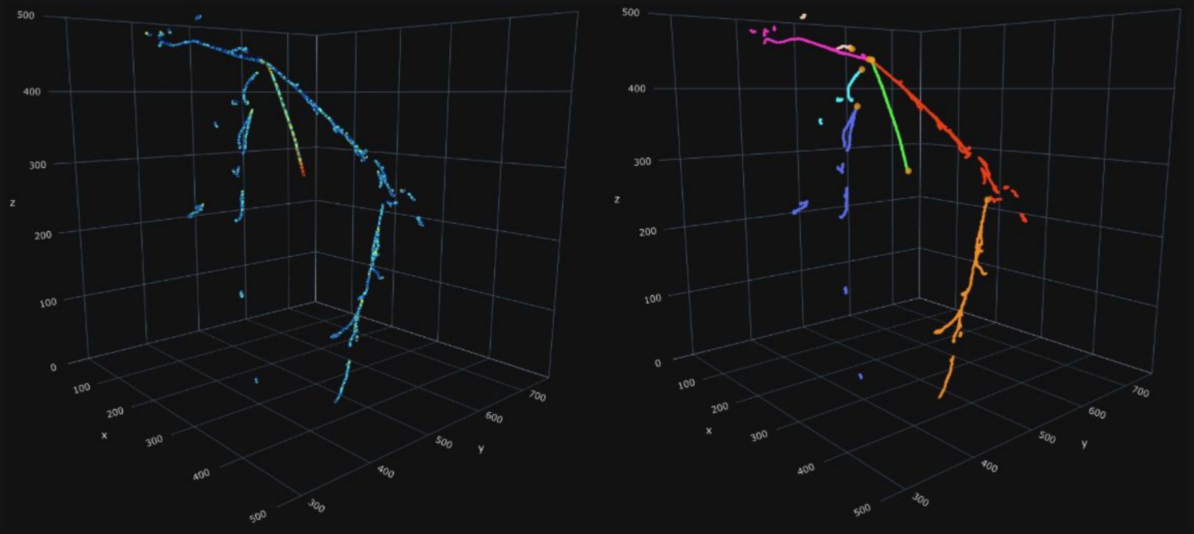
Machine Learning Basics
For SBN/2x2 ML Workshop

Original image credit: xkcd



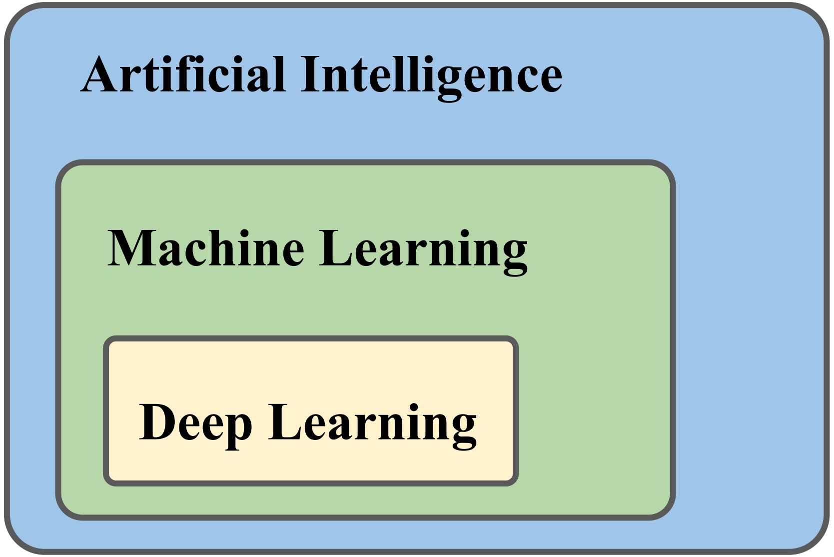
Kazuhiro Terao
SLAC National Accelerator Laboratory

Machine Learning

Machine Learning, Deep Learning, AI … what are they?

Machine Learning

Machine Learning, Deep Learning, AI … what are they?



Artificial Intelligence

- A computer with intelligence

Machine Learning

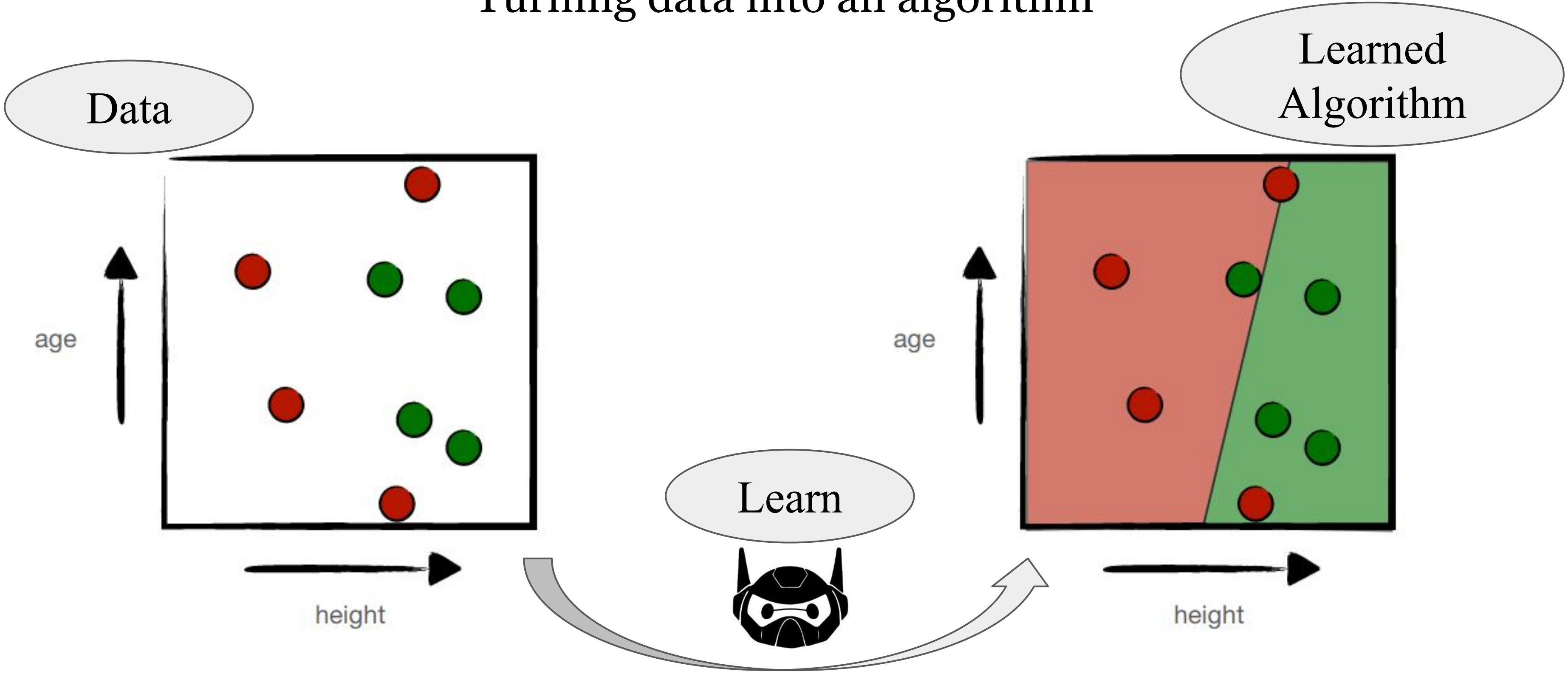
- Process to generate an intelligent algorithm from data.

Deep Learning

- ML methods that aim at complex pipelines working on low-level data

Machine Learning

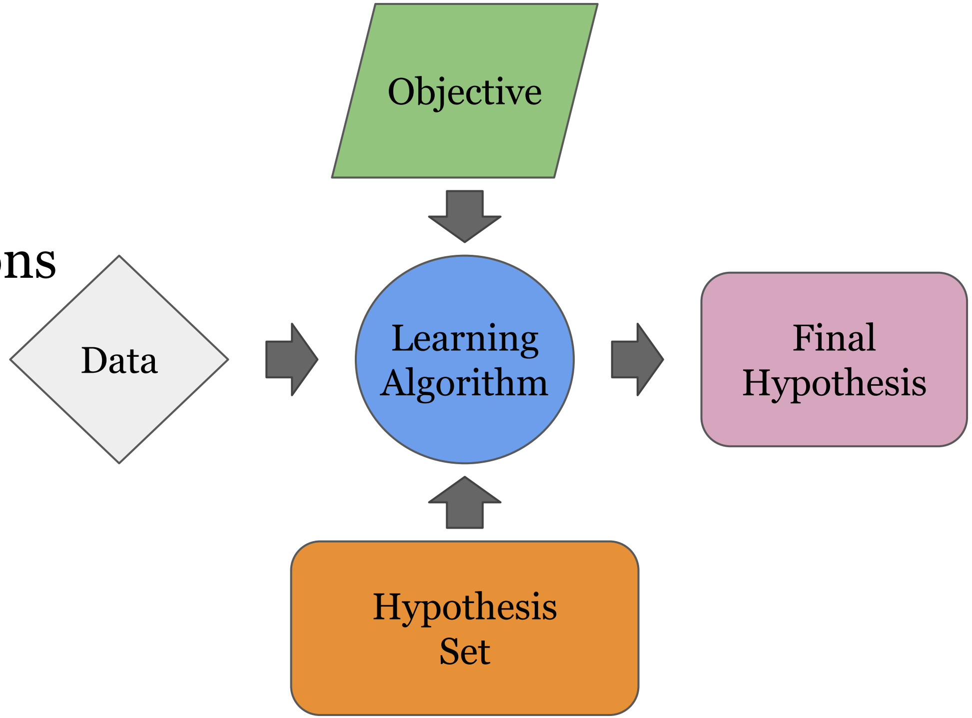
Turning data into an algorithm



Learning Framework

How to machine learn?

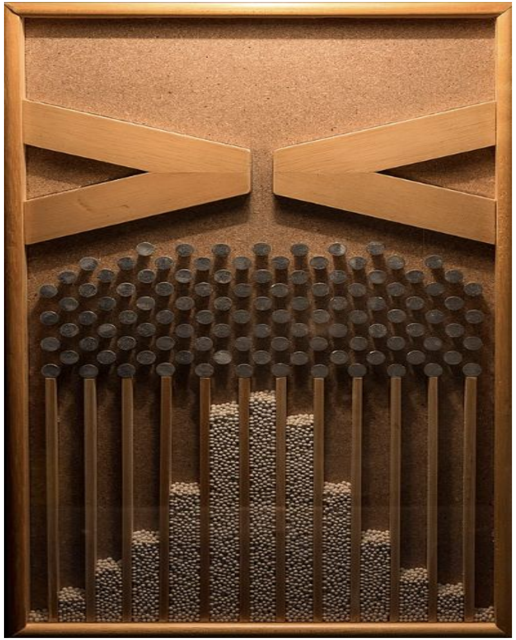
1. Prepare **data set**.
2. Define a set of potential solutions **(hypothesis models)**.
3. Define the performance metric (**learning objectives**).
4. Optimize using a **learning algorithm**.



Data

In machine/statistical learning, we assume data is **independently** sampled from **identical distribution**. Sometimes acronymed “**i.i.d**”.

- **Assumption**: present and future data follow the same distribution.
  - The algorithms optimized using the existing data can be used to “predict” or “infer” things about the future data.
  - Inherent weakness:
    - **Out-of-distribution**
    - **Distributional shift**



Hypothesis Set

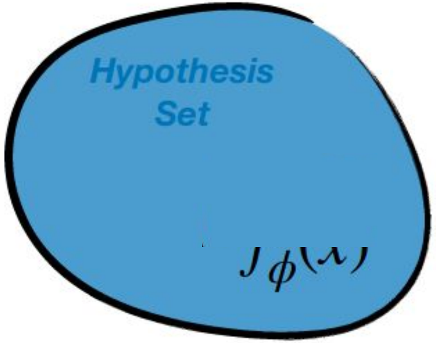
Algorithm = a numeric program with input and output

$$f(x) : \mathcal{X} \rightarrow \mathcal{Y}$$

Popular choice to form a “set of candidate algorithm”: **parametrization**

$$f_{\boxed{\phi}}(x) : \mathcal{X} \rightarrow \mathcal{Y}$$

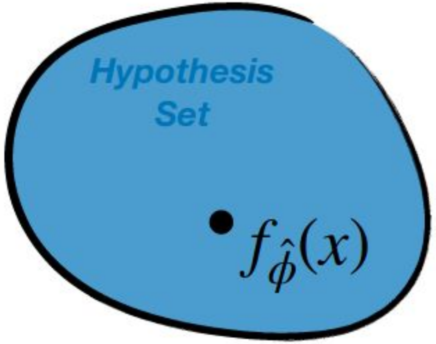
parameters



Objective

Use the objective measure in order to choose the best hypothesis within the set. Use the objective to guide the learning process. Typically this is to minimize the error metric called “loss” or “risk”.

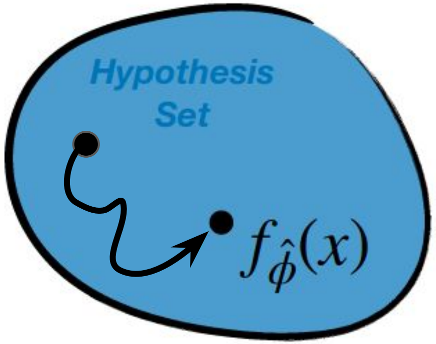
Objective = find $\hat{\phi} = \mathrm{argmax}_{\phi}\mathrm{Perf}$



Learning Algorithm

Learning = choosing the best hypothesis within the set. Use the objective to guide the learning process.

- Analytical solution (rare)
- Grid = discrete search (lots of compute)
- Iterative update (most typical)



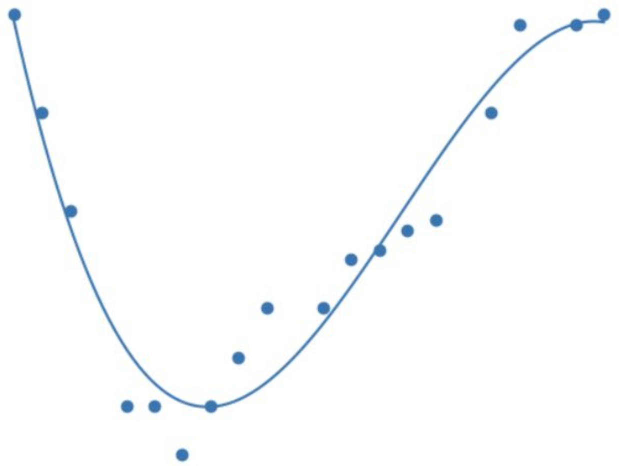
Example: multivariate linear regression

Data: $\{y_i\}$ sampled from true underlying distribution

Hypothesis set: $\sum_{j}^{m} w_j x_{ji}$

Objective: $\sum_{i}^{n} \left(y_i - \sum_{j}^{m} w_j x_{ji} \right)^2$

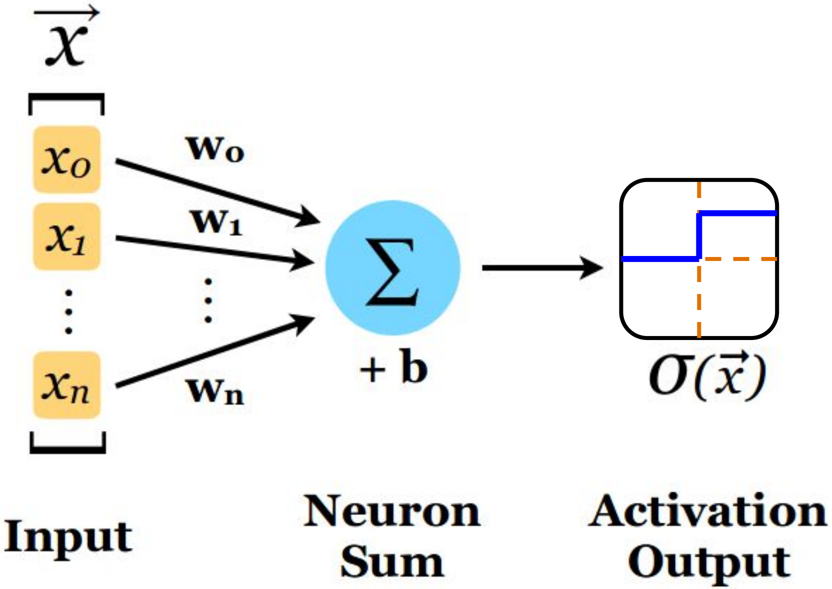
Learning: $\mathbf{w} = \left(\mathbf{X}^T \mathbf{X} \right)^{-1} \mathbf{X}^T \mathbf{y}$



Example: neural network

The basic unit of a neural net is the ***perceptron*** (loosely based on a real neuron)

Takes in a vector of inputs (x). Commonly inputs are summed with weights (w) and offset (b) then run through activation.

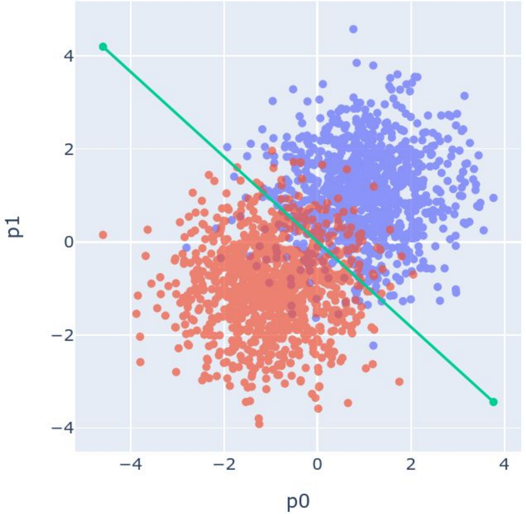


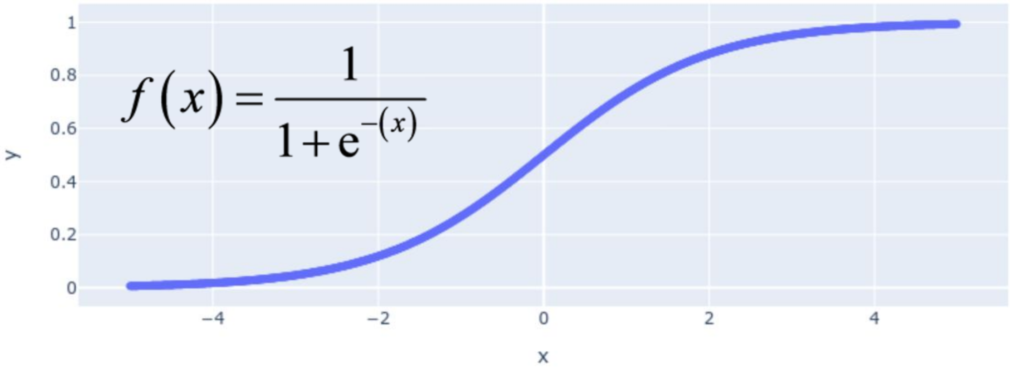
$$\sigma(\vec{x}) = \begin{cases} 1 & \vec{w}_i \cdot \vec{x} + b_i \geq 0 \\ 0 & \vec{w}_i \cdot \vec{x} + b_i < 0. \end{cases}$$

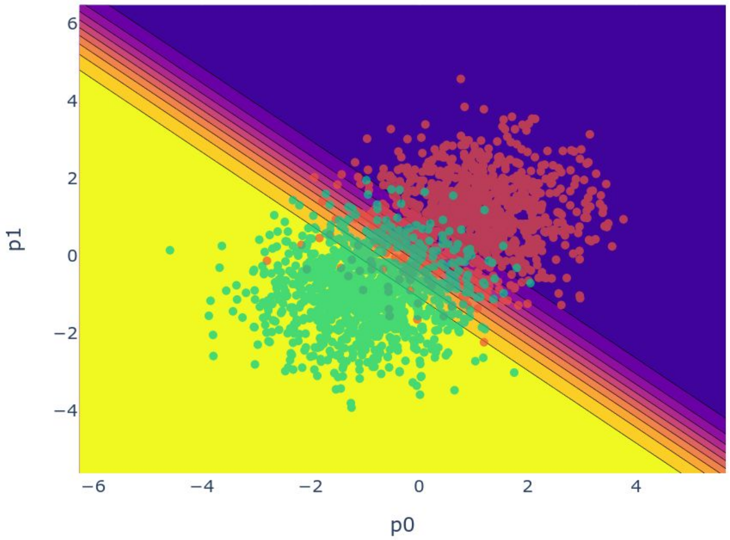
Example: neural network

Perceptron is a linear model and can be visualized as a linear line (in n-dimension).

One can use the “sigmoid” function as the activation function. In Statistics, this is known as the “logistic regression” (or linear binary classification)

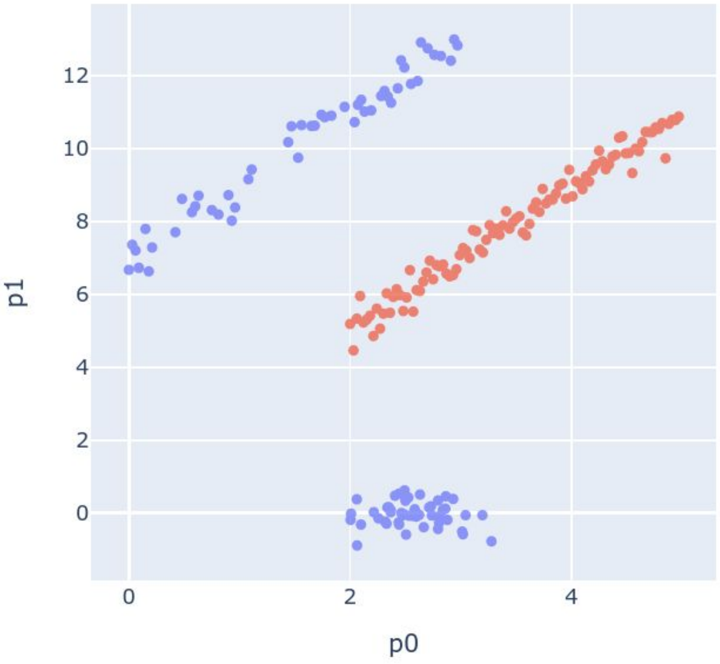






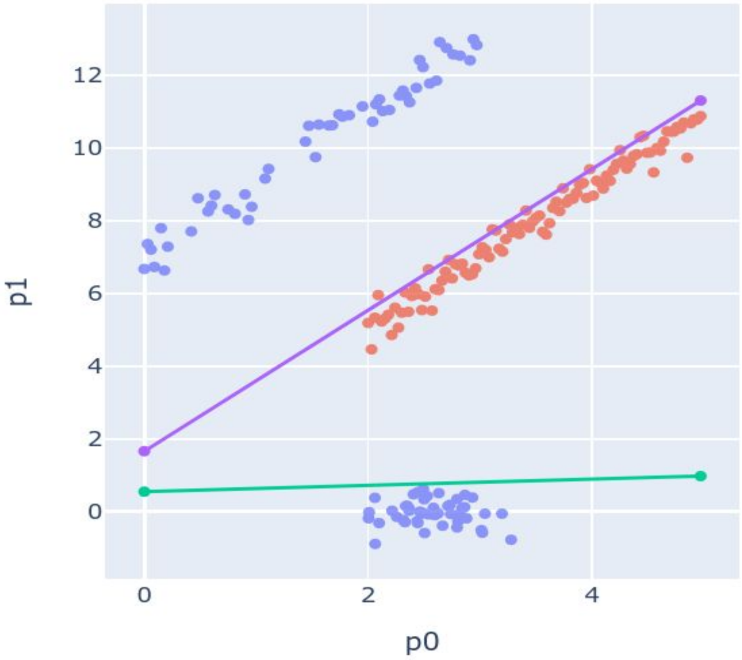
Example: neural network

What if the data is not linearly separable?

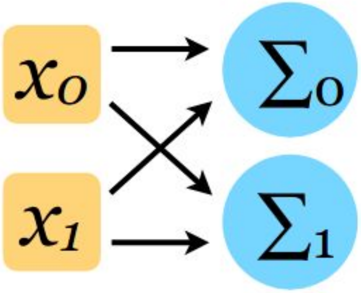


Example: neural network

What if the data is not linearly separable?

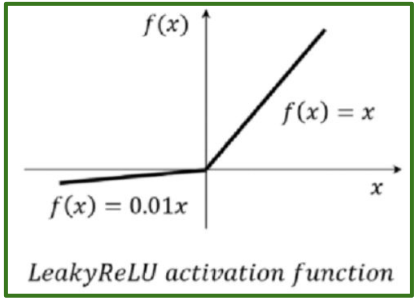


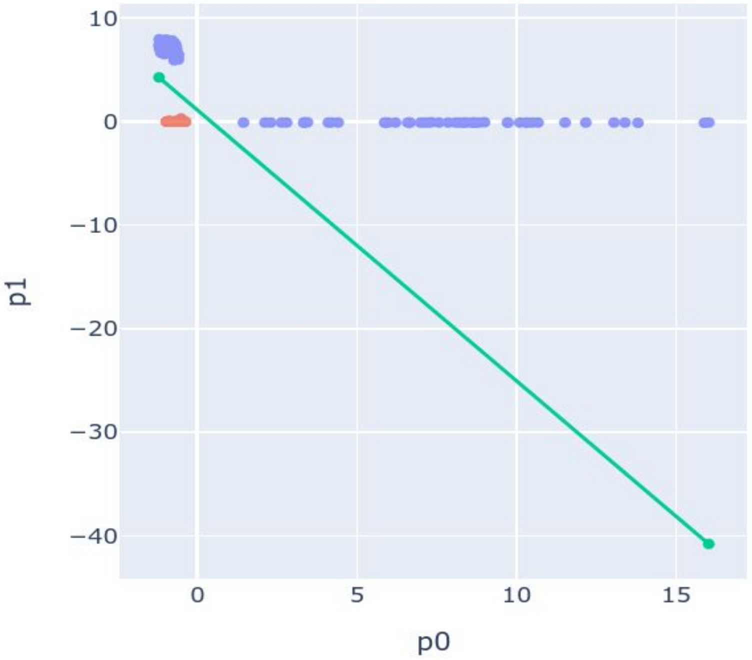
- Add a neuron



Example: neural network

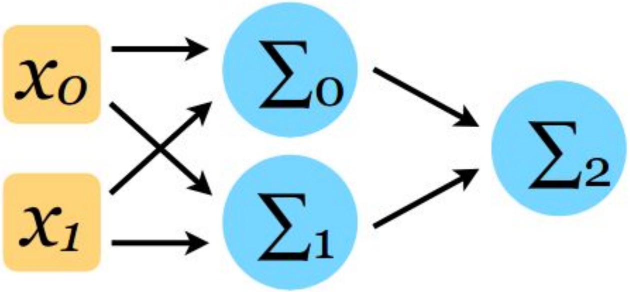
What if the data is not linearly separable?





- Add a neuron
- Add a layer

The task became linearized in the new feature space



Multi-layer Perceptron (MLP)

More on Learning Algorithms

Learning Algorithm: Gradient Descent (GD)

Goal: tune the parameters **w** and achieve $\nabla_{\mathbf{w}}\mathcal{L} = 0$

Minimize the loss iteratively: $\boxed{\mathbf{w}_{i+1} = \mathbf{w}_i - \lambda\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}, \mathbf{x})}$

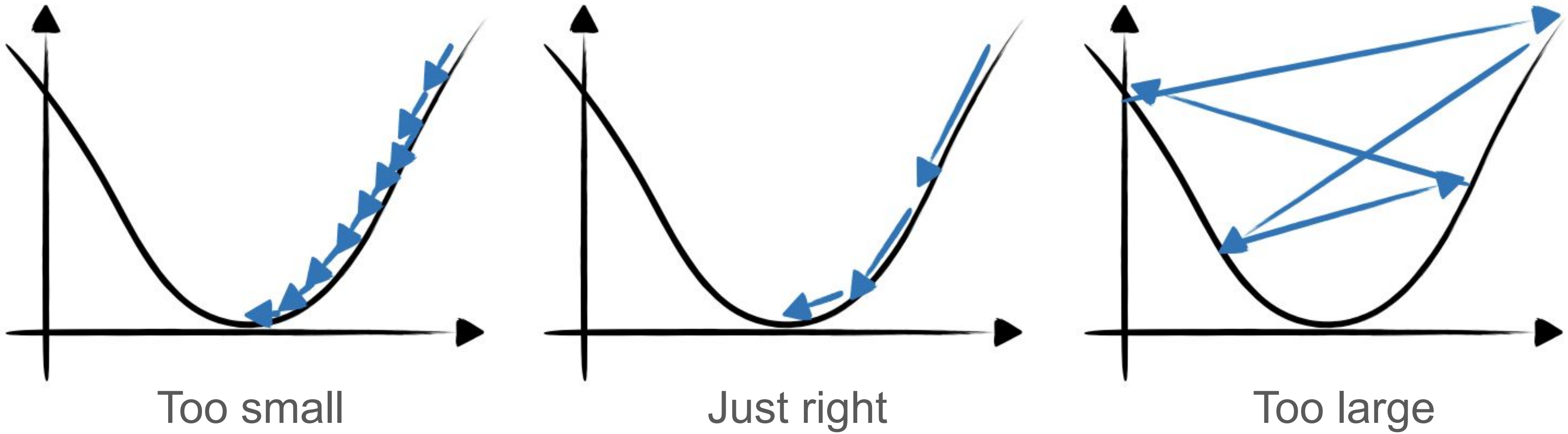
called **Gradient Descent** (GD) where λ controls the rate of learning process.

Learning Algorithm: Gradient Descent (GD)

Goal: tune the parameters **w** and achieve $\nabla_{\mathbf{w}}\mathcal{L} = 0$

Minimize the loss iteratively: $\boxed{\mathbf{w}_{i+1} = \mathbf{w}_i - \lambda\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}, \mathbf{x})}$

called **Gradient Descent** (GD) where λ controls the rate of learning process.



Learning Algorithm: Stochastic GD

Goal: tune the parameters **w** and achieve $\nabla_{\mathbf{w}} \mathcal{L} = 0$ ... **faster**?

Mini-batch Stochastic GD (SGD) use a **subset of data** for gradient.

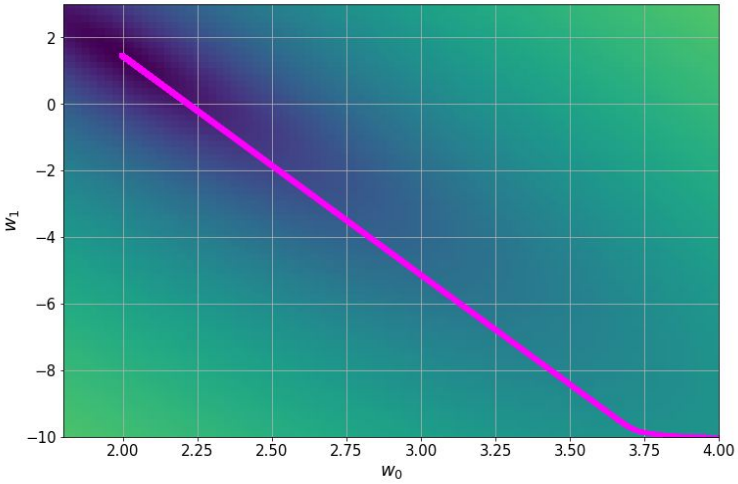
Learning Algorithm: Stochastic GD

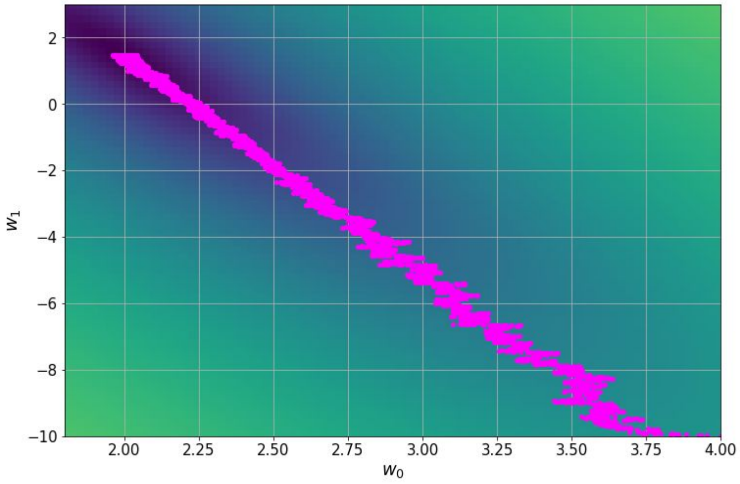
Goal: tune the parameters **w** and achieve $\nabla_{\mathbf{w}}\mathcal{L} = 0$ ... **faster**?

Mini-batch Stochastic GD (SGD) use a **subset of data** for gradient.

1. Create a batch = random subset of data.
2. Compute the gradient for the batch and update the parameters.

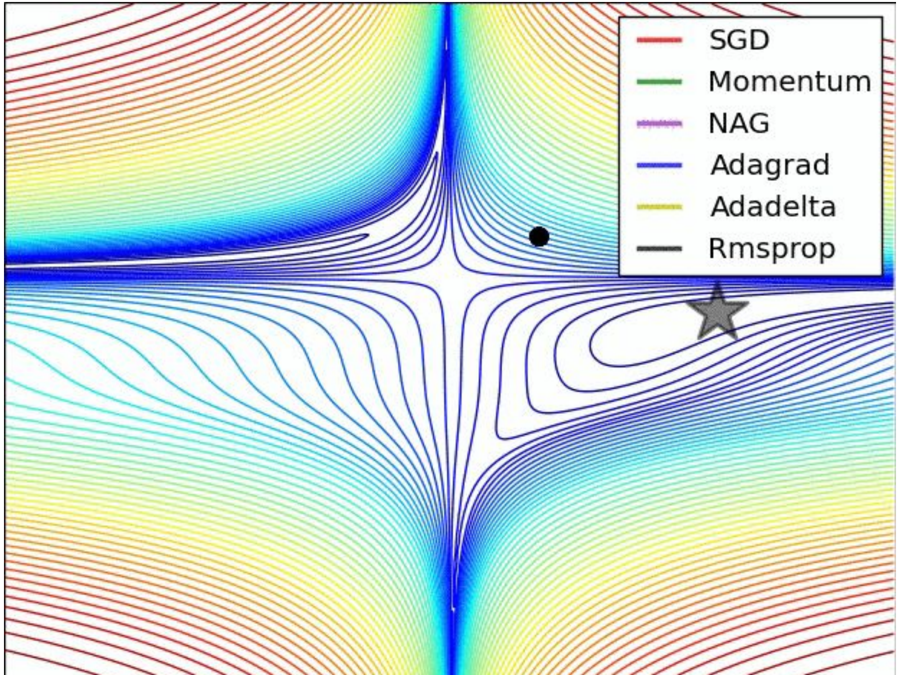
Note: when data is always new (never seen before), called "online learning"

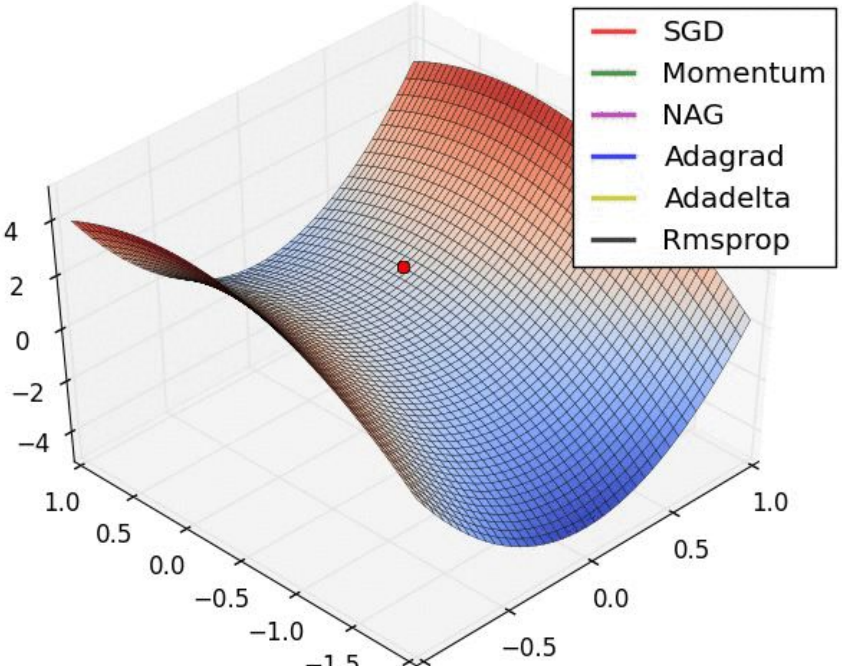




Learning Algorithm: beyond vanilla SGD

Adaptive LR sets dynamic value based on the gradient magnitude, **Momentum** learns from the history to avoid being trapped by oscillating loss values, and much more research is done… a popular default choice is “**Adam**” optimizer.

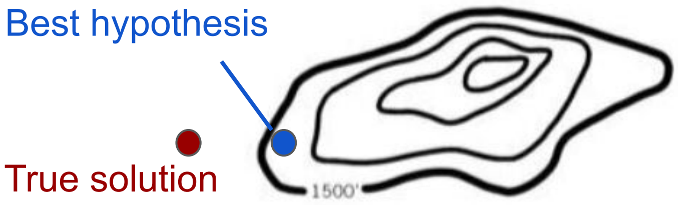




Empirical Risk v.s. Bias

Expressivity of a Hypothesis Set

High expressivity means that the hypothesis set can approximate many functions and more likely to contain a good representation of the true solution.



Model bias = loss @ best hypothesis

A sufficiently large neural network can become a "Universal Approximation Function" (i.e. can model any function), which makes neural network an interesting/popular model choice in ML.

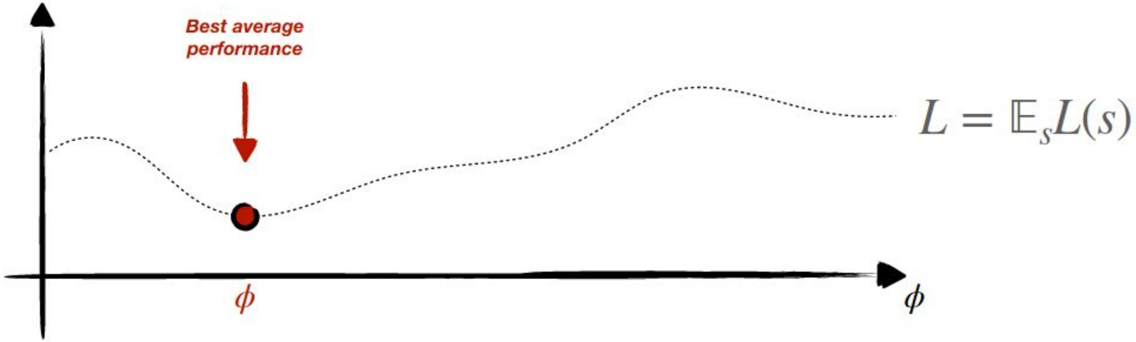
Q: is there any con about a large model?

Statistical Learning

Assume: data is a stochastic sample. $s \sim p(s)$

In order to learn, we must estimate the performance or minimize the "risk" or "loss." The true expectation loss is:

$$\bar{L}_\phi = E_{s\sim p(s)}\, L_\phi(s) = \int \mathrm{d}s\; L(s)p(s)$$



Statistical Learning

Assume: data is a stochastic sample. $s \sim p(s)$

In order to learn, we must estimate the performance or minimize the "risk" or "loss." The true expectation loss is:

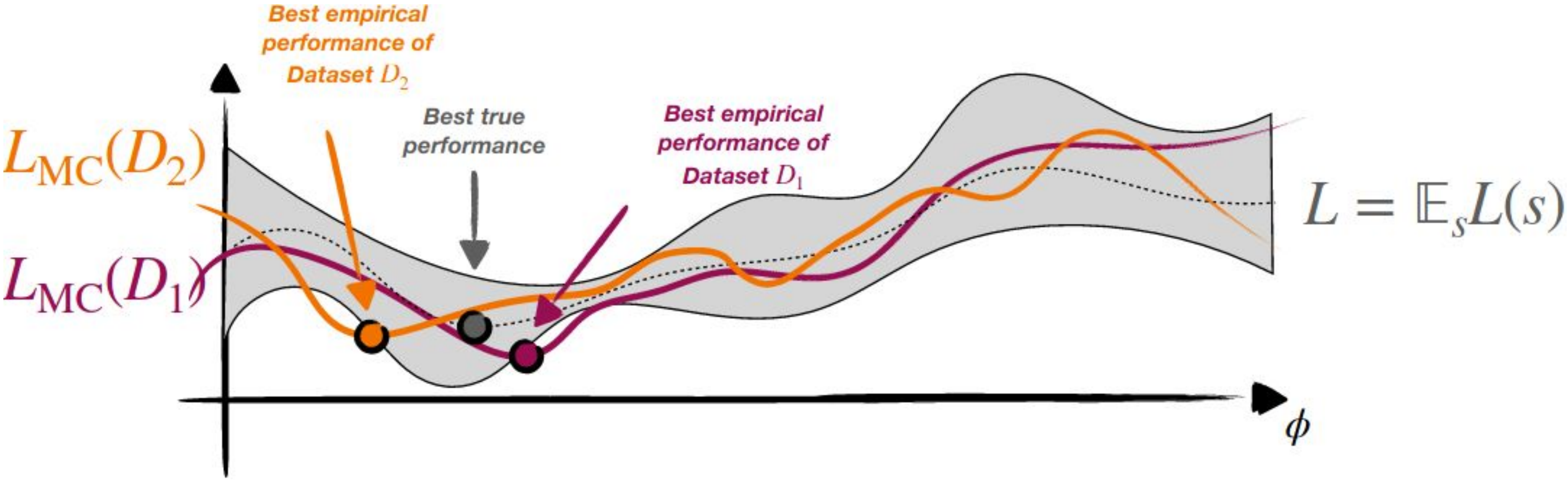
$$\bar{L}_\phi = E_{s\sim p(s)}\, L_\phi(s) = \int \mathrm{d}s\, L(s)p(s)$$

However, we only have data set (sample "s") and have no access to p(s). We can have an unbiased **approximation**, the **empirical loss**

$$\bar{L} \approx \underbrace{L_{\mathrm{MC}}}_{\text{"Monte Carlo" estimate}}(D) = \frac{1}{N}\sum_i L(s_i)$$

Empirical Risk Minimization

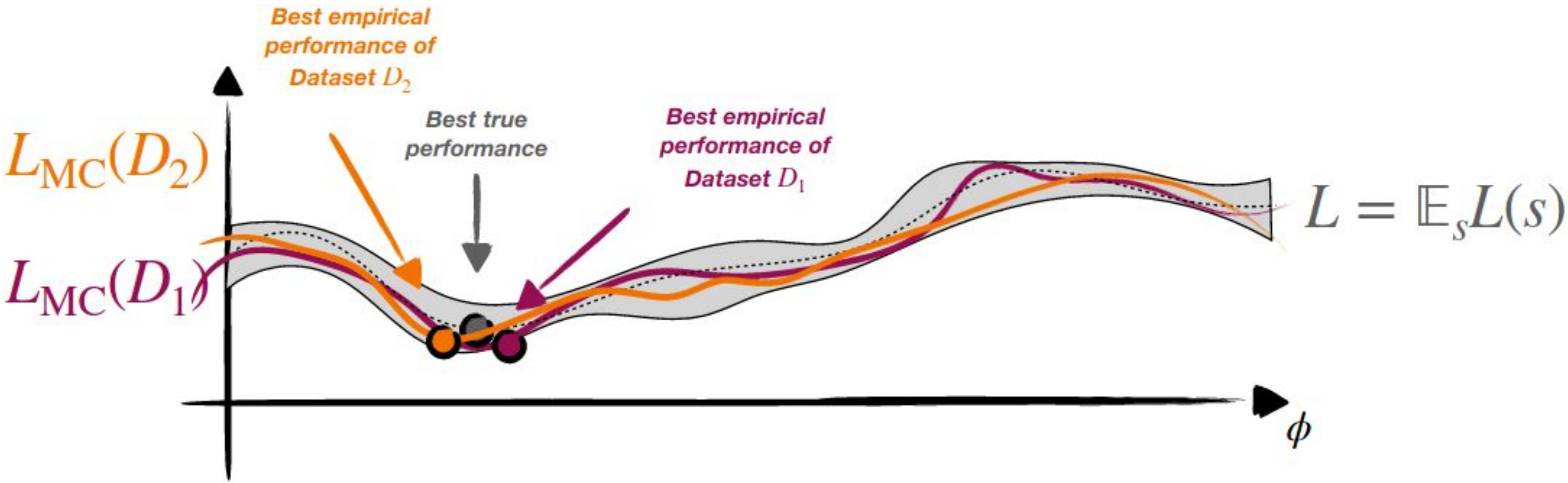
Choose a hypothesis from the set such that it performs best for the given dataset. The data size is critical.



Small data

Empirical Risk Minimization

Choose a hypothesis from the set such that it performs best for the given dataset. The data size is critical.



Big data

Bias-Variance trade off

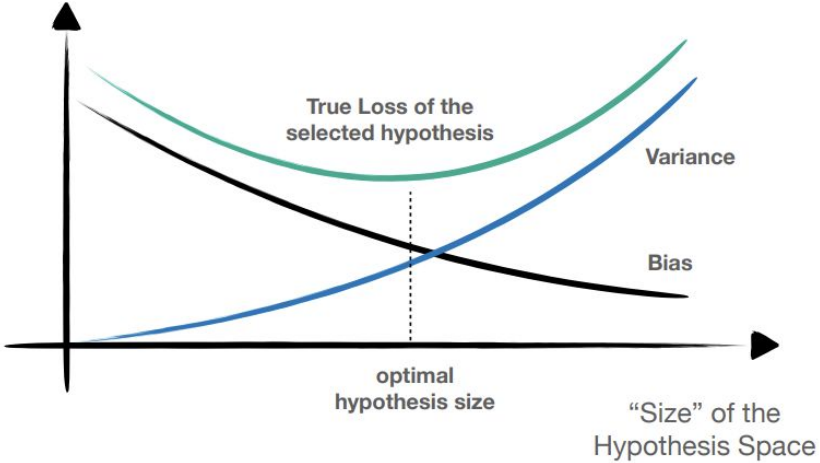
The dataset size is finite and stochastically sampled. Within the same hypothesis set, the "best solution" ***h**** depends on a specific dataset.

Variance = the spread across $\{h^*\}$ among different datasets.

The empirical loss is always worse than the bias.

$$\bar{L}_{h^*} = \bar{L}_{\text{bias}} + \Delta L_{\text{var}}$$

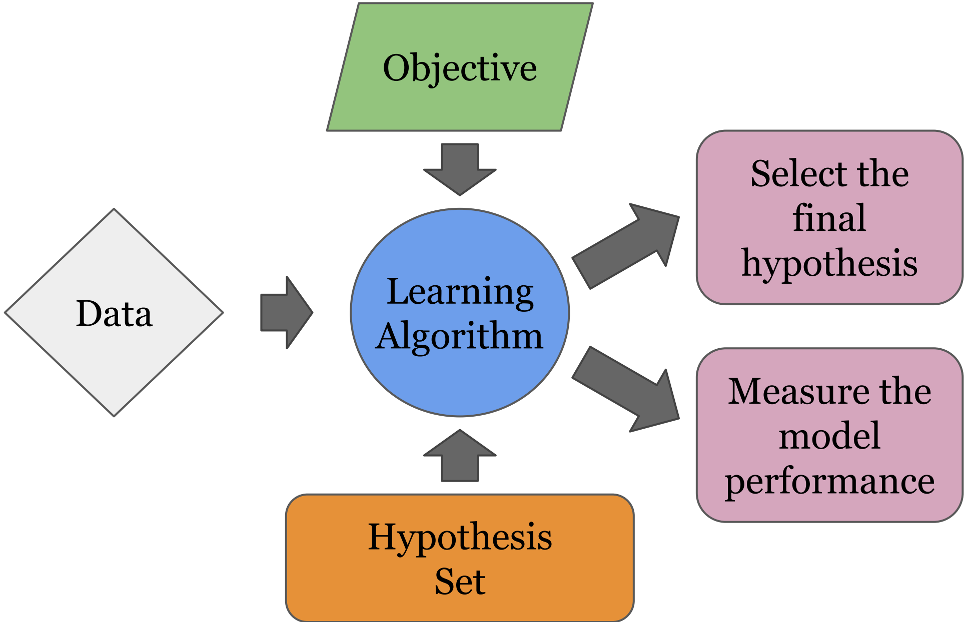
- **Less expressive model = larger bias**
- **More complex model = larger variance**



Choosing the Model w/ Trade-Offs

Recall: we can only measure the empirical risk but …

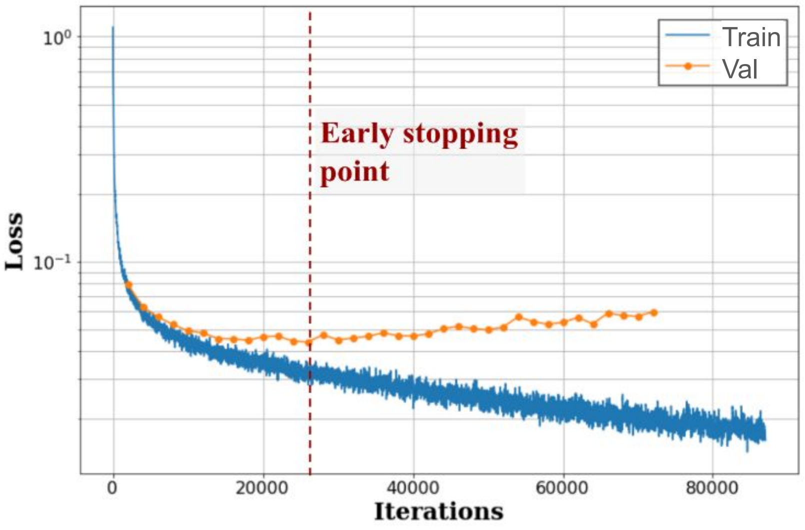
- Want to guess the best hypothesis based on the empirical risk
- Want to measure the model “generalization” performance



Choosing the Model w/ Trade-Offs

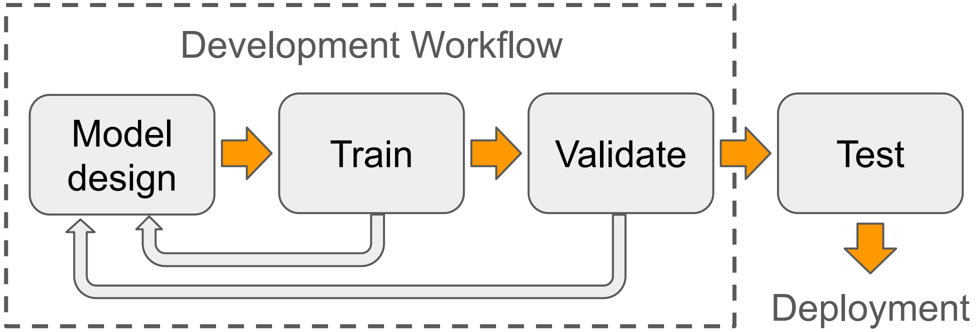
Split the data 3 ways: training, validation, and test datasets

- **Validation set** = use to choose/tune the model
- **Test set** = use to assess the model performance (bias estimate)



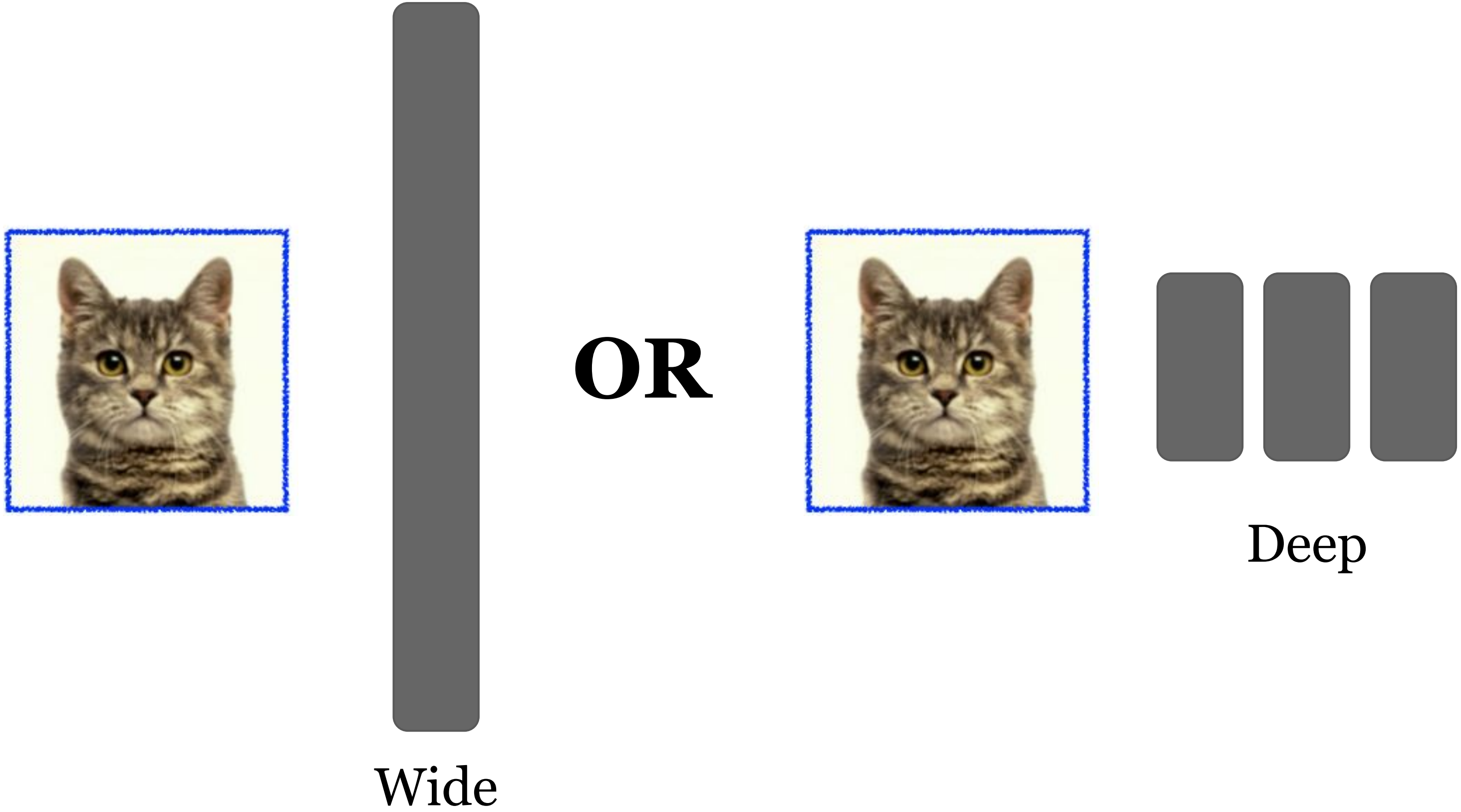
How to choose the final model

- Pick the best performing one (early stopping)
- Modify the hypothesis set (hyper-param. tuning)



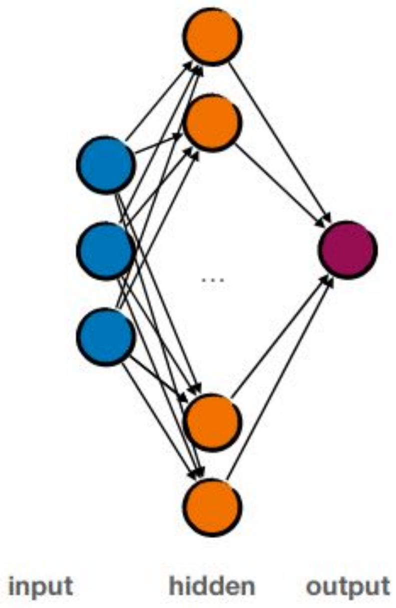
Back to Neural Networks

Neural Network: Architecture Choice



Universal Approximation Theorem

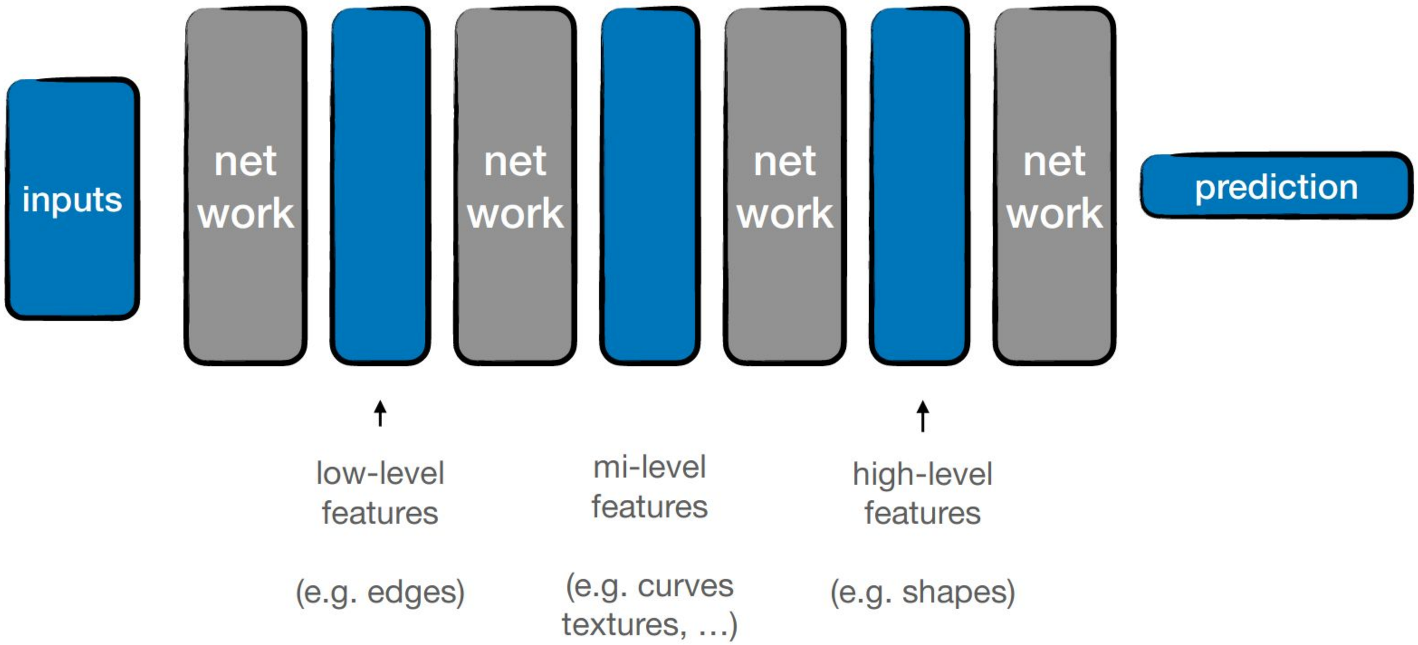
It can be shown that a MLP with single hidden layer is a universal function approximator (can represent any function).



Why do we need a deep network?

Benefits of the depth

A neural network becomes exponentially more expressive with the depth due to composition of features into higher level concepts.

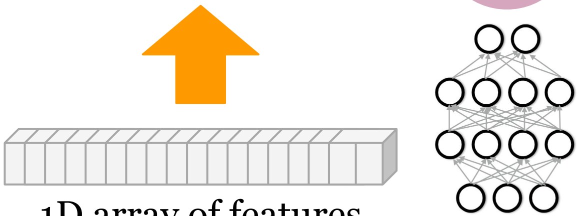


Convolutional Neural Network for Image Data

Next step:

How?

Goal: Dog or Cat ?



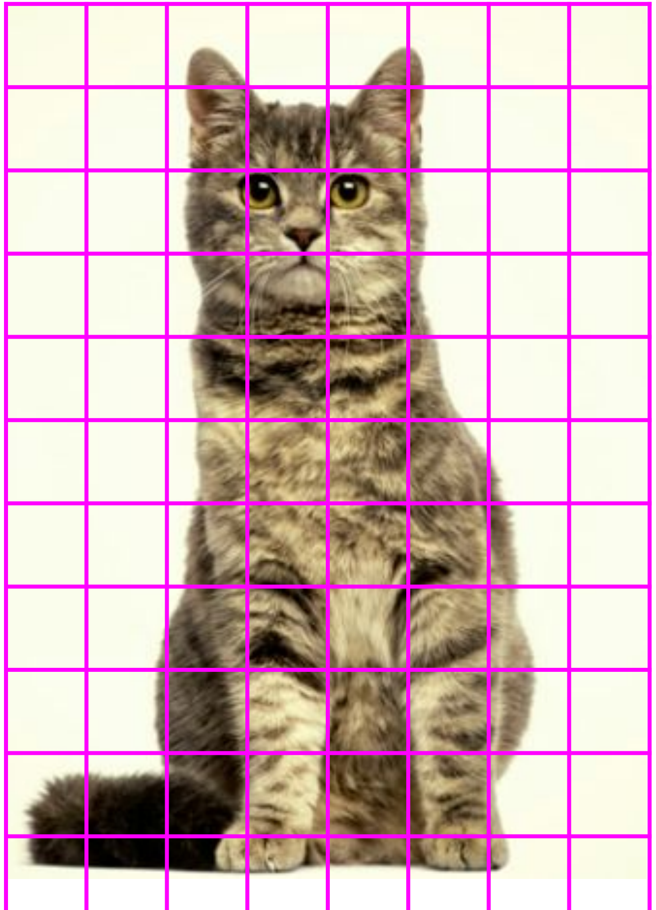
1D array of features

Fully-connected NN
can be useful.

How can we extract
"features" from image?

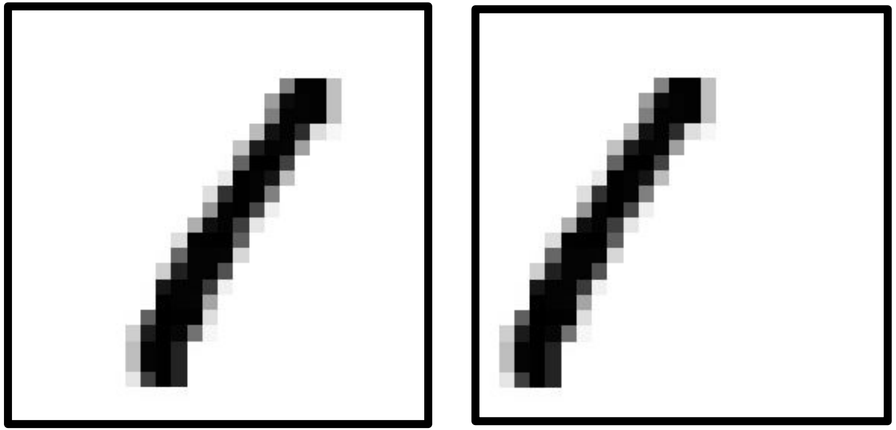
Deep Learning

Next step:

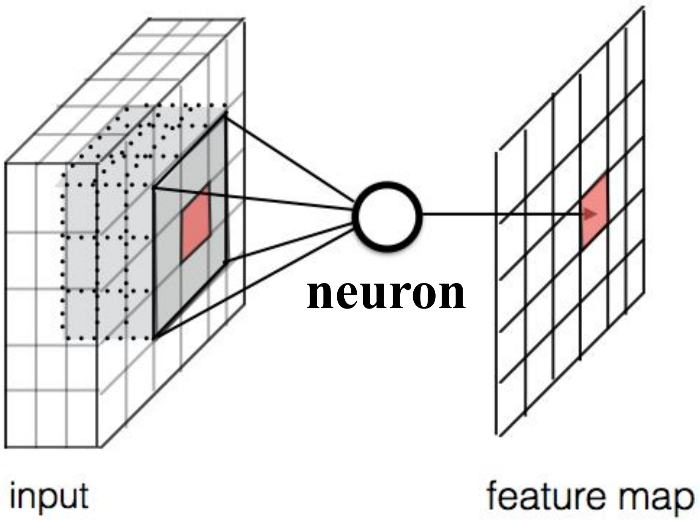


How about flattened image + MLP?

- For an input image of 100x100 pixels RGB image, how many weights does 1 neuron carry? 30,000 for just 1 neuron!
- Two image of the same cat, but in a different position w.r.t. the frame. Would neuron react the same? No! Position information is encoded!



CNNs introduce a ***limitation to MLP*** by forcing a neuron to look at only local, (approx.) translation invariant features

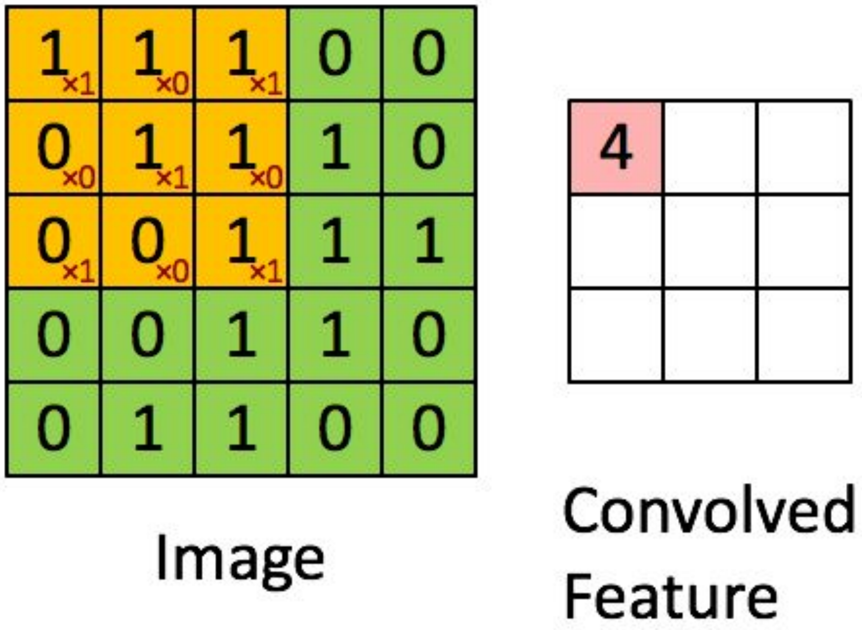


$$f_{i,j}(X) = \sigma\left(W_i \cdot X_j + b_i\right),$$

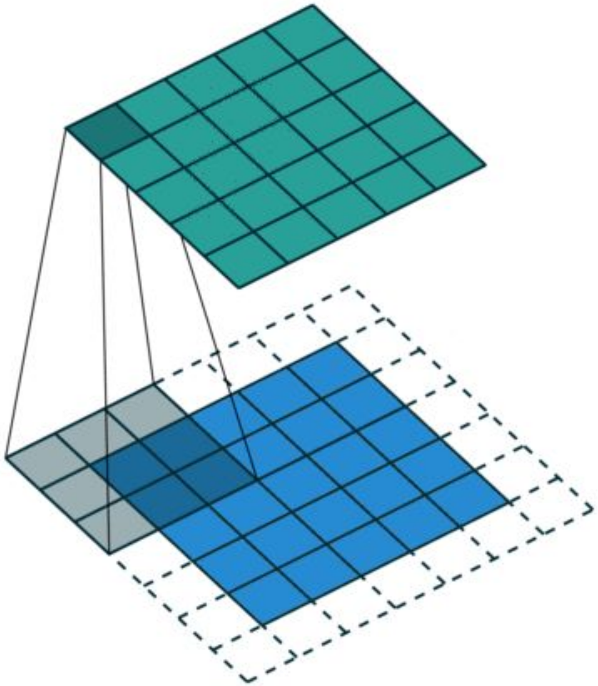
Still a linear transformation!
Weights=matrix, output=scalar
Analyze a fixed-size, local sub-matrix from the input.

- Traverse over 2D space to process the whole input
- **Locality** and **translation-invariance**

Convolution 3x3
Stride 1, no padding

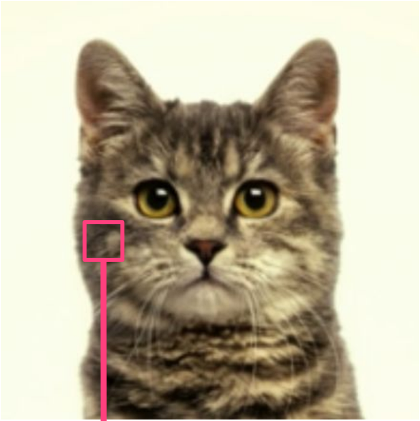


Convolution 3x3
Stride 1, padding 1



Goal: Dog or Cat ?

Goal: Dog or Cat ?



Convolution

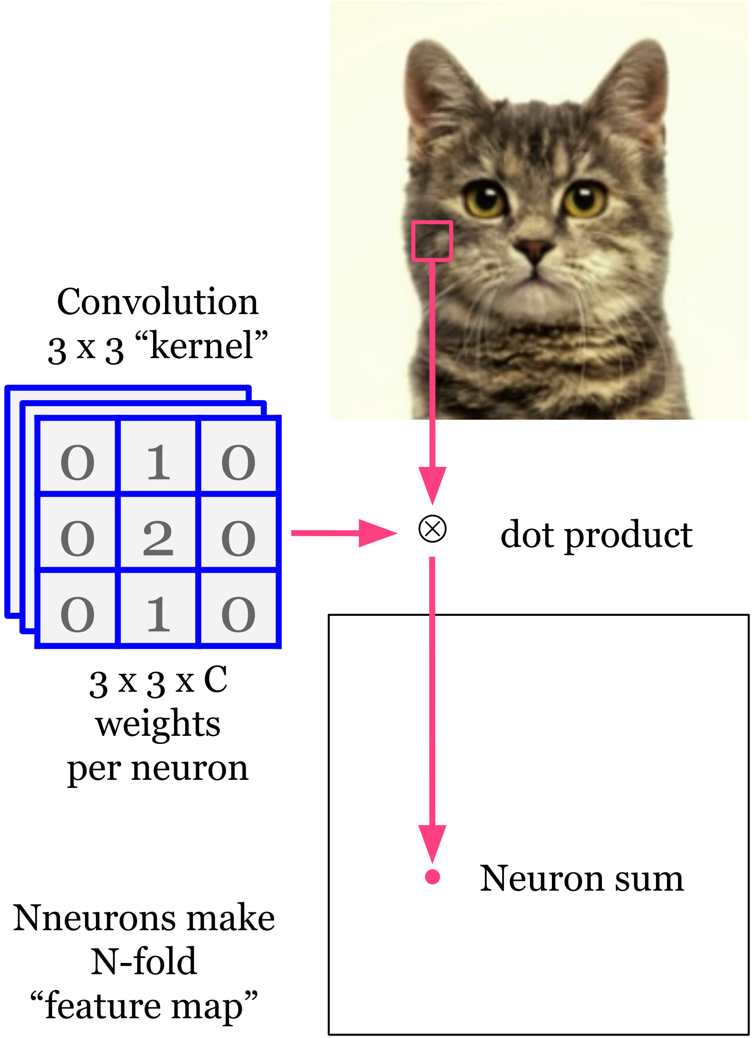
3 x 3 "**kernel**"

0	1	0
0	2	0
0	1	0

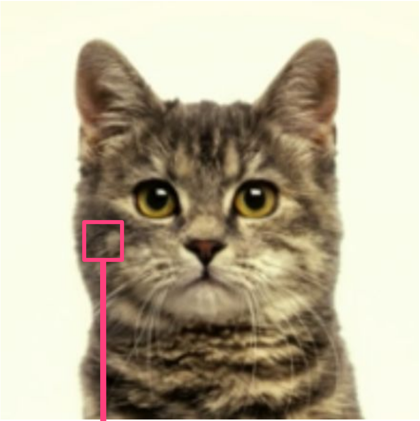
3 x 3 x C
weights
per neuron

$\otimes$ dot product

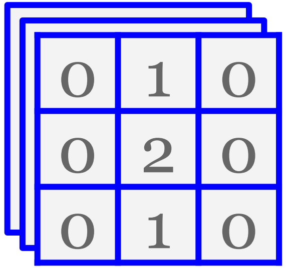
Goal: Dog or Cat?



Goal: Dog or Cat ?

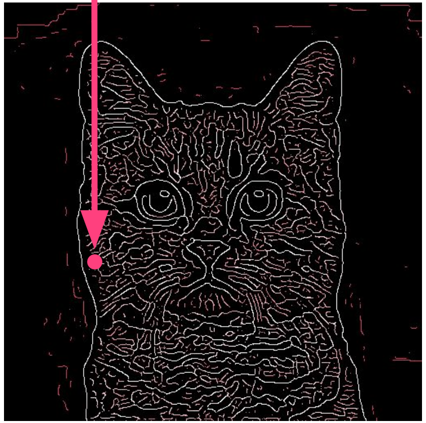


Convolution
3 x 3 “kernel”



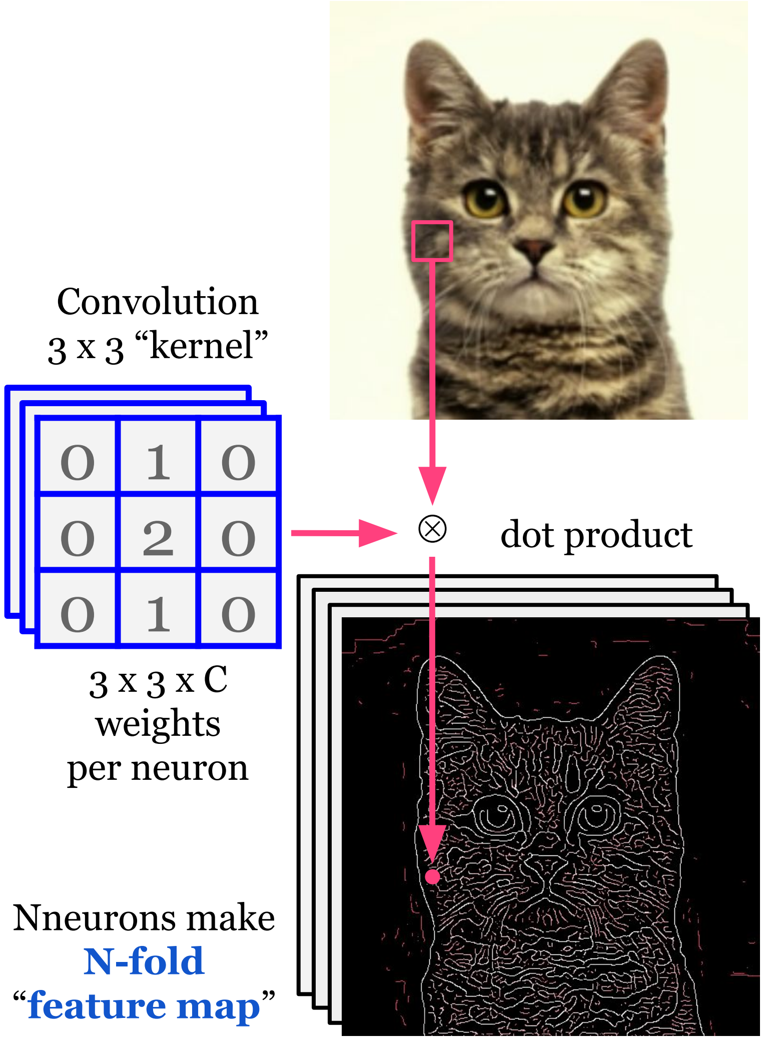
3 x 3 x C
weights
per neuron

⊗ dot product

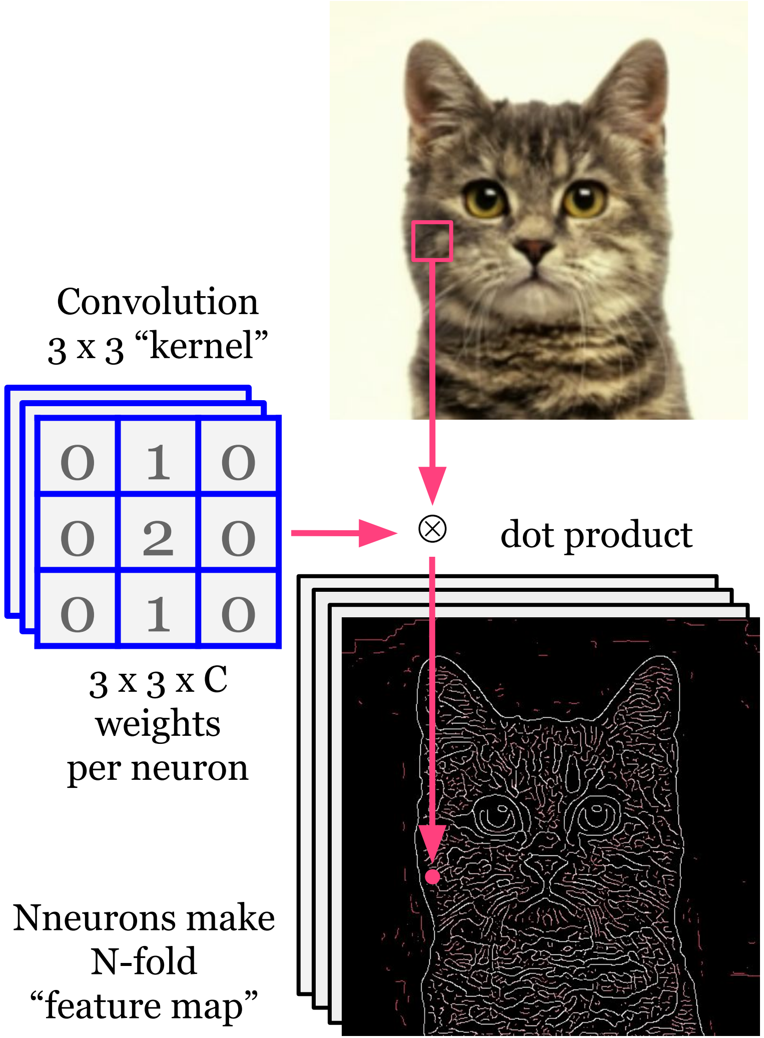


“**feature map**”

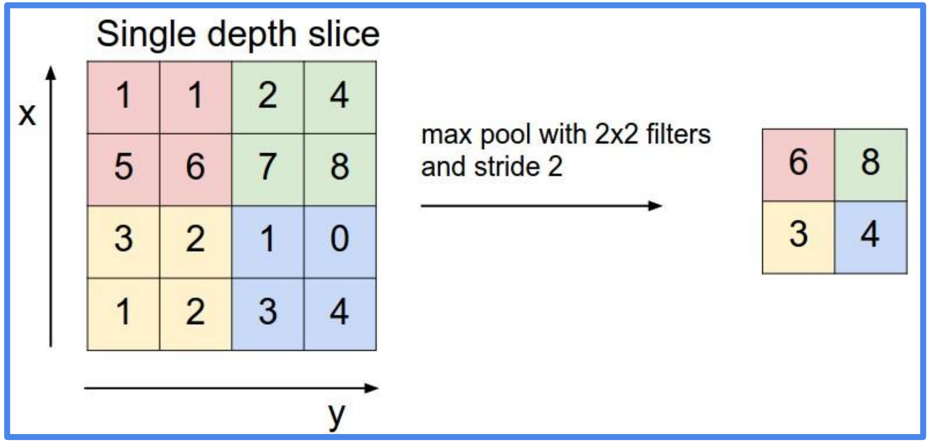
Goal: Dog or Cat ?

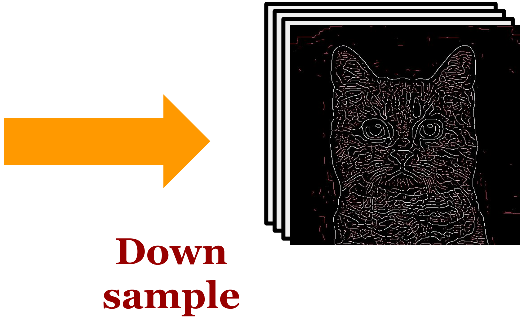


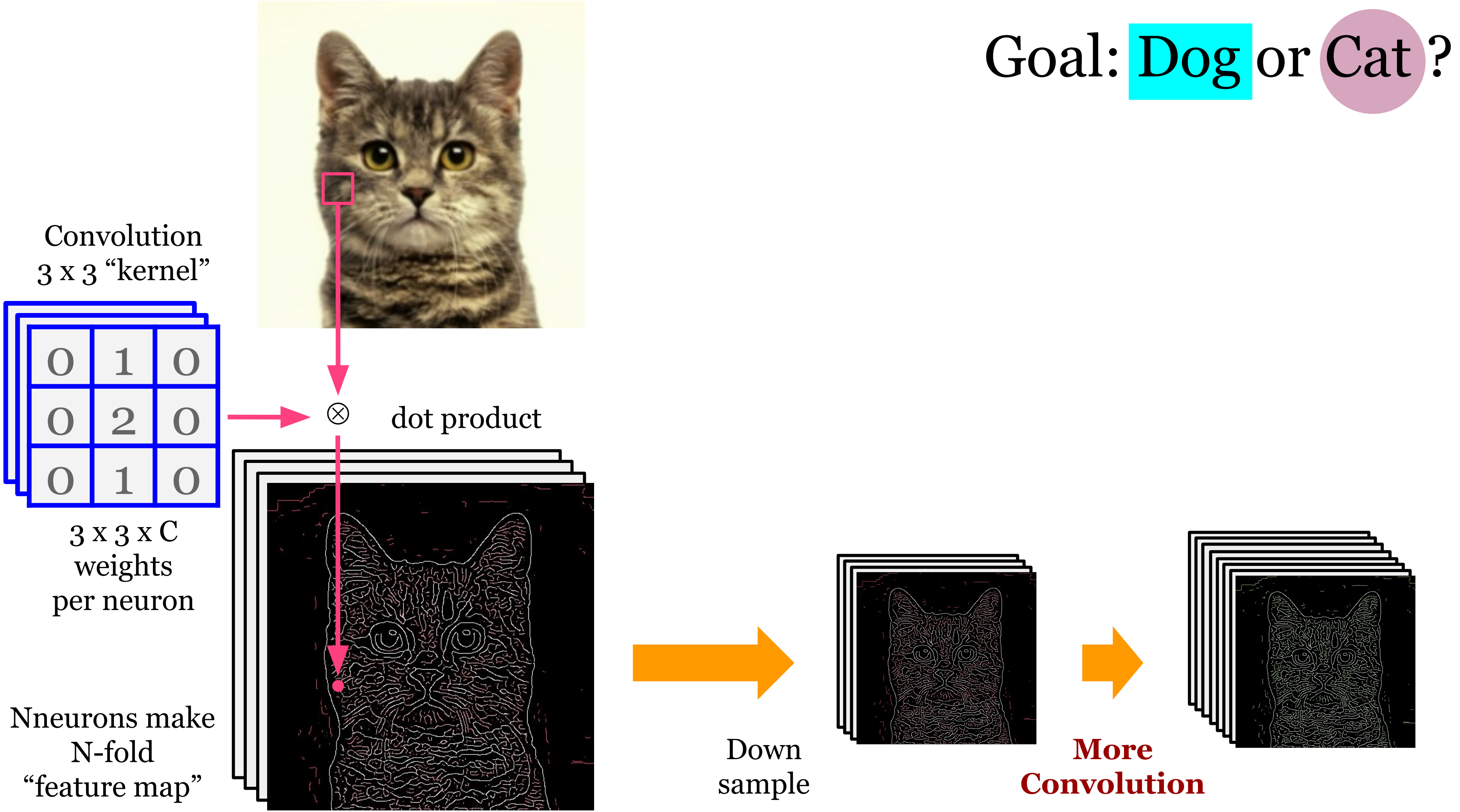
Goal: Dog or Cat?



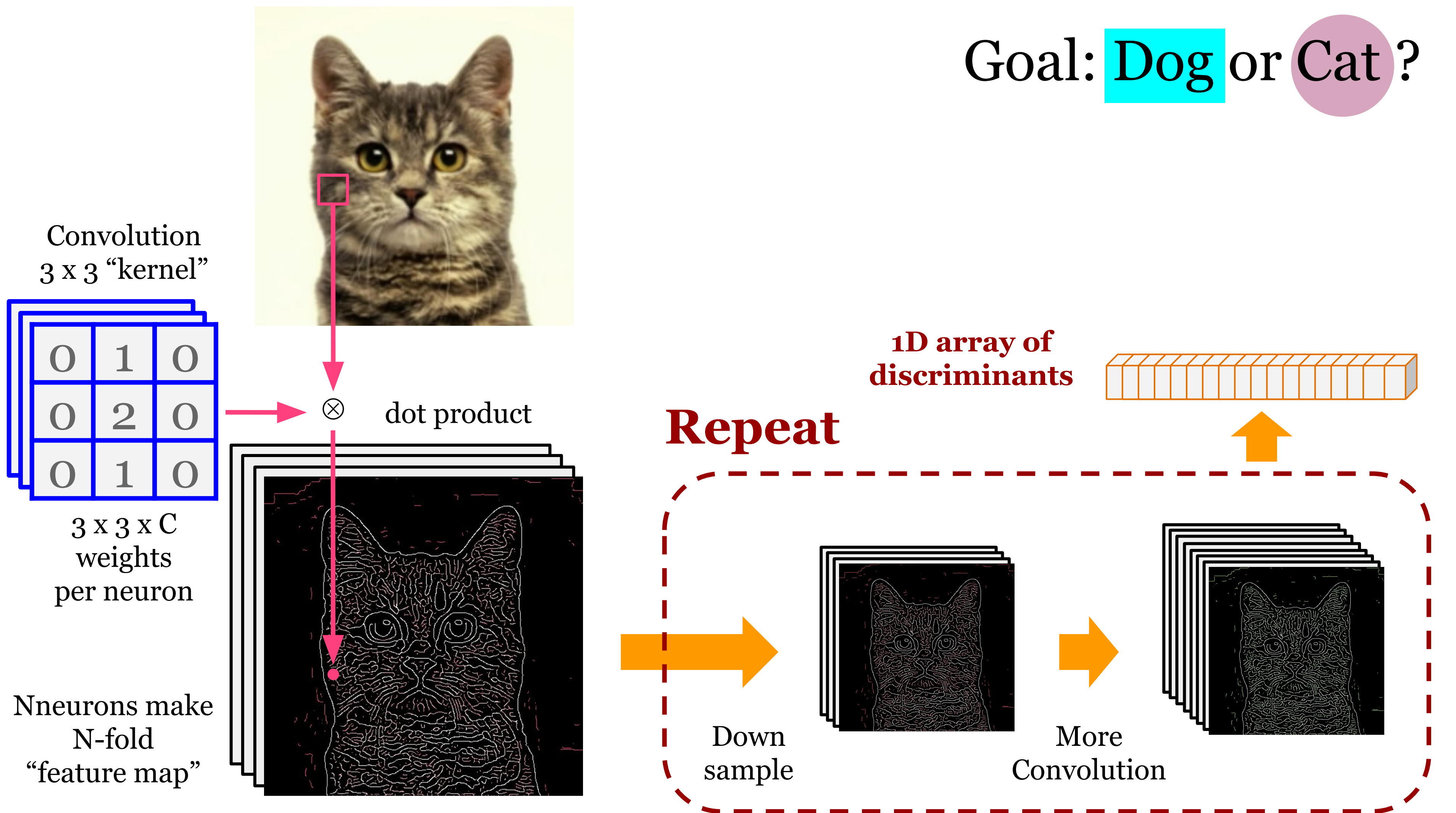
e.g) max pooling

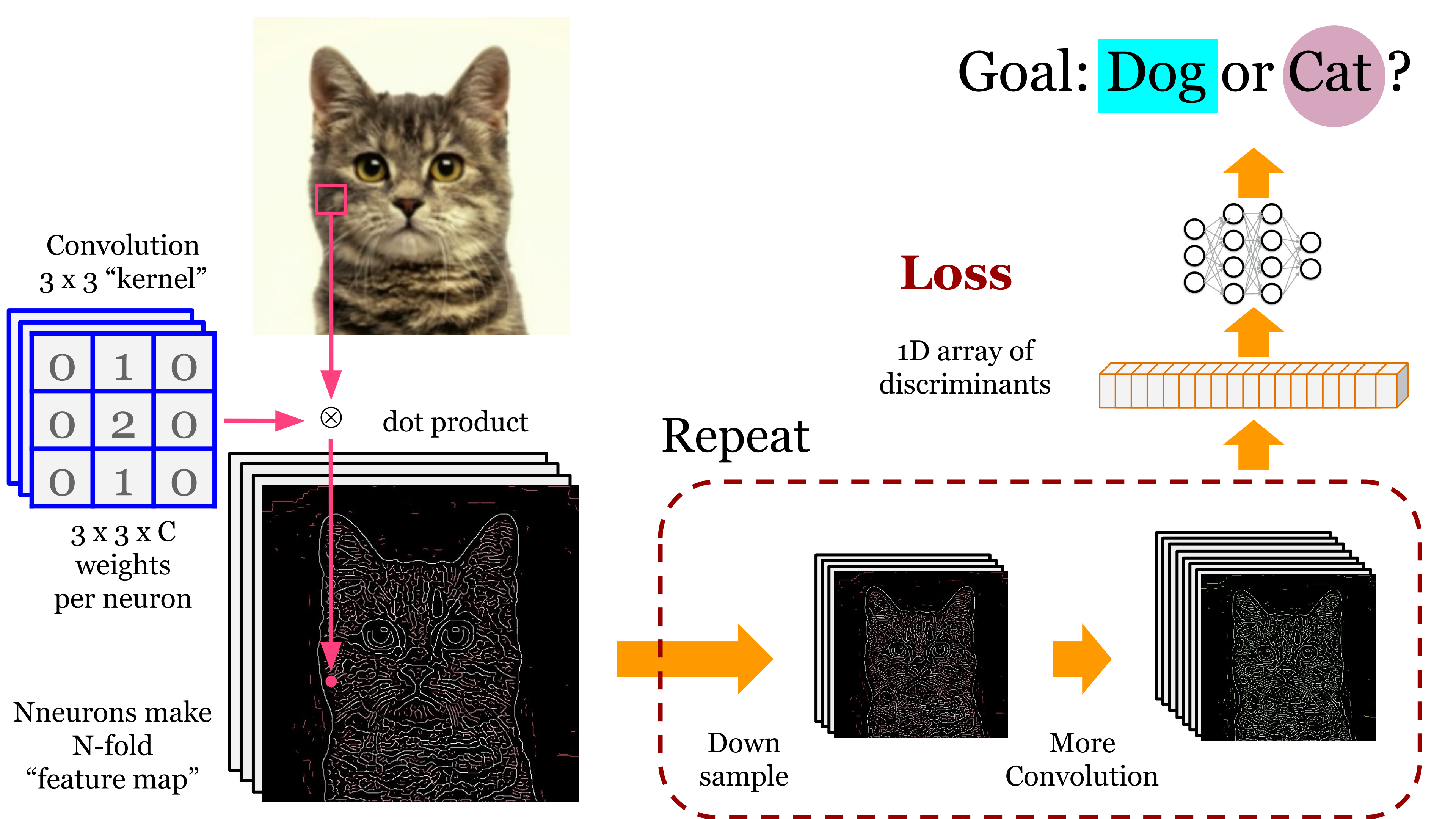


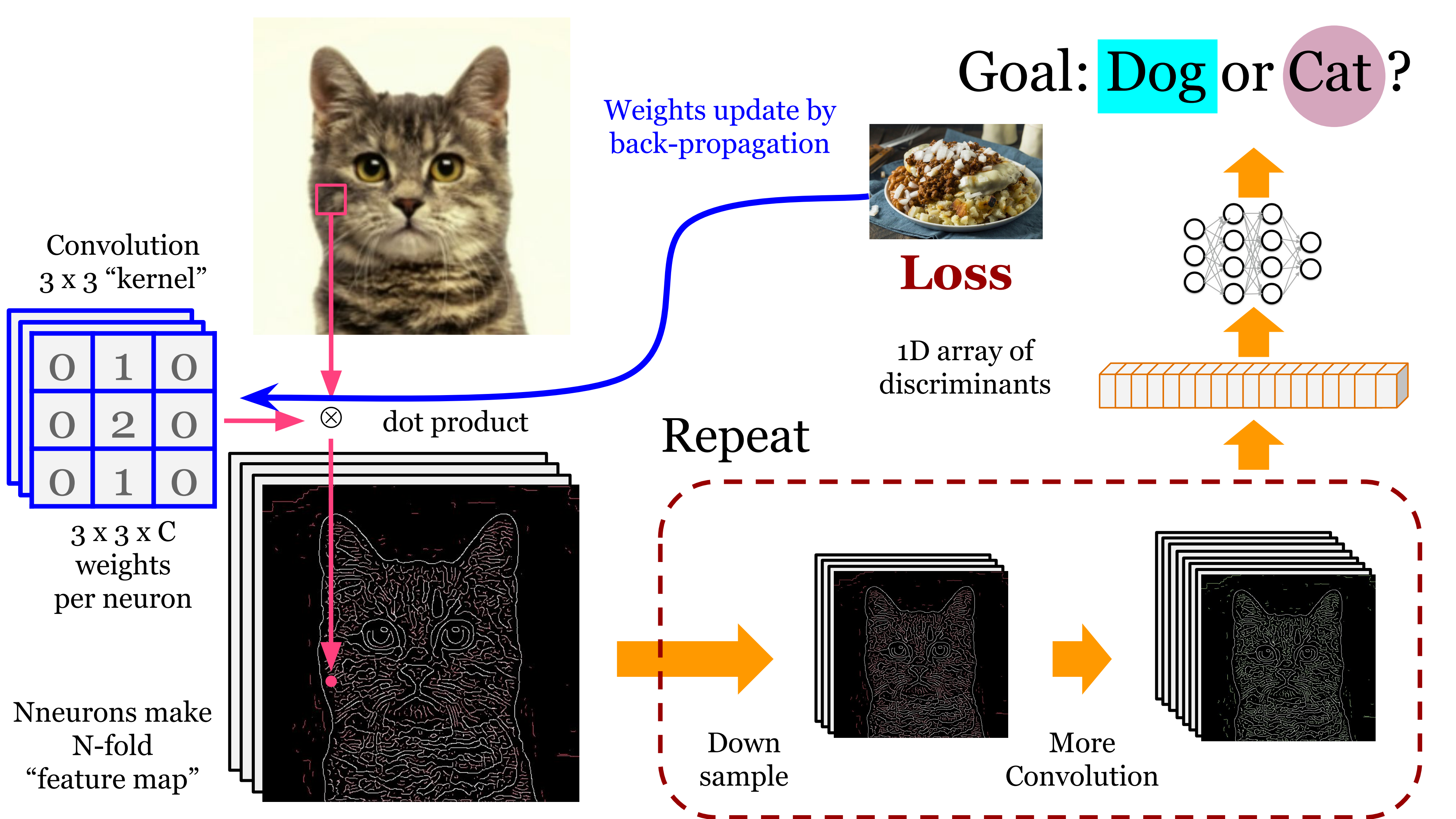




Goal: Dog or Cat ?
Convolution
3 x 3 "kernel"
0 1 0
0 2 0
0 1 0
3 x 3 x C
weights
per neuron
dot product
Nneurons make
N-fold
"feature map"
Down
sample
More
Convolution


Goal: Dog or Cat ?
Convolution
3 x 3 “kernel”
0 1 0
0 2 0
0 1 0
3 x 3 x C
weights
per neuron
dot product
Nneurons make
N-fold
“feature map”
1D array of
discriminants
Repeat
Down
sample
More
Convolution


Goal: Dog or Cat ?
Convolution
3 x 3 “kernel”
0 1 0
0 2 0
0 1 0
3 x 3 x C
weights
per neuron
Nneurons make
N-fold
“feature map”
dot product
Loss
1D array of
discriminants
Repeat
Down
sample
More
Convolution



Goal: Dog or Cat ?
Weights update by
back-propagation
Loss
Convolution
3 x 3 "kernel"
0 1 0
0 2 0
0 1 0
3 x 3 x C
weights
per neuron
dot product
1D array of
discriminants
Repeat
Nneurons make
N-fold
"feature map"
Down
sample
More
Convolution

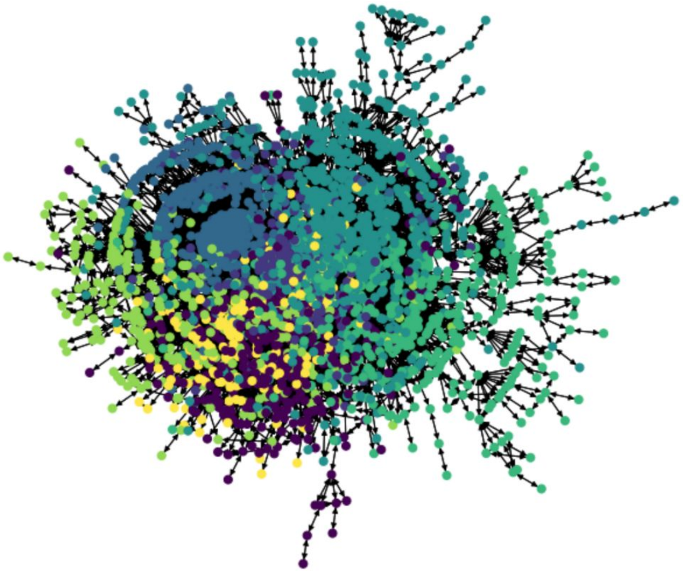
Graph Neural Networks
for Unstructured Data

Graph Neural Networks

CNNs for data that live (are projected) in images.

Feature extraction by analyzing local neighbors defined by regular grids. But more in general, data is irregular with complex neighbor/relation definition.

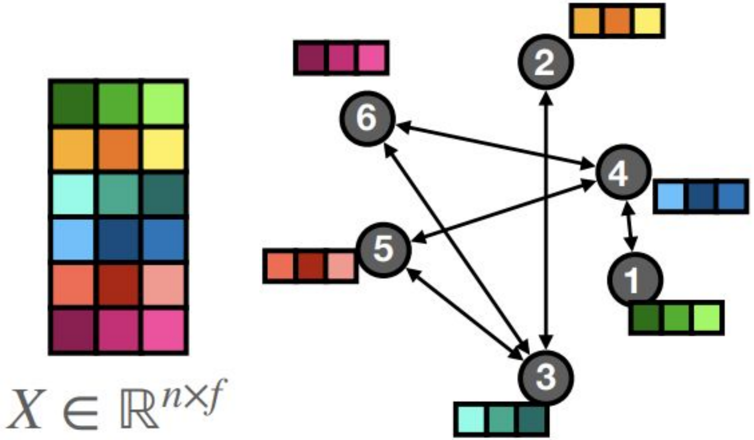
A social network



Citation/Reference Map

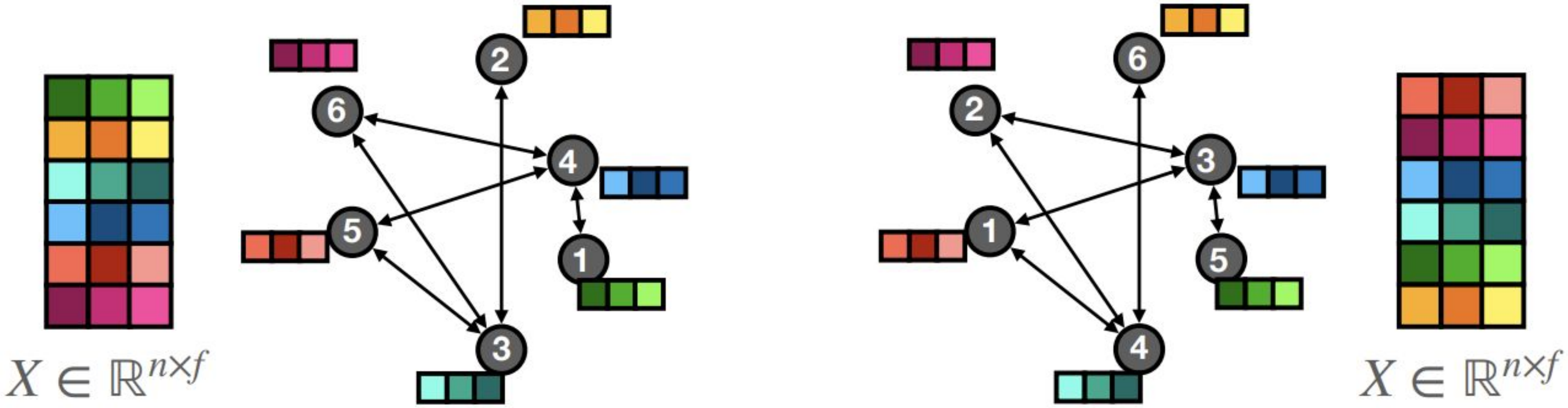
Graph Neural Networks

Graph operations can be encoded into matrix multiplication just like convolutions = can utilize parallel processors (e.g. GPUs)



Graph Neural Networks

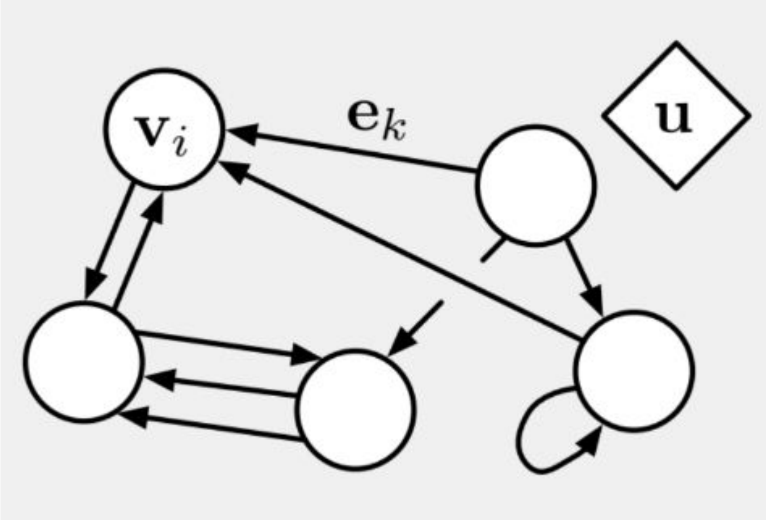
Graph operations can be encoded into matrix multiplication just like convolutions = can utilize parallel processors (e.g. GPUs)



However it shouldn't depend on an arbitrary ordering scheme. GNNs can exploit permutation invariance

Graph Neural Networks

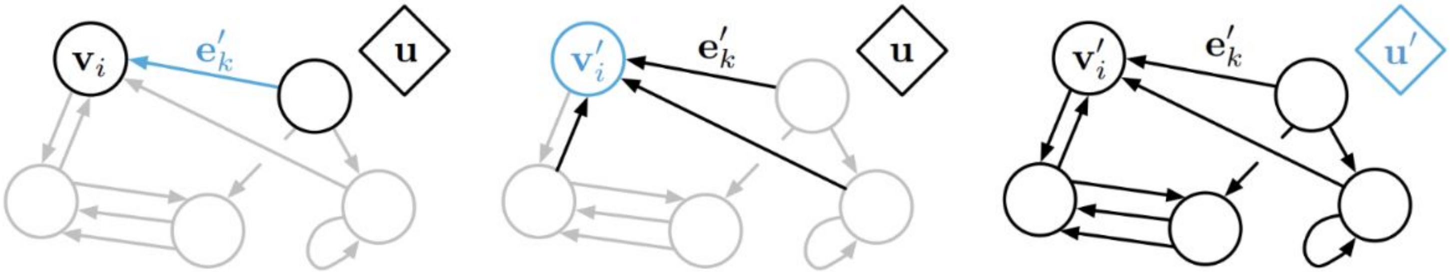
Parts: “nodes”, “edges”, and the “graph” (as a whole)



Graph Neural Networks

Parts: “nodes”, “edges”, and the “graph” (as a whole)

Convolutions: extract “local features” from connected neighbors



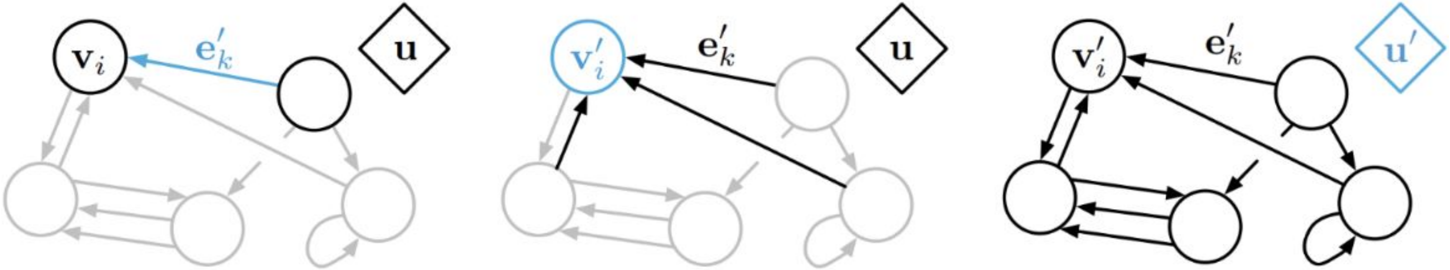
(a) Edge update (b) Node update (c) Global update

Graph Neural Networks

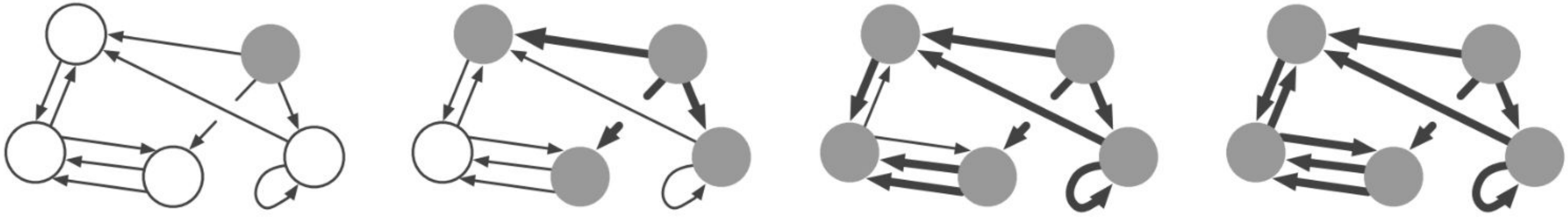
Parts: "nodes", "edges", and the "graph" (as a whole)

Convolutions: extract "local features" from connected neighbors

Message Passing: repeated convolution propagates information



(a) Edge update (b) Node update (c) Global update



$m = 0$ $m = 1$ $m = 2$ $m = 3$

Tasks for GNNs

- Node classification/regression
- Edge classification/regression
- Graph classification/regression