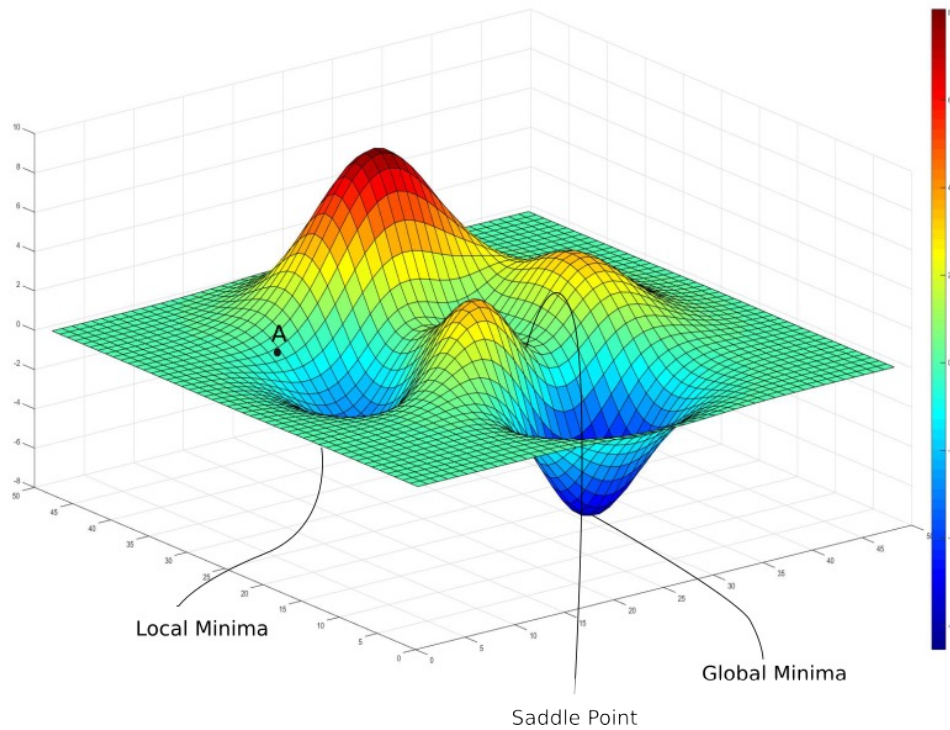
- Regularization to mitigate overfitting
 - Weight-based, early stopping, dropout, etc
- Incorporating domain knowledges
 - Network architectures (e.g., Convolutional Neural Nets)
- Improving computation with huge amount of data
 - Hardware architecture to improve parallel computation
- Improving gradient-based optimization
 - Choosing better **initialization** points

Initialization

Why initialization matters in deep learning

- Error is nonconvex in NN
- Vanishing/exploding gradient problem

Error is Nonconvex in Neural Networks



- We mostly adopt gradient-descent-style algorithms for optimization.
- No guarantee to converge to global optimal.
- Need to run it many times.
- Initialization matters.

Vanishing Gradient Problem

- Backpropagation

- $\frac{\partial e_n(W)}{\partial w_{i,j}^{(\ell)}} = \delta_j^{(\ell)} x_i^{(\ell-1)}$

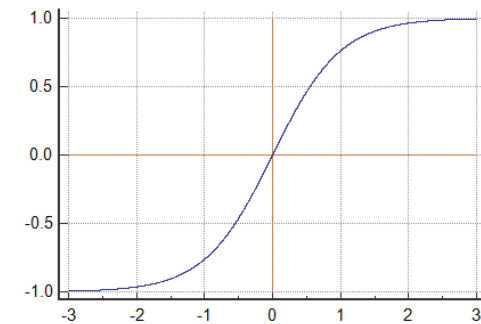
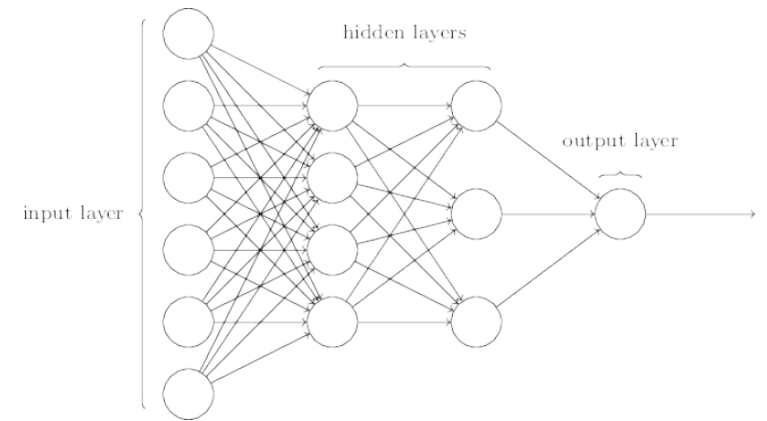
- $\delta_j^{(\ell)} = \theta' \left(s_j^{(\ell)} \right) \sum_{k=1}^{d^{(\ell+1)}} \delta_k^{(\ell+1)} w_{j,k}^{(\ell+1)}$

- If we use activation function $\theta(s) = \tanh(s)$

- $\theta'(s) = 1 - \theta(s)^2 < 1$

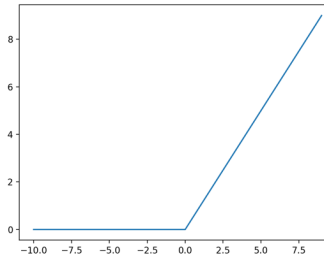
- In deep learning with a lot of layers,

- the gradient might vanish
 - hard to update the early layers



Vanishing Gradient Problem

- $\delta_j^{(\ell)} = \theta' \left(s_j^{(\ell)} \right) \sum_{k=1}^{d^{(\ell+1)}} \delta_k^{(\ell+1)} w_{j,k}^{(\ell+1)}$
- There is also a corresponding “exploding gradient problem”
- What can we do
 - Choose different activation functions
 - One common choice is Rectified Linear Unit (ReLU) and its variant
 - $\theta(s) = \max(0, s)$
 - Choose better **initialization**
 - Many approaches



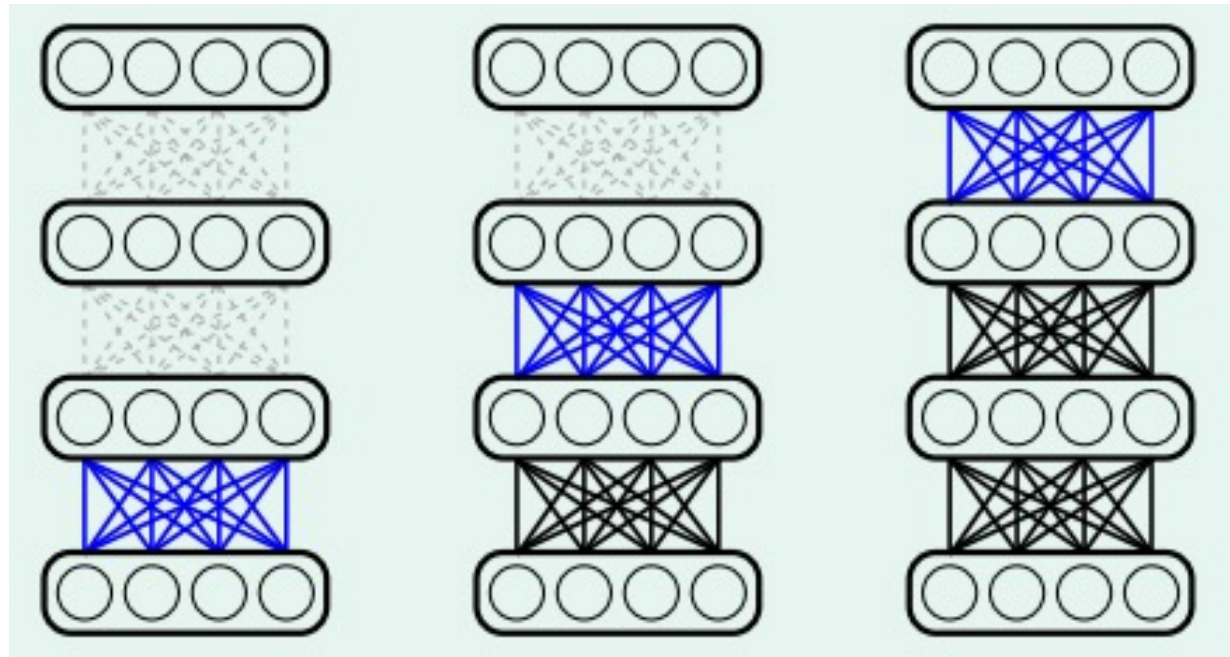
Weight Initialization

- Initializing weights to all 0 is a bad idea
 - Q6b of HW1
 - Hint: Look at the backpropagation formulation
- Randomly Initializing weights to regions so that vanishing/exploding gradients are less likely to happen
 - Activation-function dependent
 - e.g., Xavier initialization for tanh
- Learning the initialization that might be closer to the optimal
 - E.g., using autoencoder

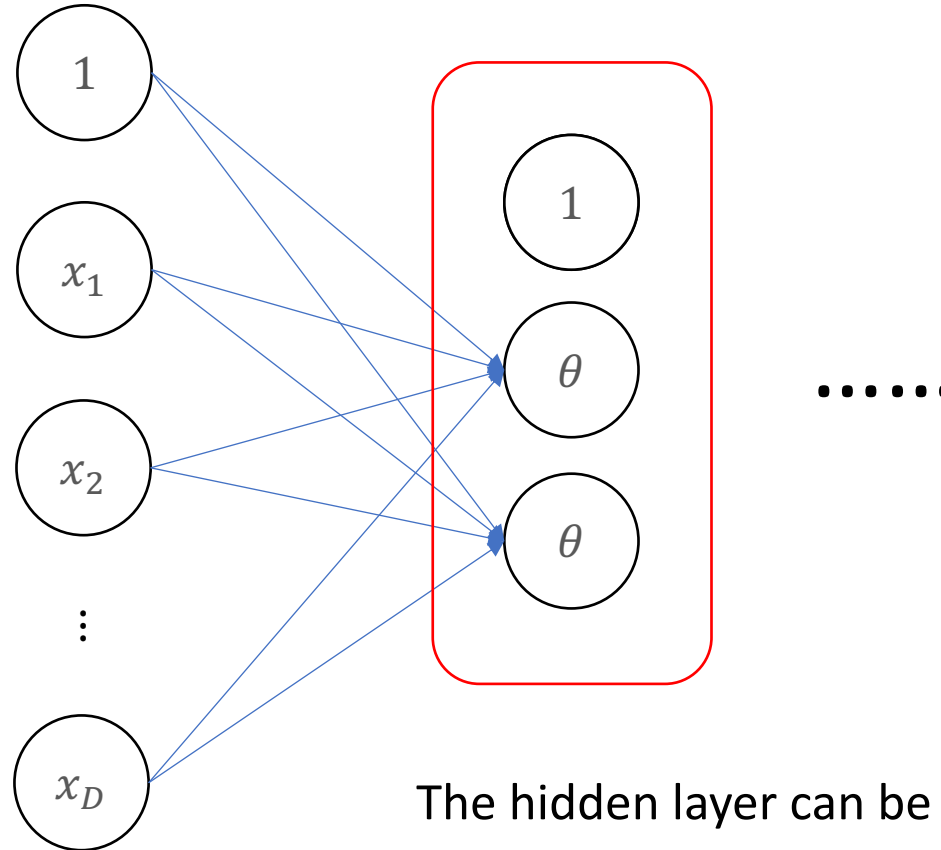
$$\delta_j^{(\ell)} = \theta' \left(s_j^{(\ell)} \right) \sum_{k=1}^{d^{(\ell+1)}} \delta_k^{(\ell+1)} w_{j,k}^{(\ell+1)}$$

Initialization

- Hard to initialize the entire network well.
- Intuition: Initialize the weights **layer by layer** such that each layer **preserves** the properties of the previous layer.



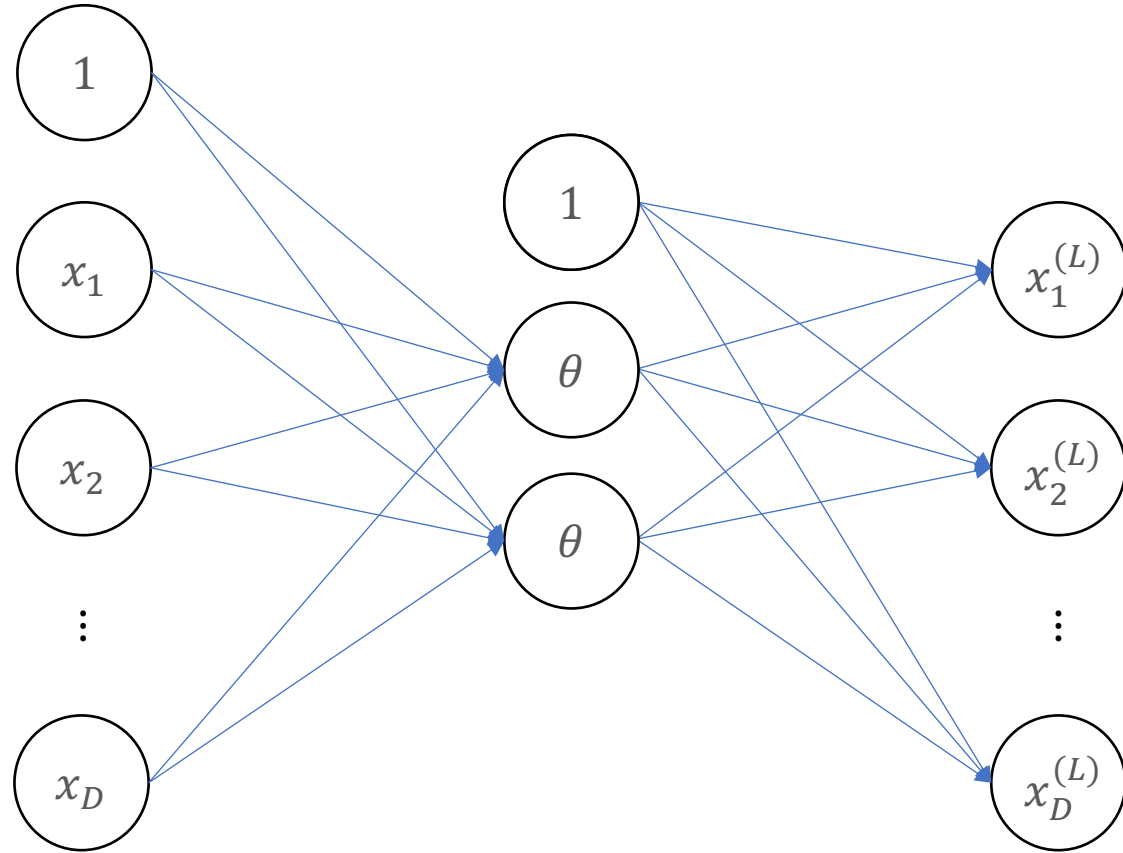
Autoencoders



The hidden layer can be considered as a feature transform.

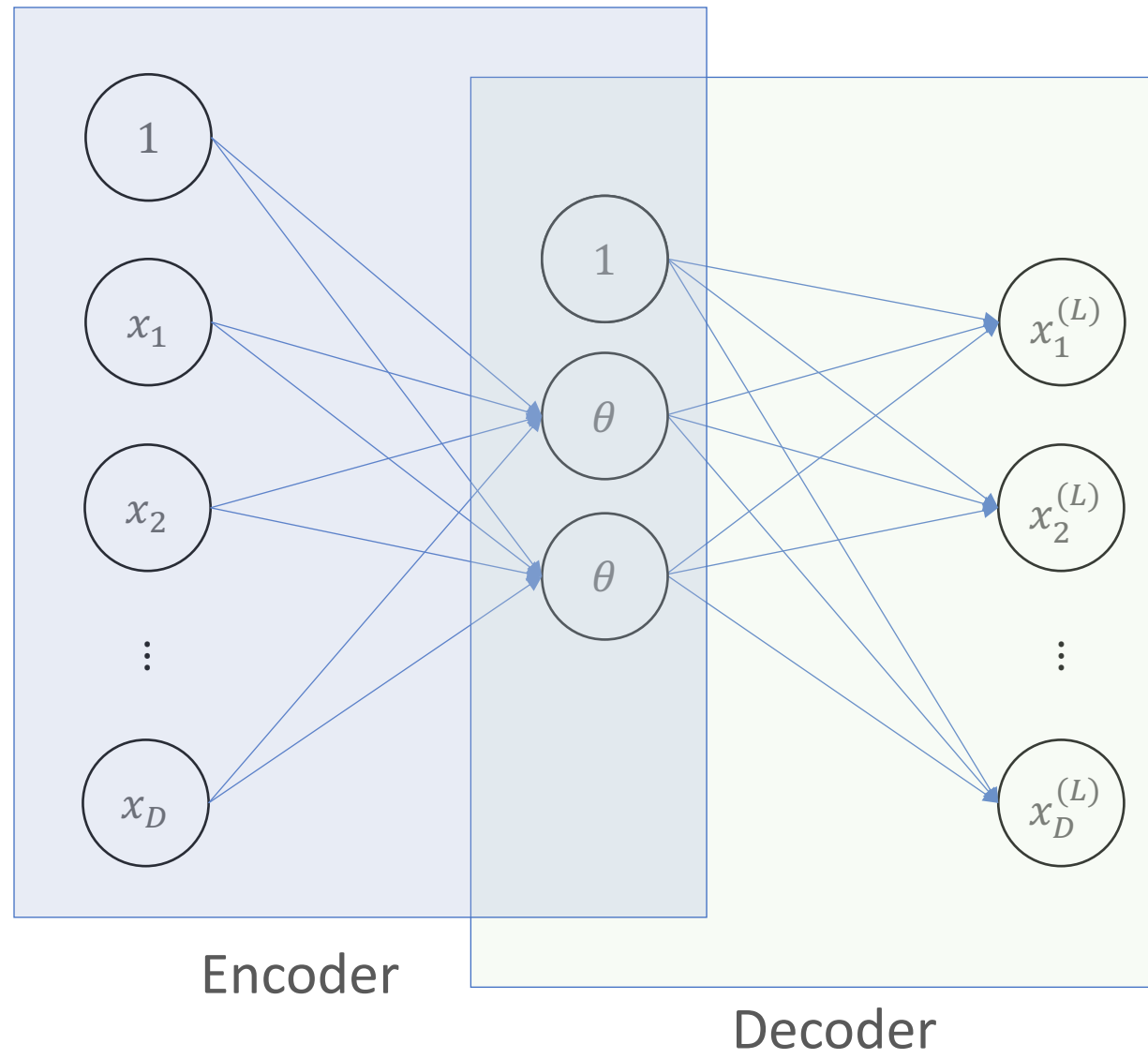
Can we ensure this transformation contain as much information about the original input as possible?

Autoencoders

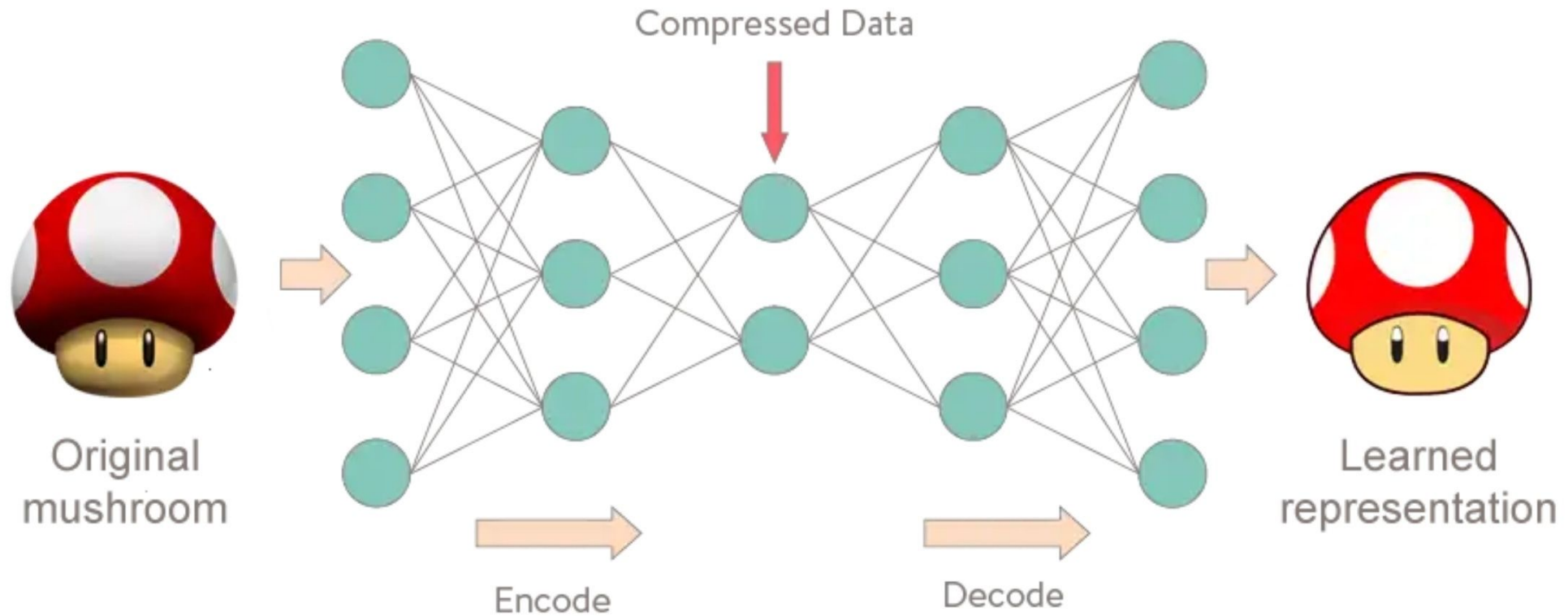


Minimize error of $\left\| \vec{x} - \vec{x}^L \right\|$

Autoencoders



Autoencoder



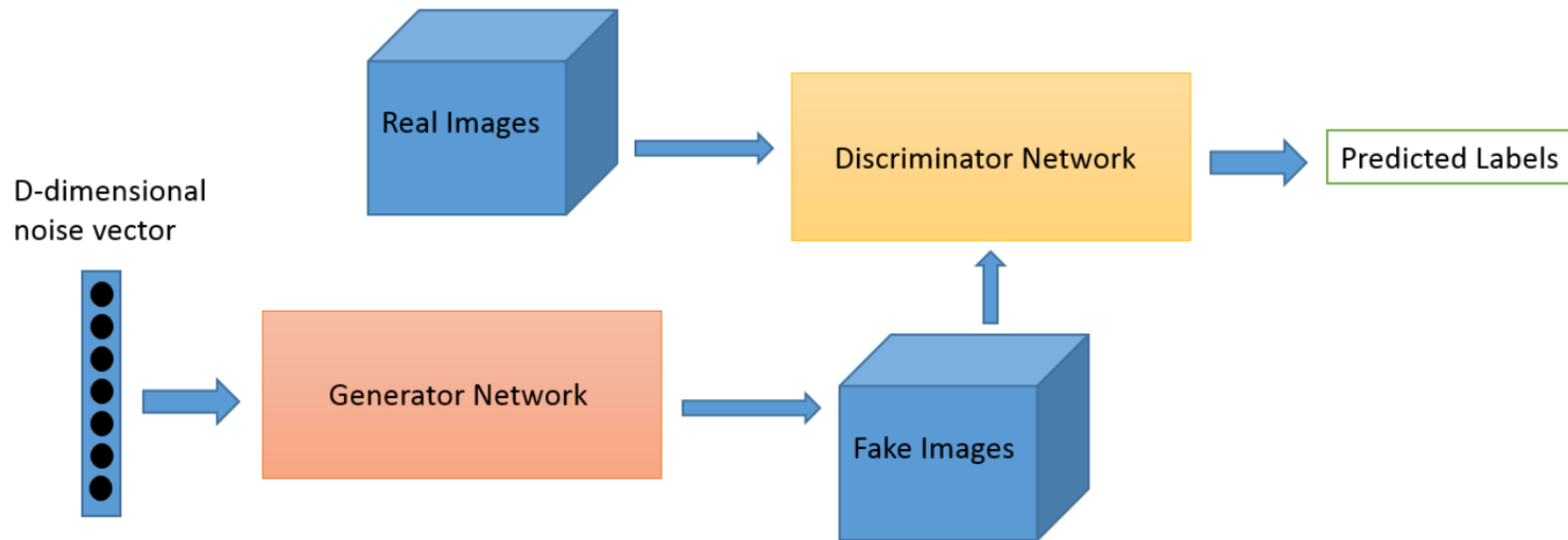
Unsupervised learning!

Cool Stuffs for Deep Learning

[Safe to Skip for the exam]

Generative Adversarial Nets (GAN)

- A Competition: Generator vs Discriminator
 - Discriminator wants to correctly classify the images (true images or not)
 - Generator wants to generate images that discriminator can't classify

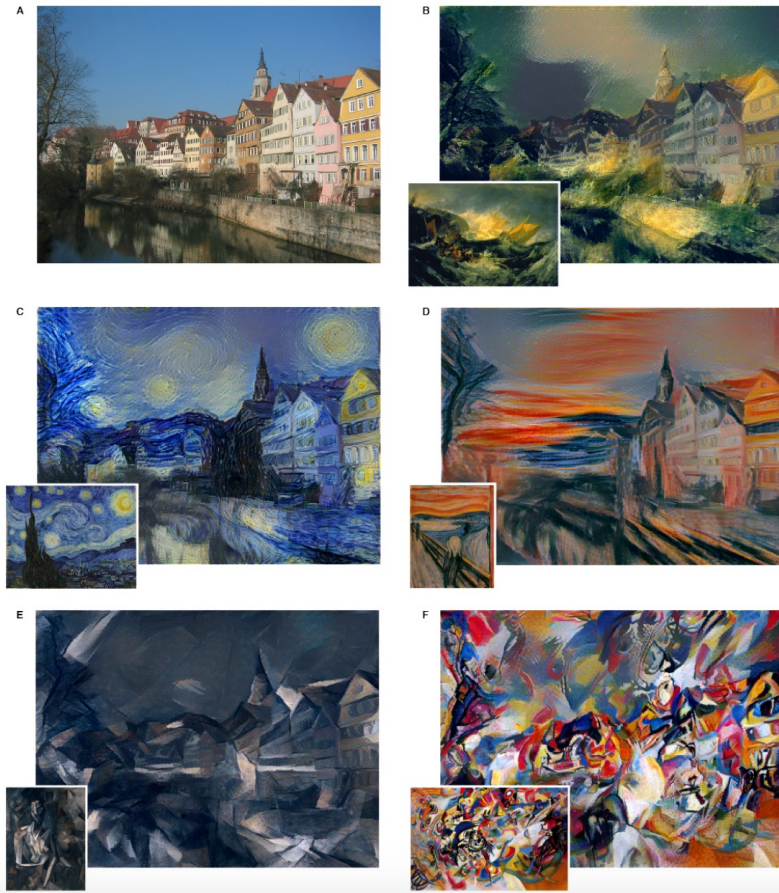


[Safe to Skip for the Exam]



<https://thisPersonDoesNotExist.com/>

Style Transfer



- Informal intuitions:
 - Recall that we can treat hidden layers as **feature transforms**
 - Deep learning is learning **representation** of data
 - How to achieve style transfer:
 - Learn a **content representation** for an image using hidden layers
 - Learn a **style representation** for an image using hidden layers
 - Compute an image that jointly minimizes the distance from the content image's content representation and the style image's style representation
 - <https://arxiv.org/pdf/1508.06576.pdf>

A Quick Flashback to RBF

RBF Function

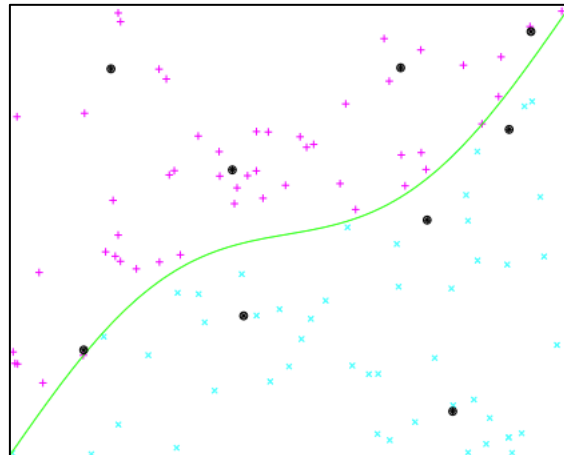
- $$h(\vec{x}) = \sum_{k=1}^K w_k \phi \left(\frac{\|\vec{x} - \vec{\mu}_k\|}{r} \right)$$

- Connection to linear models
 - Parametric RBF is essentially linear model with nonlinear transformation
- Connection to nearest neighbor
 - Radial Basis Function is defined by “similarity”
 - A prediction for a point is based on the “similarity” of the points to be predicted and other points

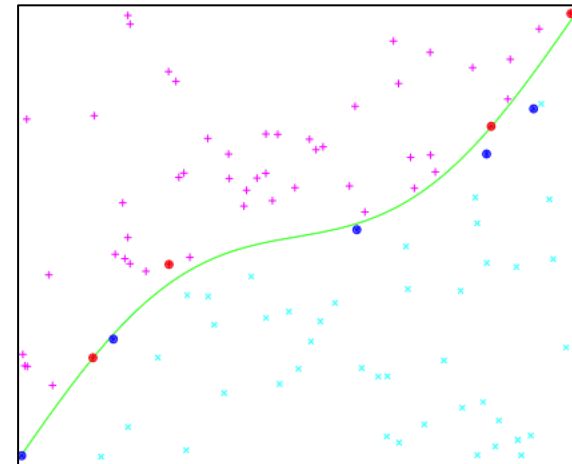
More Discussion on RBF

- $h(\vec{x}) = \sum_{k=1}^K w_k \phi\left(\frac{\|\vec{x} - \vec{\mu}_k\|}{r}\right)$
- Connection to SVM
 - Gaussian RBF Kernel: $g(\vec{x}) = \text{sign}\left(\sum_{\alpha_n^* > 0} \alpha_n^* y_n K(\vec{x}_n, \vec{x}) + b^*\right)$
 - Use **cluster centers** or **support vectors** to characterize a hypothesis

RBF

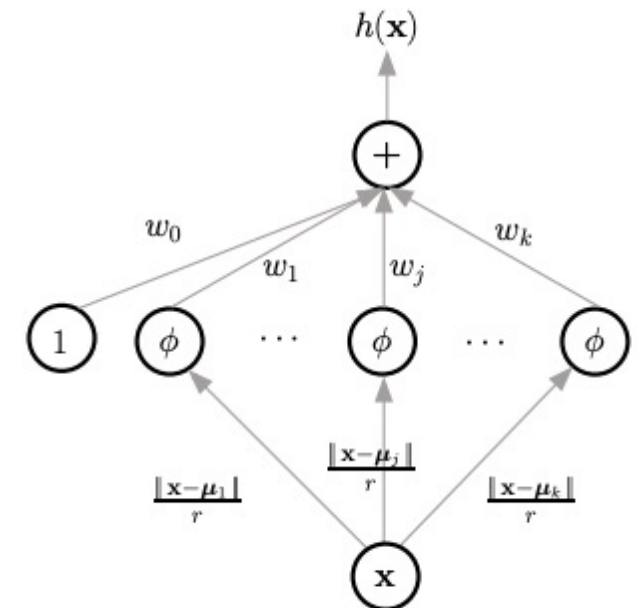


SVM



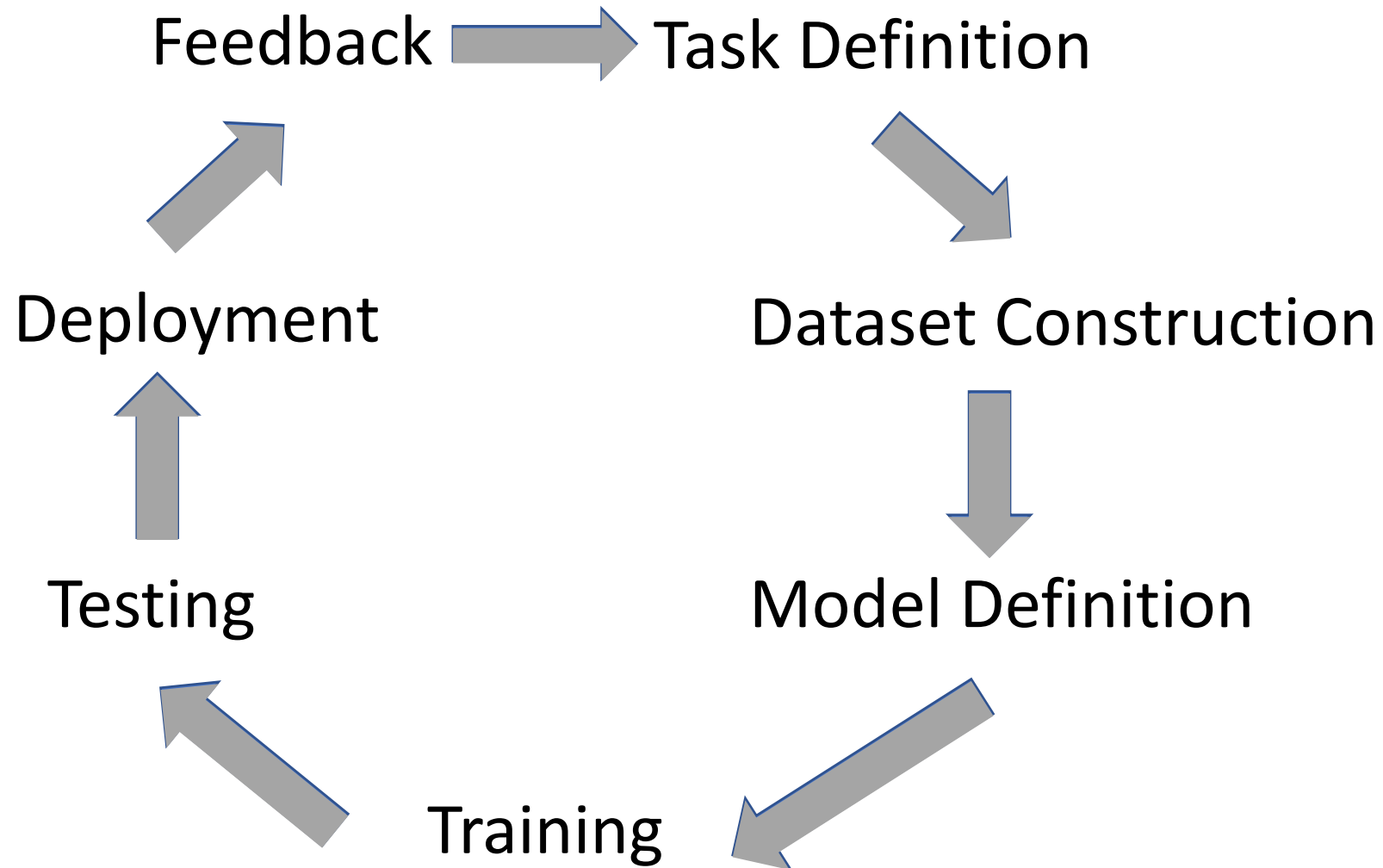
More Discussion on RBF

- $h(\vec{x}) = \sum_{k=1}^K w_k \phi\left(\frac{\|\vec{x} - \vec{\mu}_k\|}{r}\right)$
- Connection to Neural Network
 - RBF can be graphically presented:
 - (Similar to SVM, it's a single-hidden layer NN)

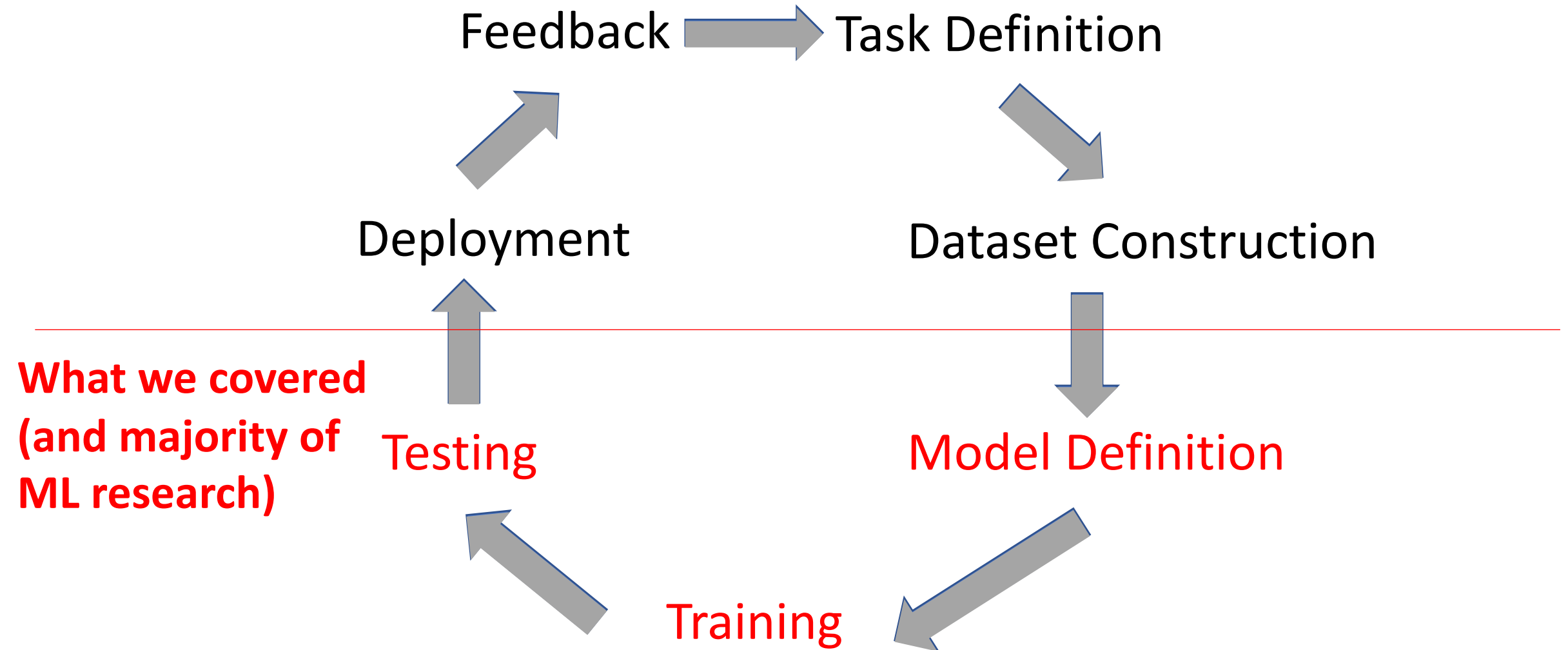


Machine Learning Life Cycle

Machine Learning Lifecycle



Machine Learning Lifecycle



Machine Learning Lifecycle

To have “positive” impacts,
we need to be careful in
every stage

Feedback → Task Definition

Deployment

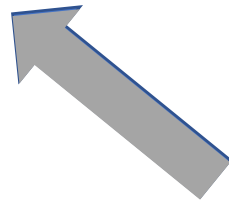
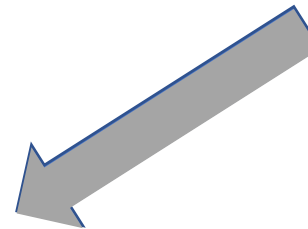
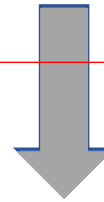
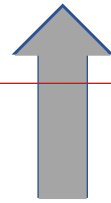
Dataset Construction

Testing

Model Definition

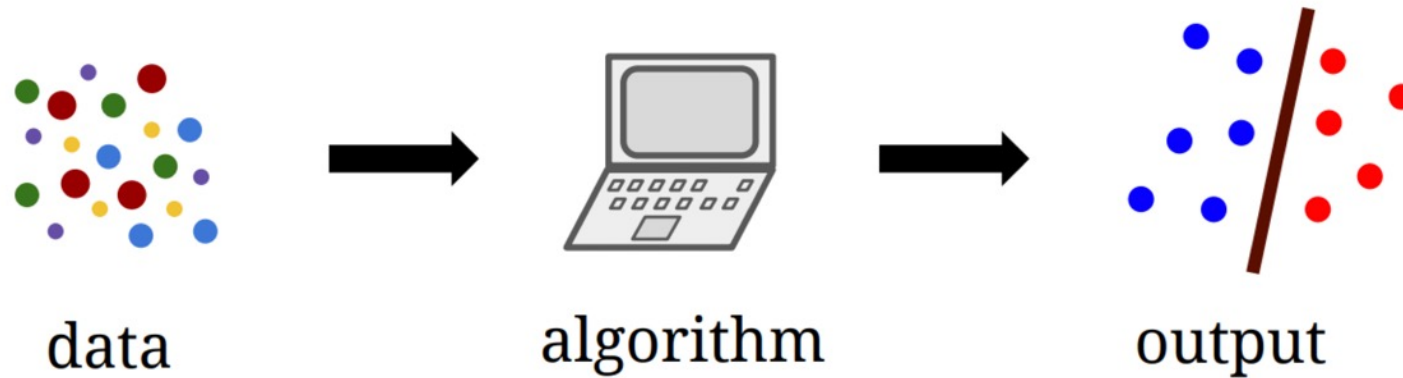
Training

What we covered
(and majority of
ML research)



Supervised Learning

- Standard setup of (supervised) machine learning

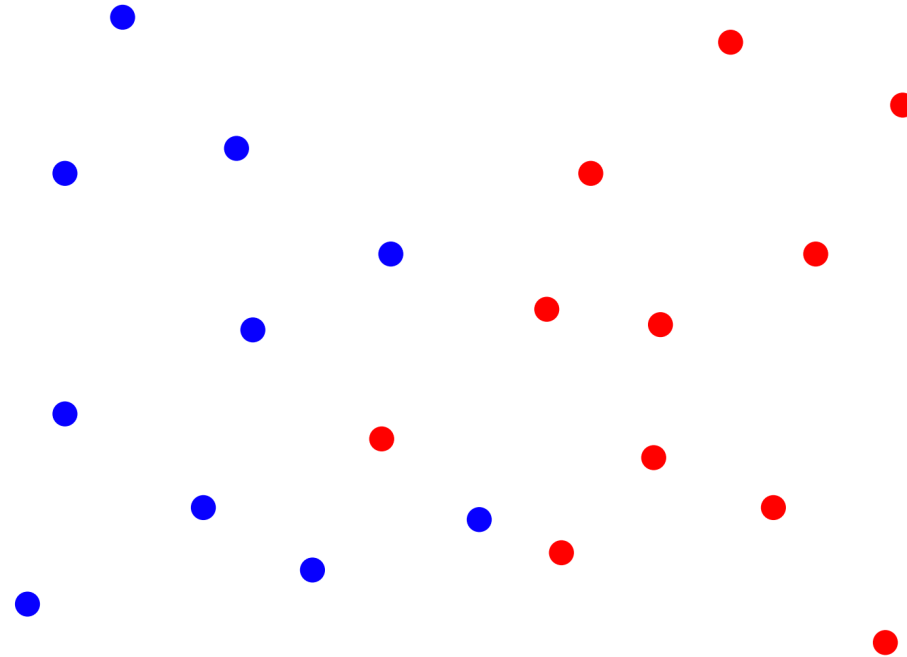


- Finding patterns from the given training datasets
 - Use the pattern to make predictions on new testing data
-
- Fundamental assumption:
 - Training and testing data points are i.i.d. drawn from the same distribution

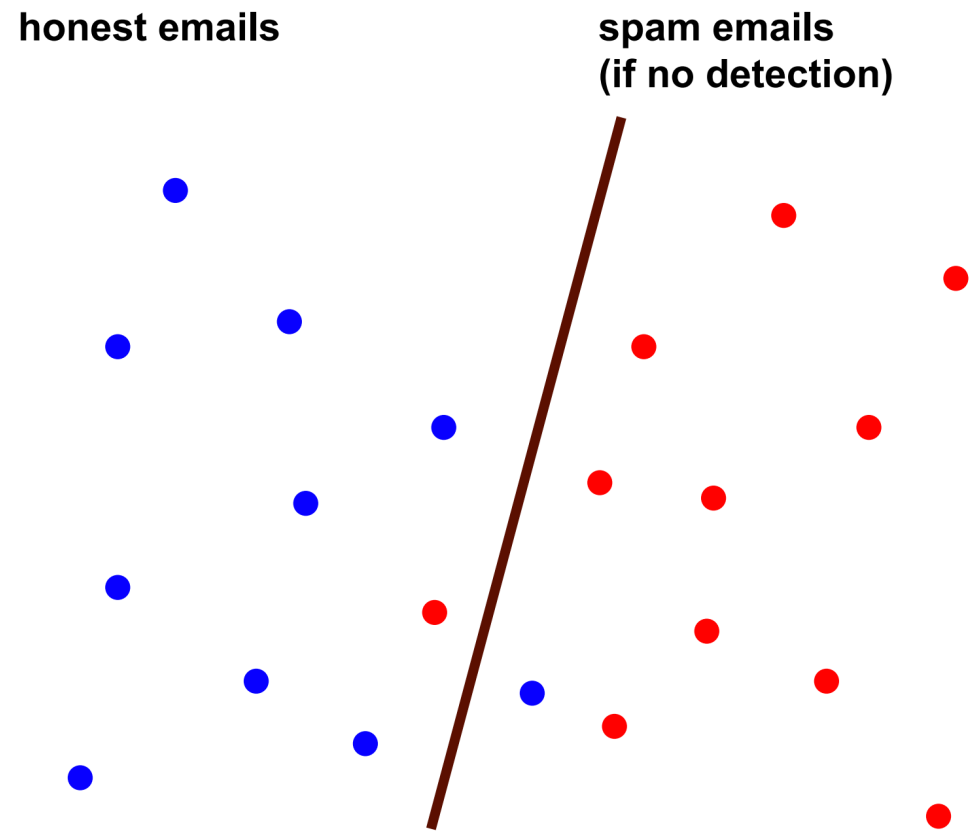
Example: Spam Filter

honest emails

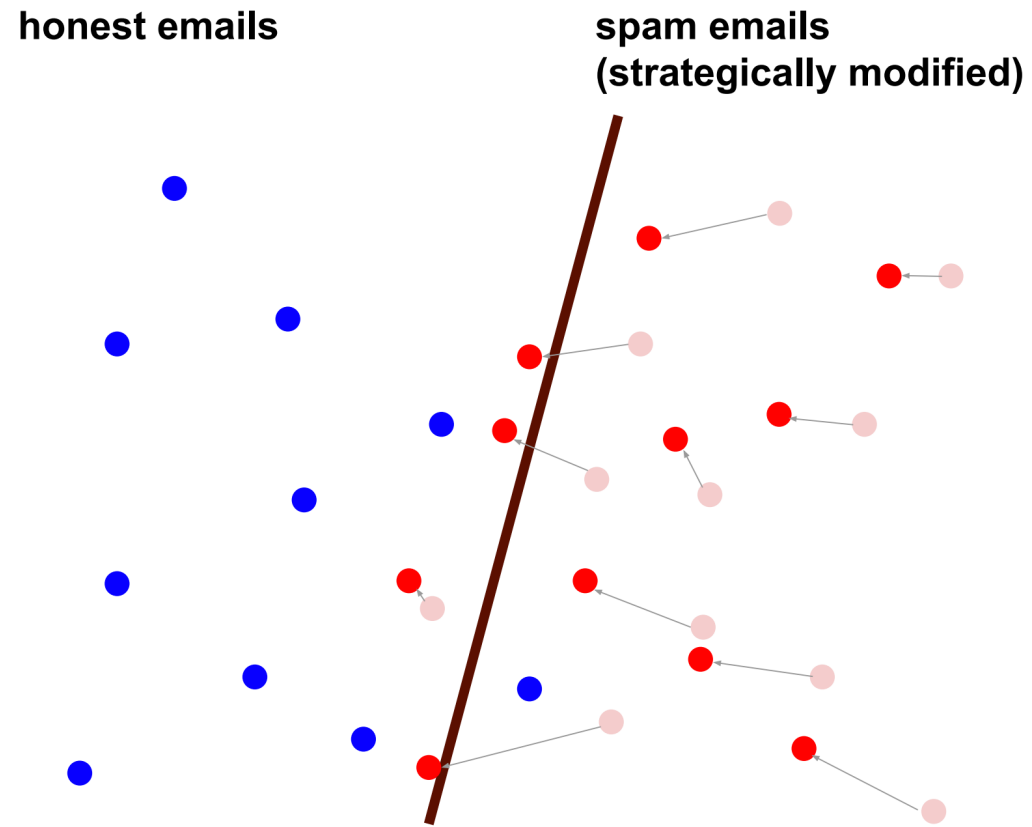
**spam emails
(if no detection)**



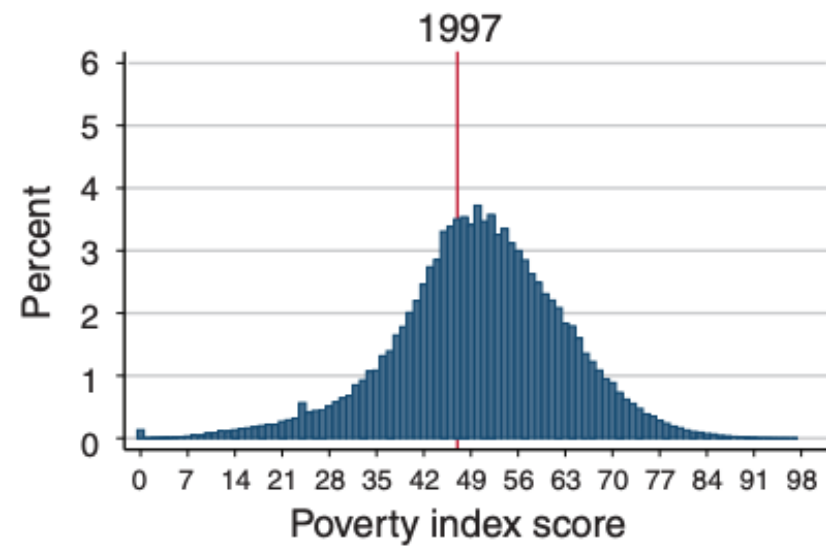
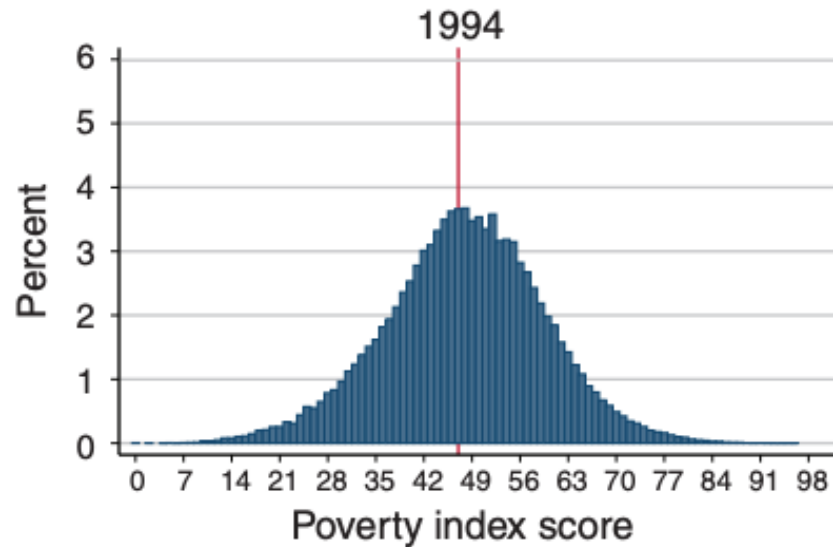
Example: Spam Filter



Example: Spam Filter



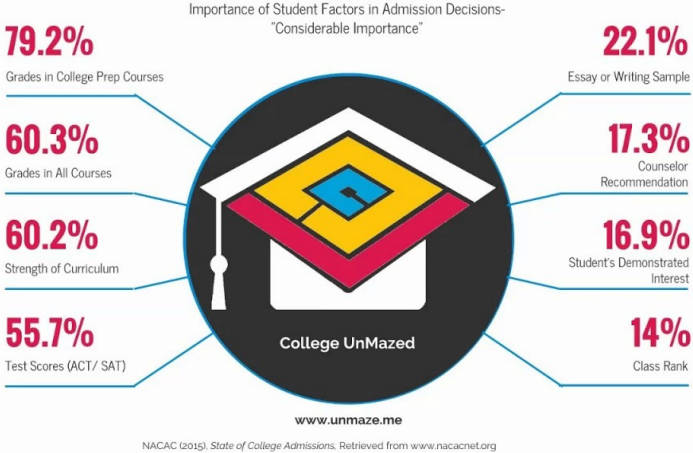
Social Program Eligibility [Camacho and Conover, 2012]



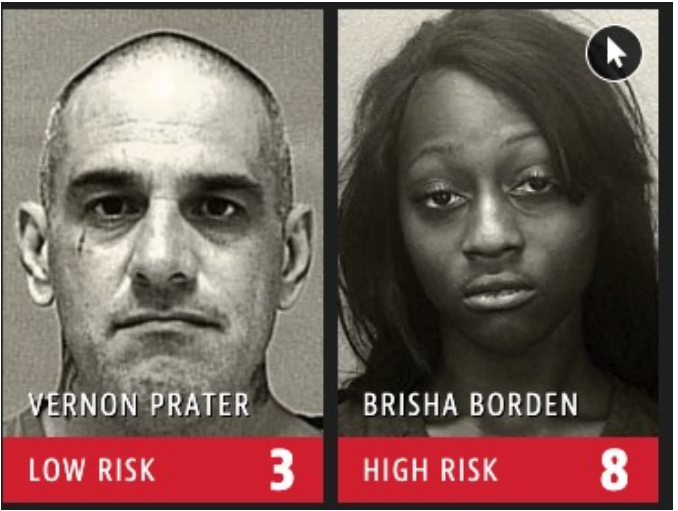
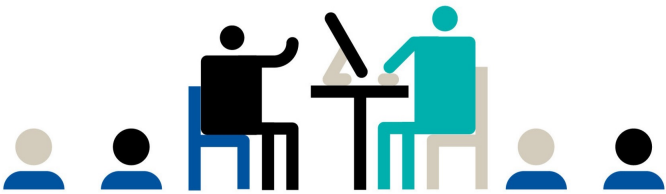
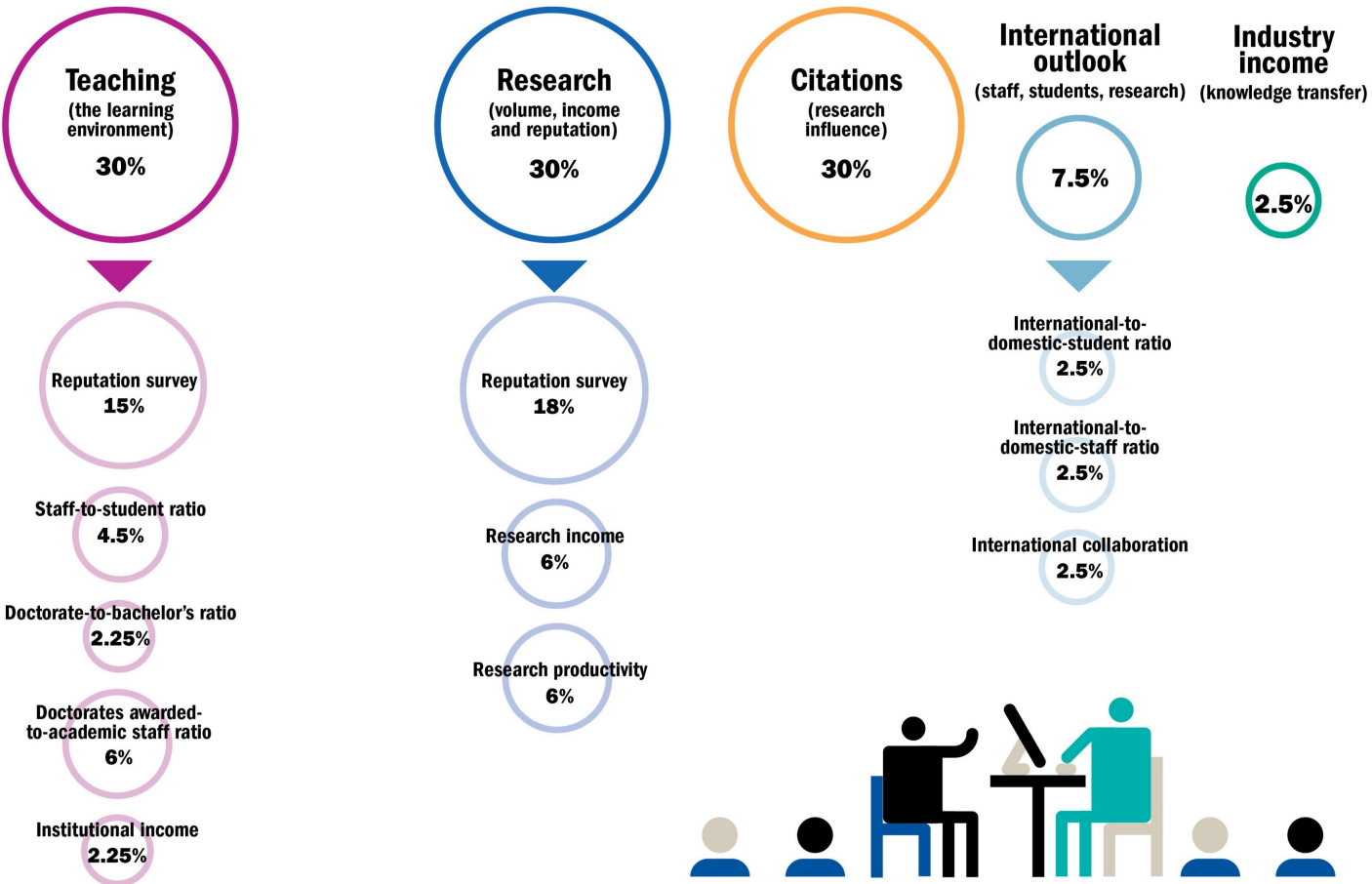
Goodhart's law:

“If a measure becomes the public's goal,
it is no longer a good measure.”

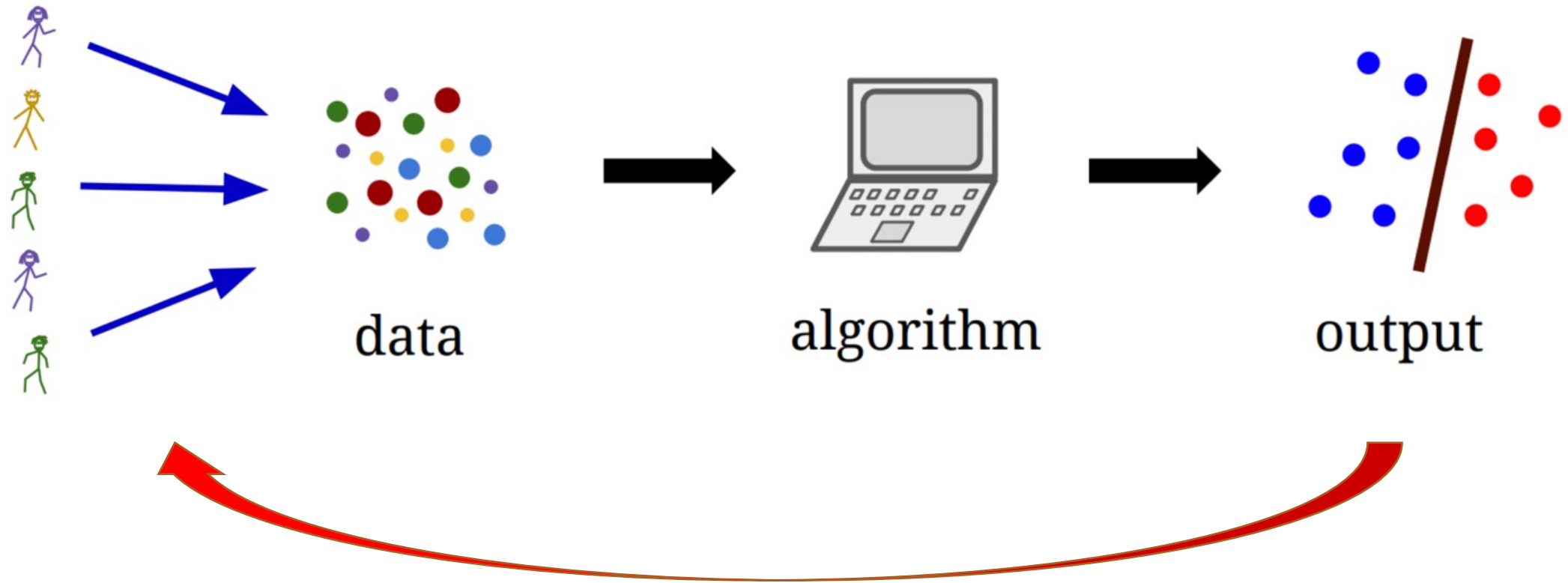
COLLEGE ADMISSIONS



Methodology



Strategic Classification



Machine Learning Lifecycle

