CS 5043
Advanced Machine Learning

Andrew H. Fagg
Symbiotic Computing Laboratory
School of Computer Science

What is Machine Learning?

What is Machine Learning?

- Fundamentally: using data to automatically construct models
- The models must be predictive!
  - To be useful, a model must produce meaningful output given novel situations.

Classes of Machine Learning Problems

Classes of Machine Learning Problems

Unsupervised learning:

- The training set contains only inputs
- Fundamental question: what is the structure of these inputs?
  - A common case: algorithm assigns categorical labels to each sample (clustering)
  - But we can also ask continuous questions:
    - Are there linear or nonlinear manifolds that the data live on?
    - Is there a lower-dimensional representation of the data?

Classes of Machine Learning Problems

Supervised learning:

- Training set contains both input / desired output (labels) pairs
- Model outputs given some new input can be continuous, probabilistic or categorical

Classes of Machine Learning Problems

Reinforcement learning:

- Different than direct prediction or classification: RL is about taking sequences of actions in some environment
- At each step:
  - In response to an input, the model (agent) produces some action
  - The feedback signal is an evaluation of the results of this and previous actions

Classes of Machine Learning Problems

Reinforcement learning:

- A common case: a single evaluation can be a function of the sequence of outputs that is generated
  - How much time did it take to solve a task?
  - How much energy did you use while solving the task?
  - How well did you solve the task?

- Learning problem: for a given input, what is the output action that maximizes the expected reinforcement over time?

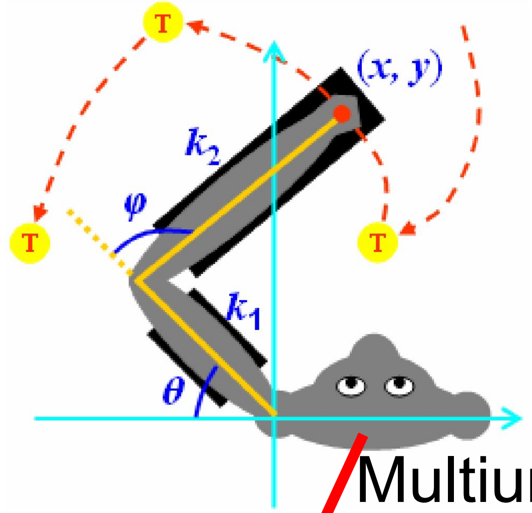
Supervised Learning Example

Brain-machine interfaces

Brain-Machine Interfaces

Estimate of intended movement

Command prosthetic arm



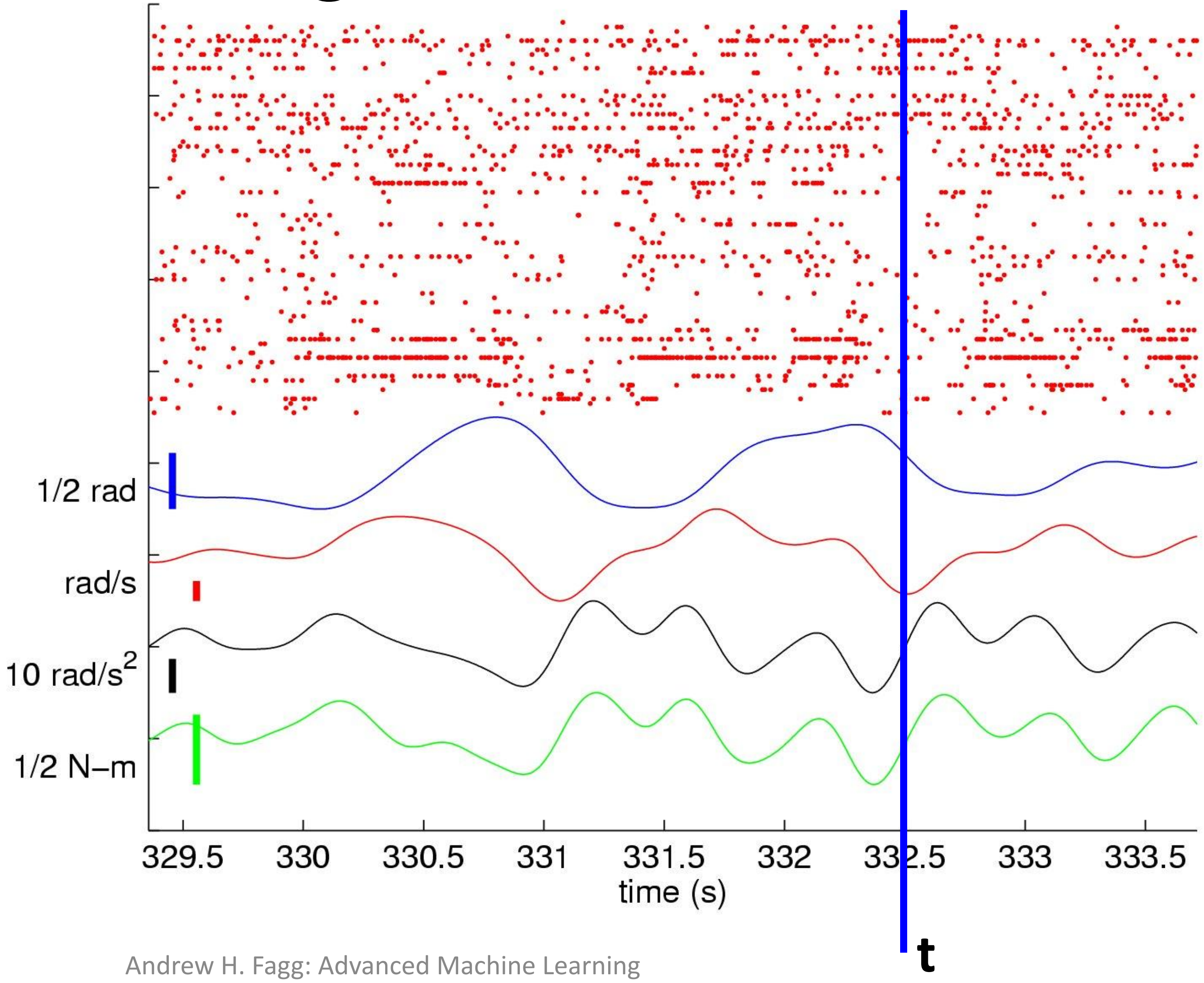
Multiunit recording

Predictive model

In collaboration with Nicholas G. Hatsopoulos, Aaron Suminski and Lee E. Miller

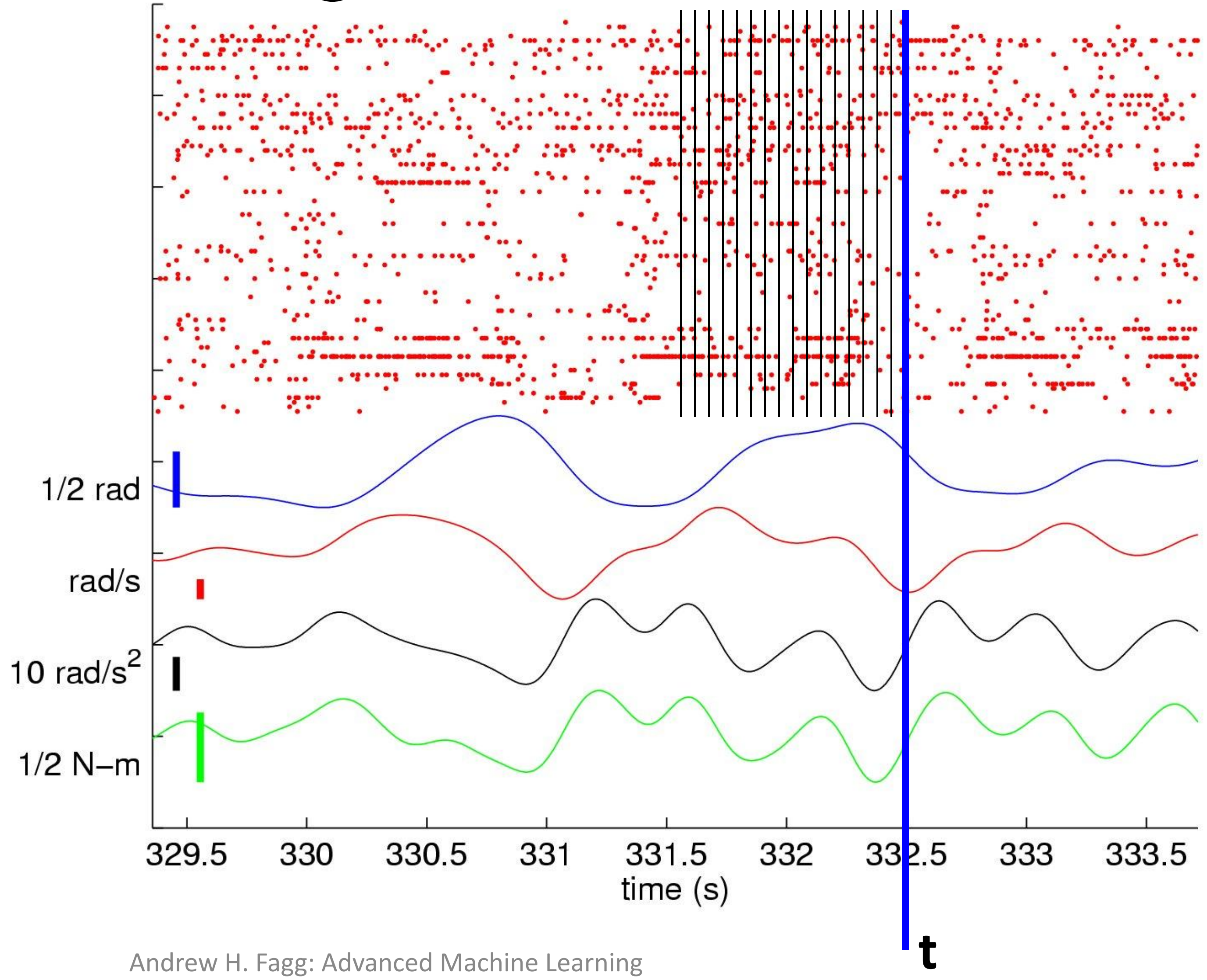
Decoding Arm State

Want to predict arm motion at time t given recent history of spiking behavior



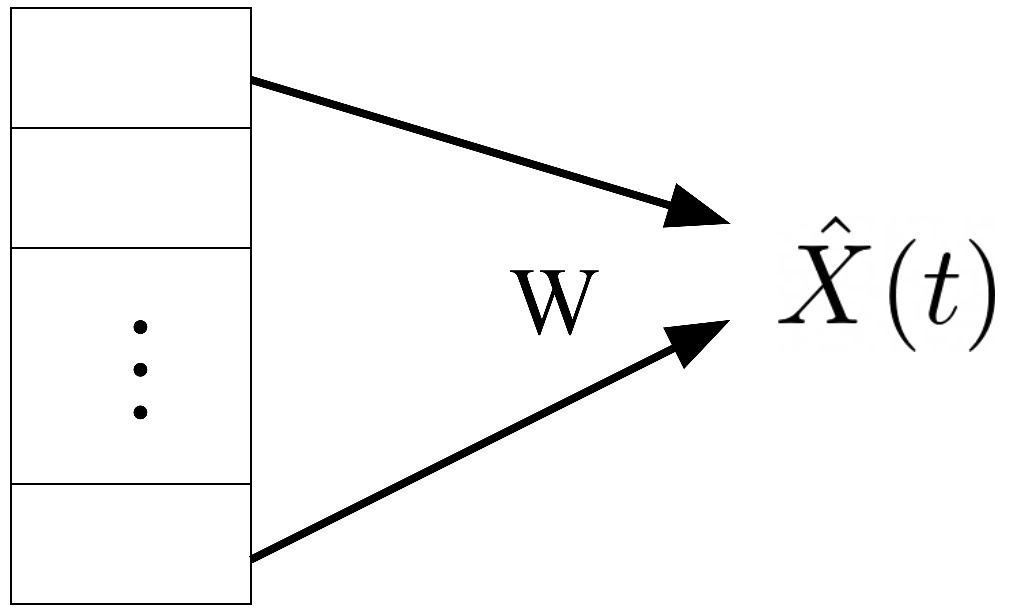
Decoding Arm State

50ms bins: 20 descriptors of neural activation for each cell



Wiener Filter

Each feature (F_i) is a count of spikes by a neuron for a 50 ms bin

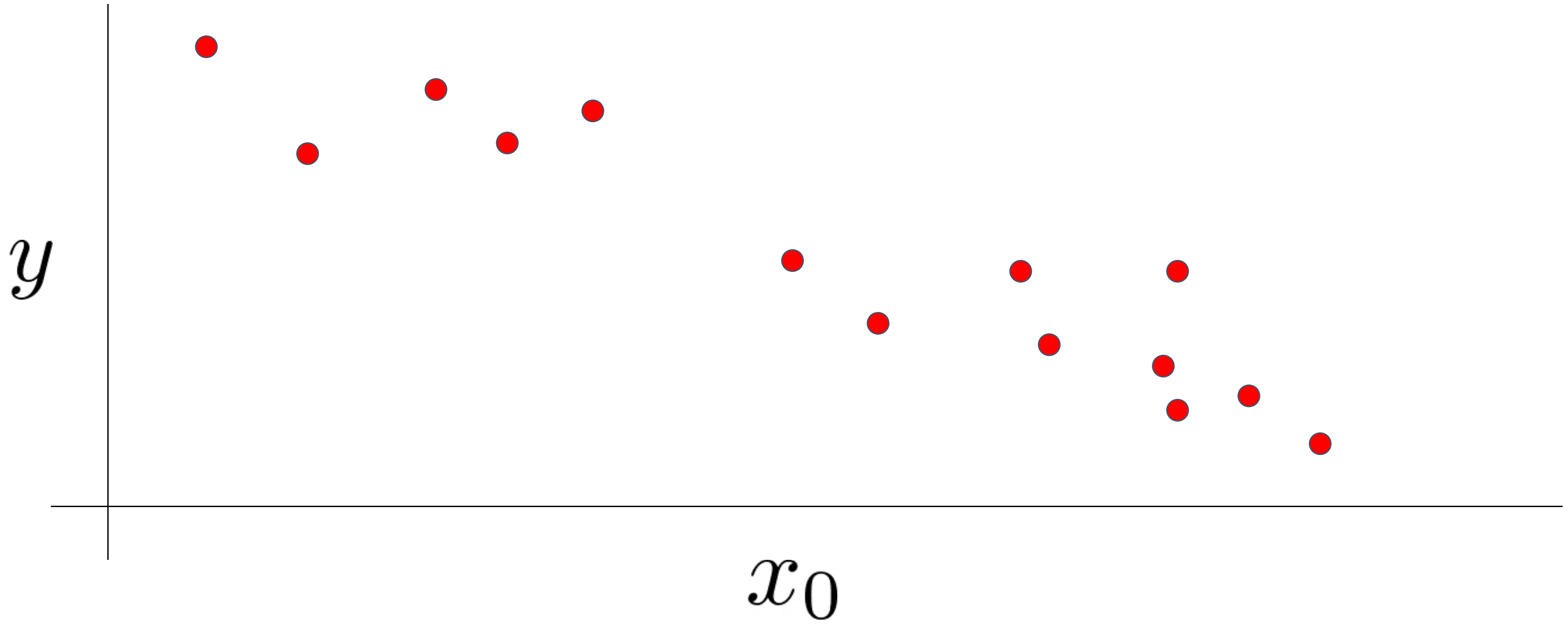


$$\hat{X}(t) = g_W(F(t)) = W^T F(t)$$

Column vector encoding spike counts for N cells at T taps up to time t

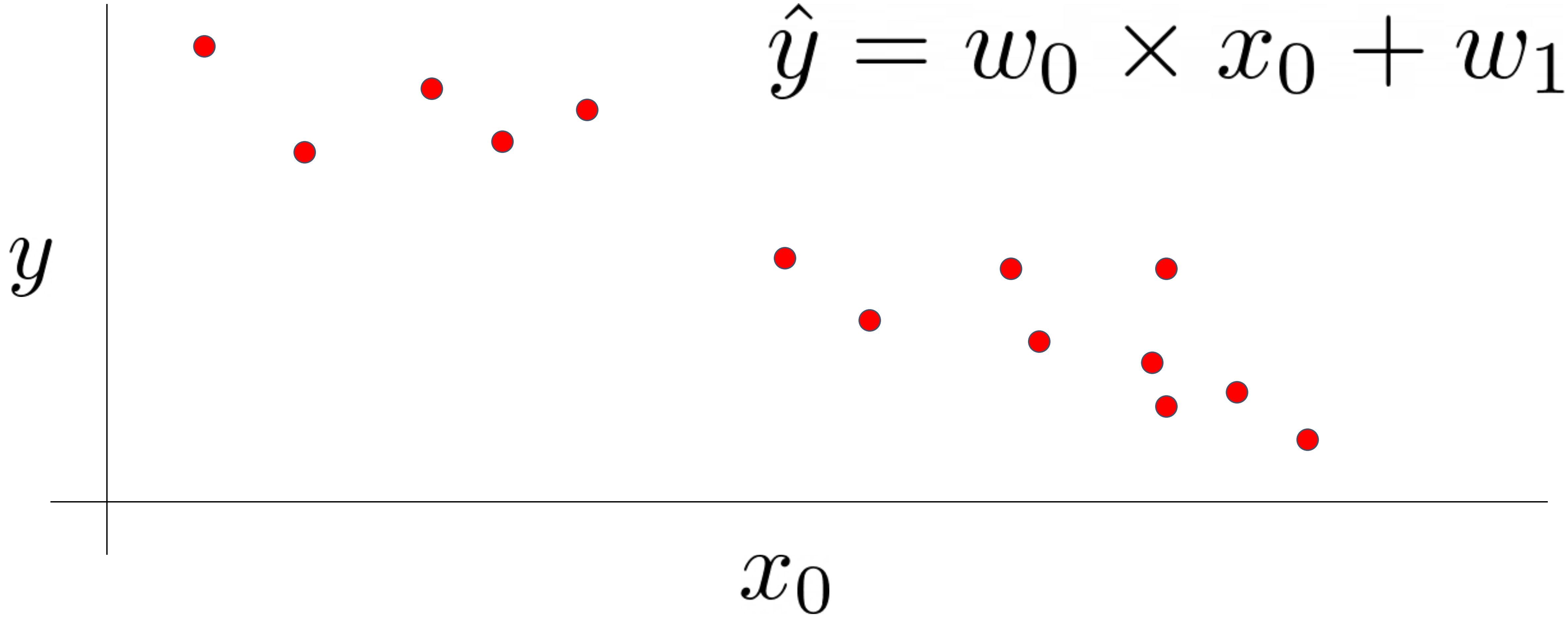
Regression

Modeling the Relationship between Two Variables



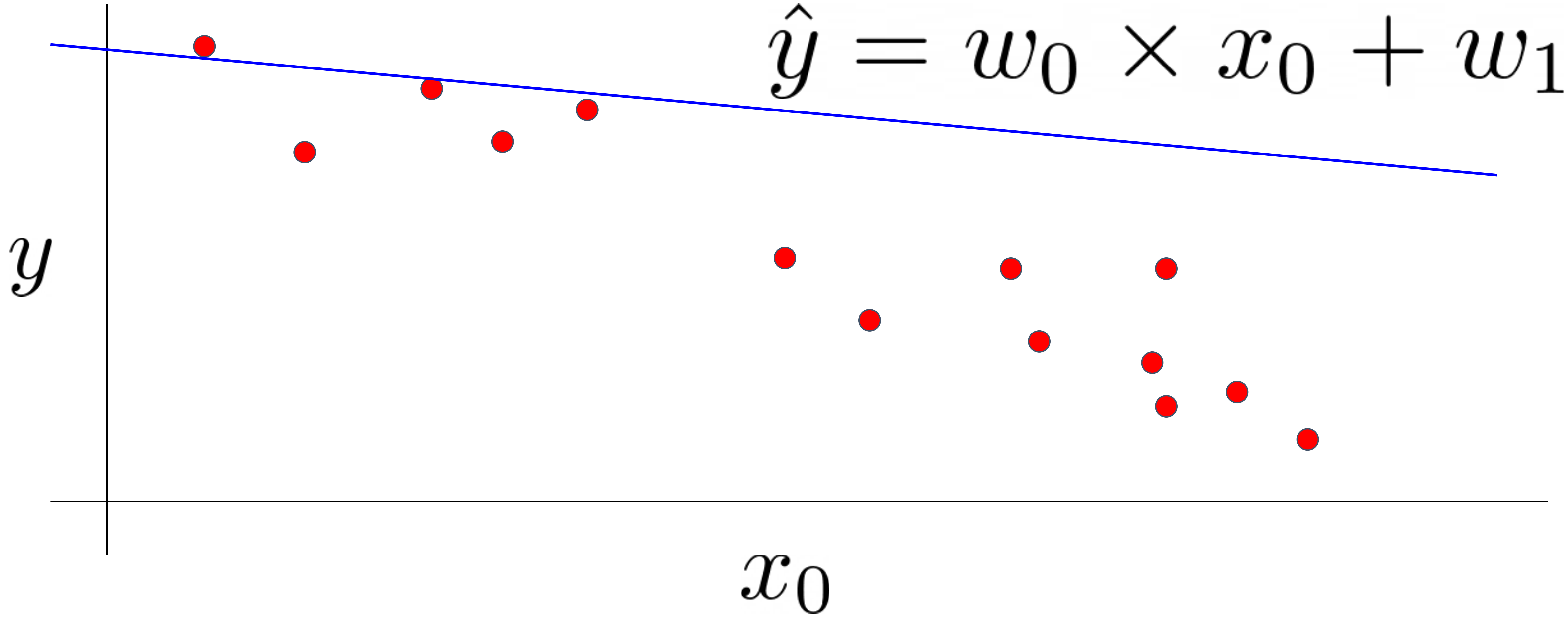
Modeling the Relationship between Two Variables

$$\hat{y} = w_0 \times x_0 + w_1$$



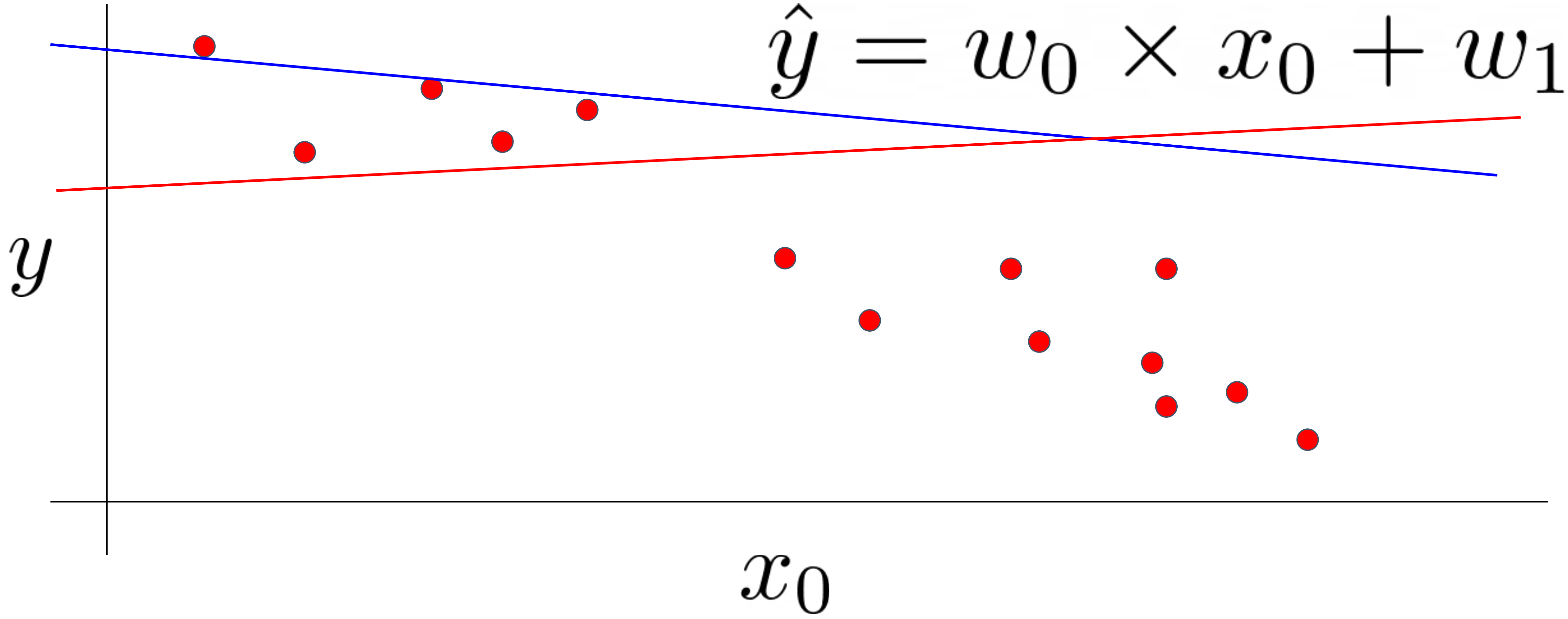
Modeling the Relationship between Two Variables

$$\hat{y} = w_0 \times x_0 + w_1$$



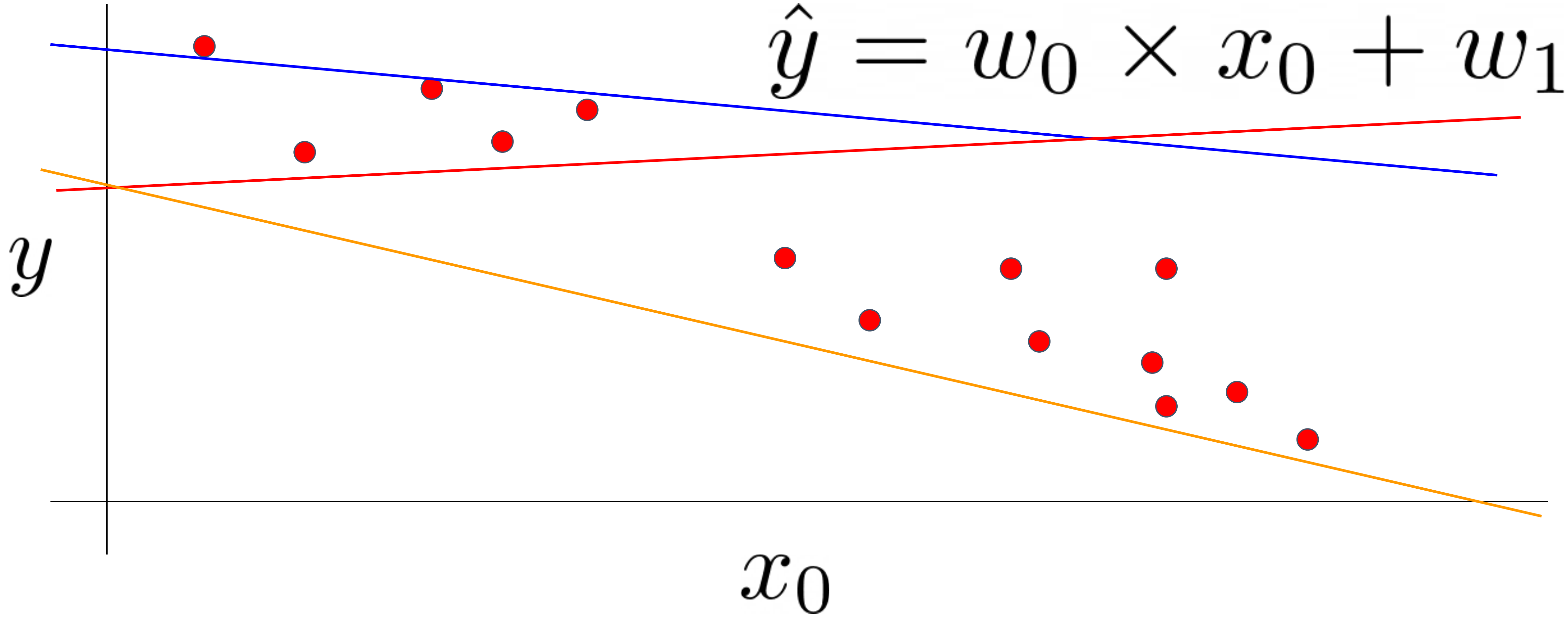
Modeling the Relationship between Two Variables

$$\hat{y} = w_0 \times x_0 + w_1$$



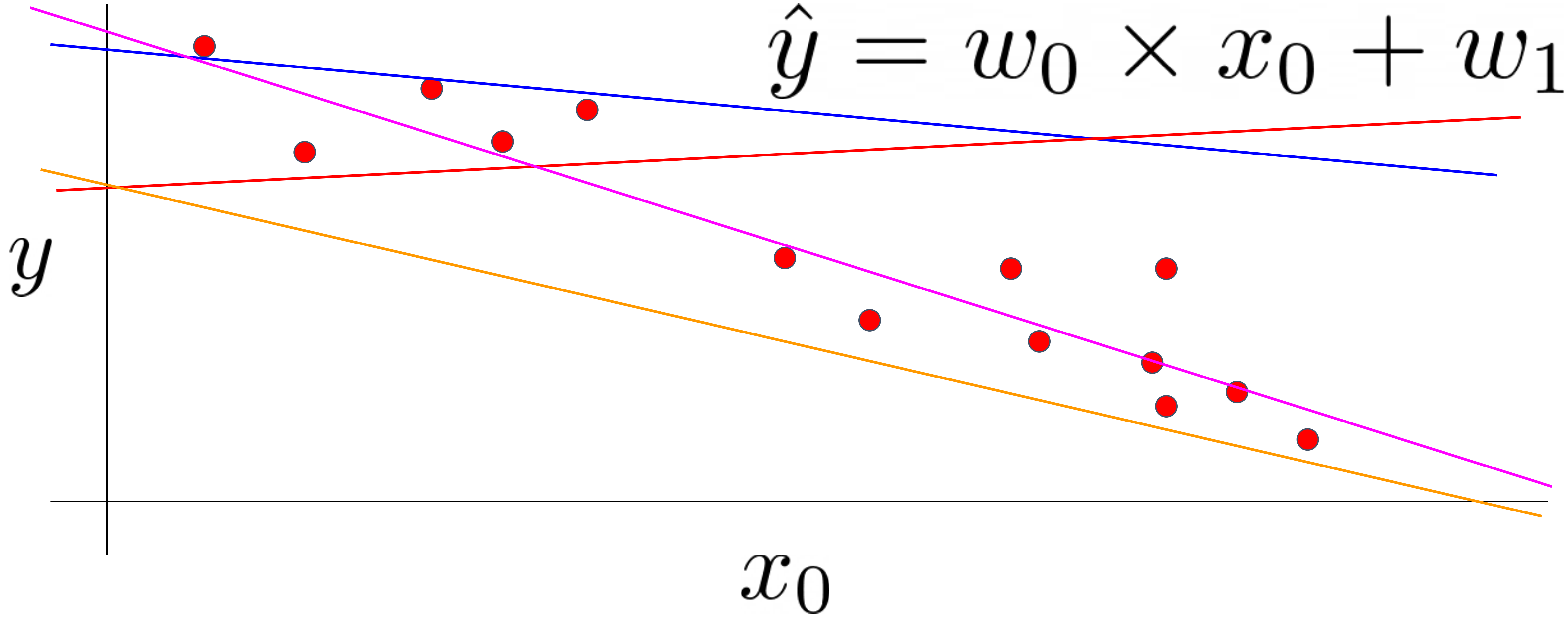
Modeling the Relationship between Two Variables

$$\hat{y} = w_0 \times x_0 + w_1$$



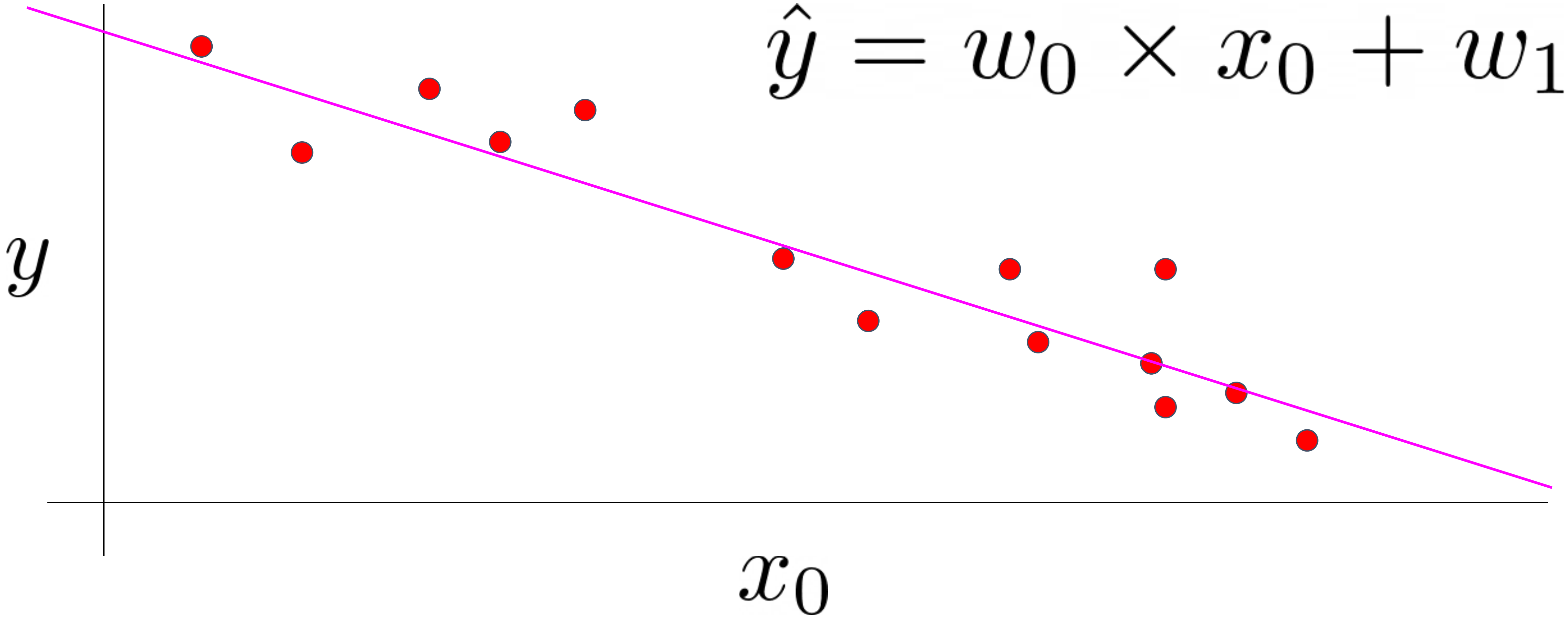
Modeling the Relationship between Two Variables

$$\hat{y} = w_0 \times x_0 + w_1$$



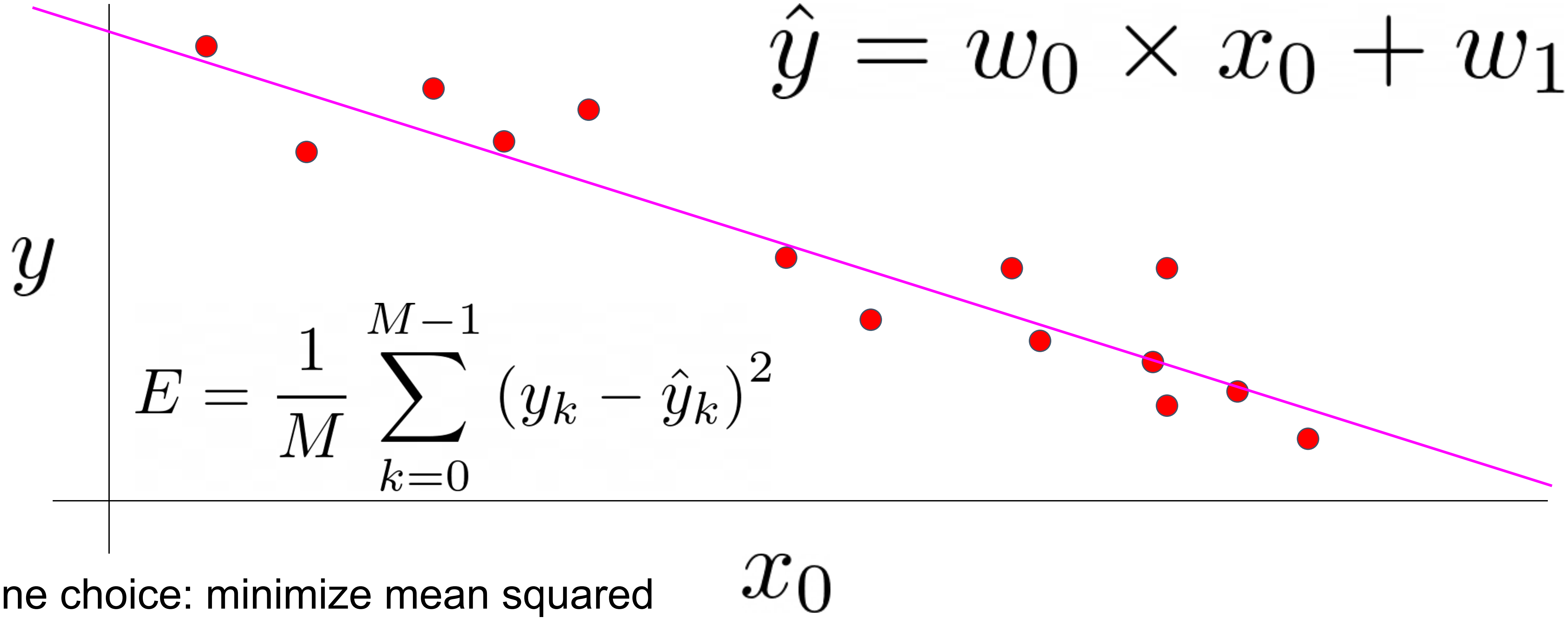
Modeling the Relationship between Two Variables

$$\hat{y} = w_0 \times x_0 + w_1$$



How did we choose?

Modeling the Relationship between Two Variables



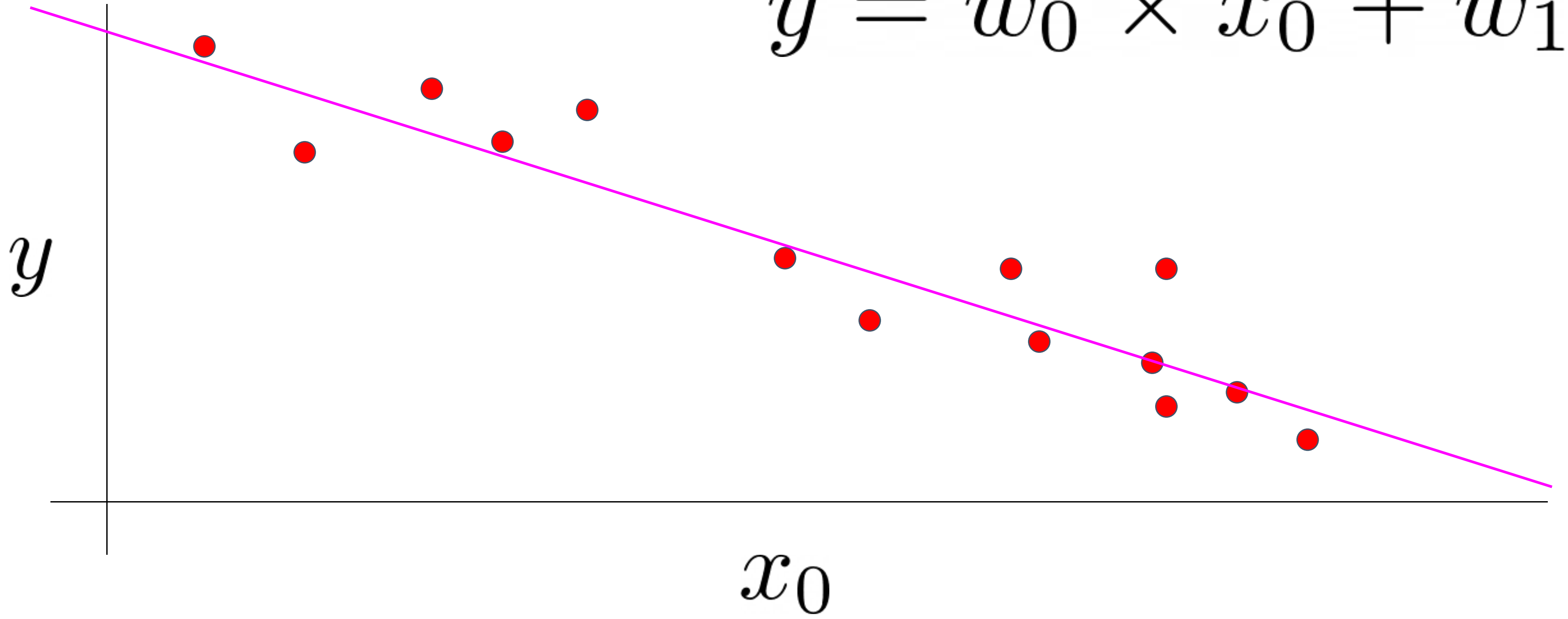
$$\hat{y} = w_0 \times x_0 + w_1$$

$$E = \frac{1}{M} \sum_{k=0}^{M-1} (y_k - \hat{y}_k)^2$$

One choice: minimize mean squared prediction error over M examples (abbrev: MSE or LMS)

Using the Model to Make Predictions

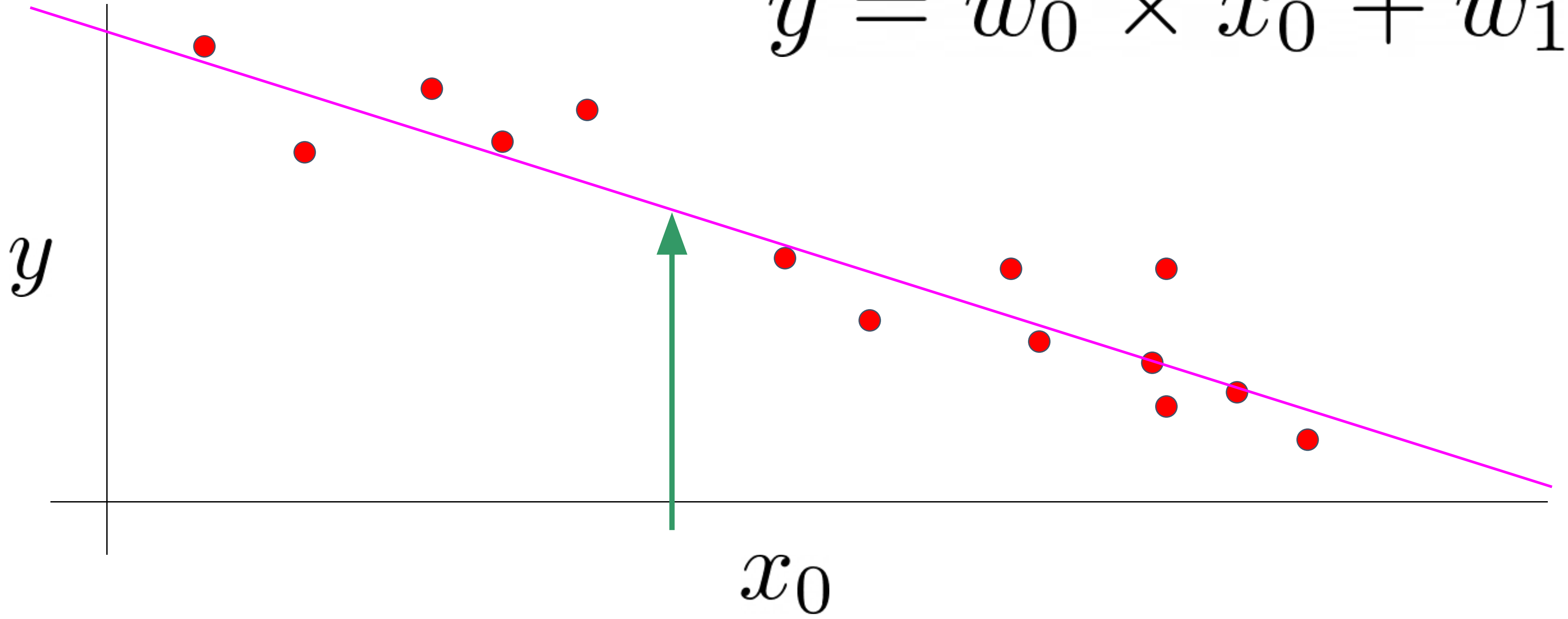
$$\hat{y} = w_0 \times x_0 + w_1$$



How did we choose?

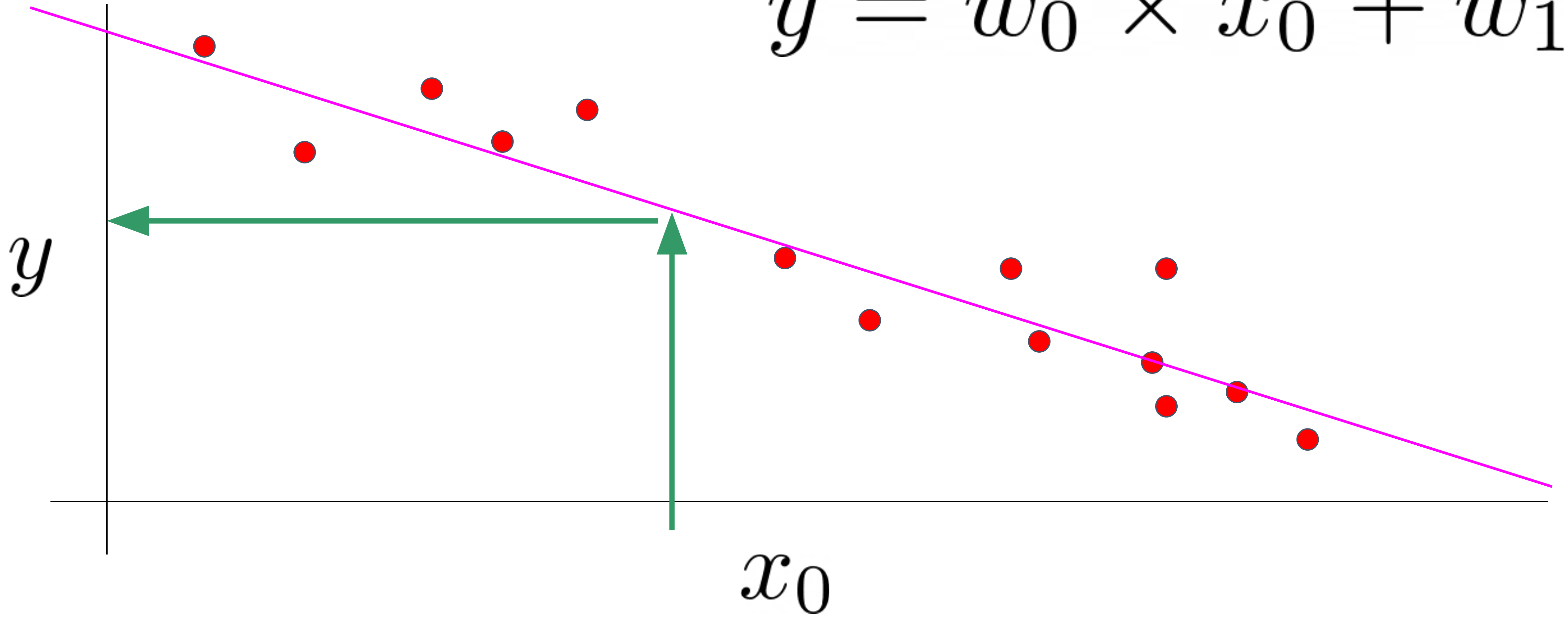
Using the Model to Make Predictions

$$\hat{y} = w_0 \times x_0 + w_1$$



Using the Model to Make Predictions

$$\hat{y} = w_0 \times x_0 + w_1$$



Computing a “Good” Model

Must define what we mean by good

- Common for this case:
  - Compute the sum of the squared prediction errors for a given W:

$$E_W = \sum_{t=0}^{T-1} \left(X(t) - \hat{X}(t) \right)^2$$

  - Choose the parameters so as to minimize this error metric
    Least Mean Squared (LMS) error:

$$\hat{W} = \arg\max_{W} \; E_W$$

Finding a Quality Model

- For BMI problem, we typically have:
  - ~1000 parameters
  - ~1000-20,000 examples
- Easy for LMS to over-fit these data
  - Great performance on the training data
  - But … poor performance on independent data
- One approach: modify the error function to punish large magnitude parameters
  - Regularization!

Even Harder Problems…

Let’s consider higher-dimensional inputs …

    10K, 100K, 1M variables …

Image Understanding

How do we make sense of natural scenes?

Schulter, et al. (2018)

Image Understanding

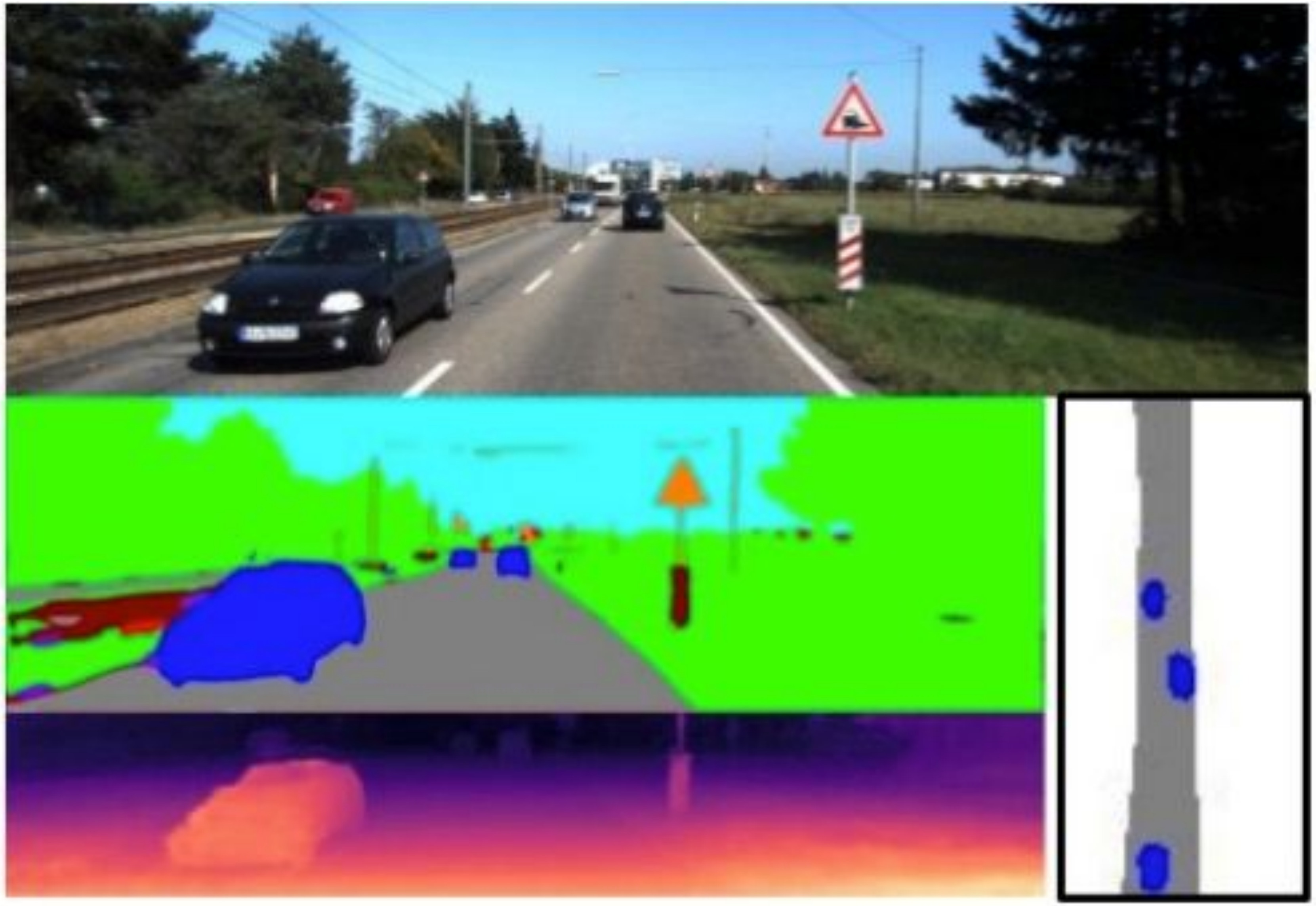


Image Understanding

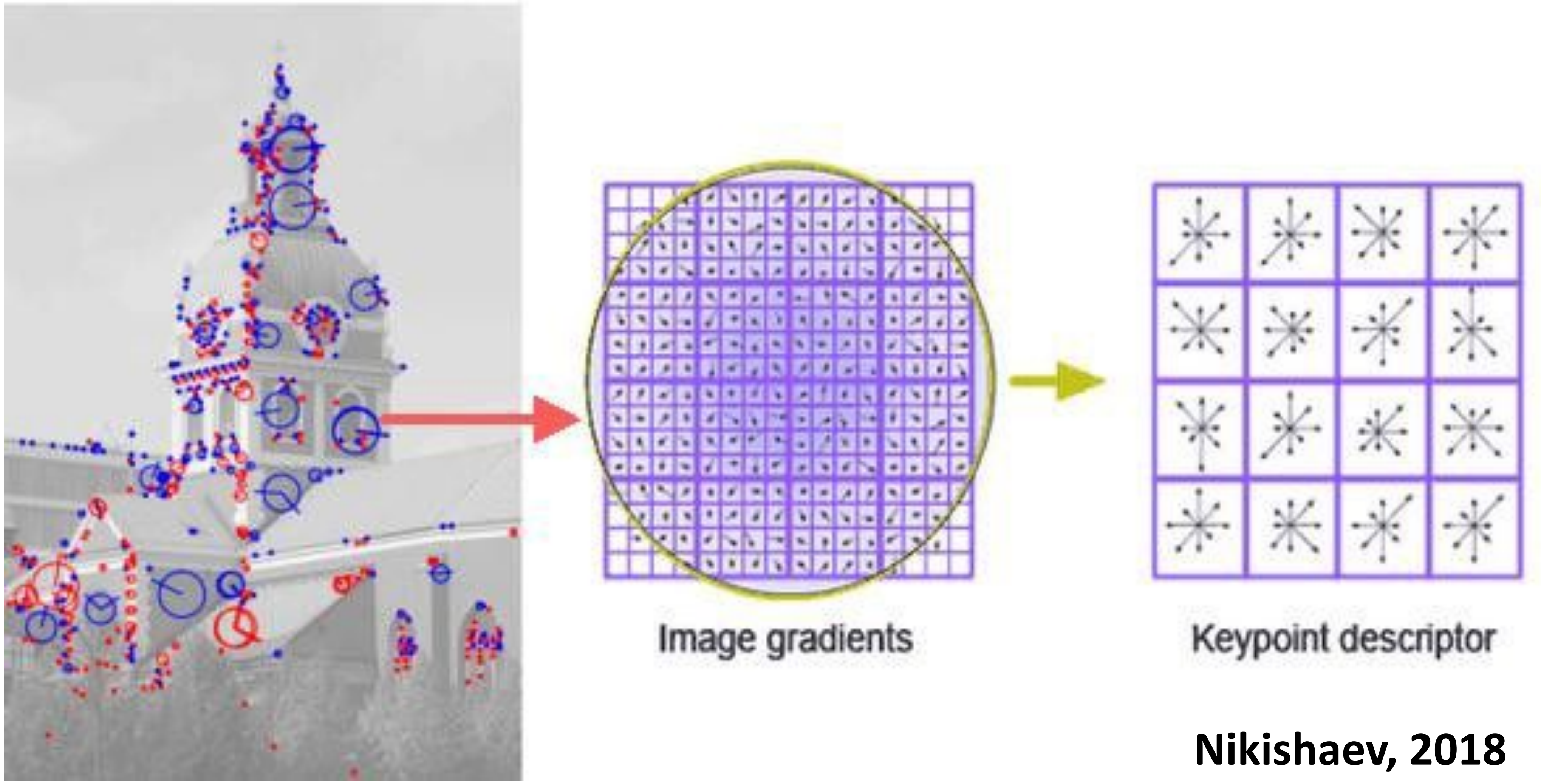
- High-dimensional inputs
- Many different cases
- Even a linear model would require a large number of parameters
- Big risk of overfitting

Computer Vision

Much of computer vision has been about hand-crafting feature detectors that allow us to extract just the right information….

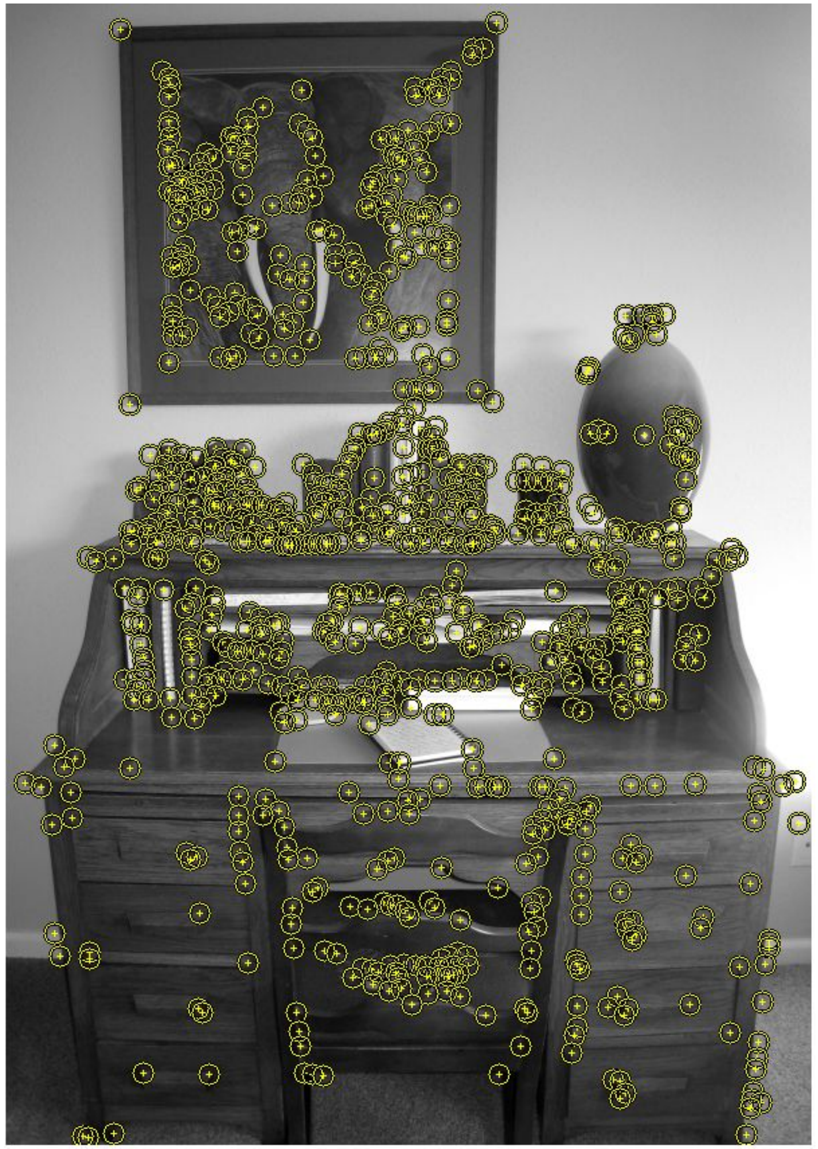
- Edges, corners, blobs, …
- Many have been hand-tuned
- Identified features allow us to summarize an image with a smaller number of descriptors
- It is (was) then feasible to use ML to effectively learn the parameters of the simpler model

Computer Vision: SIFT Features



Nikishaev, 2018

Computer Vision: SURF Features

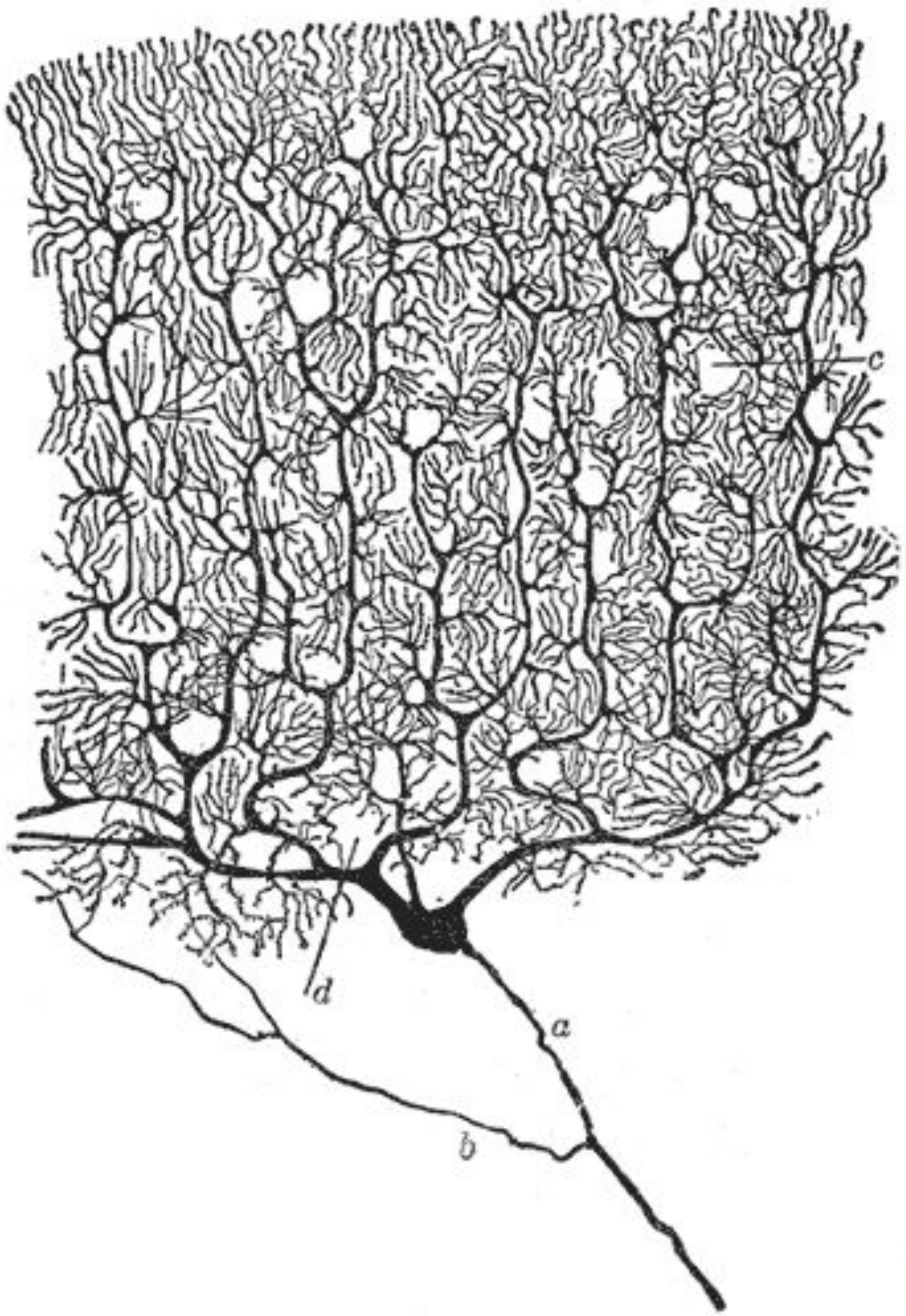


Kleliboux et al. (2010)

But …

- What are the right features to be paying attention to?
- Could we also learn an appropriate set of operators from the data?
- This takes us back to more complex models…

Neural Networks

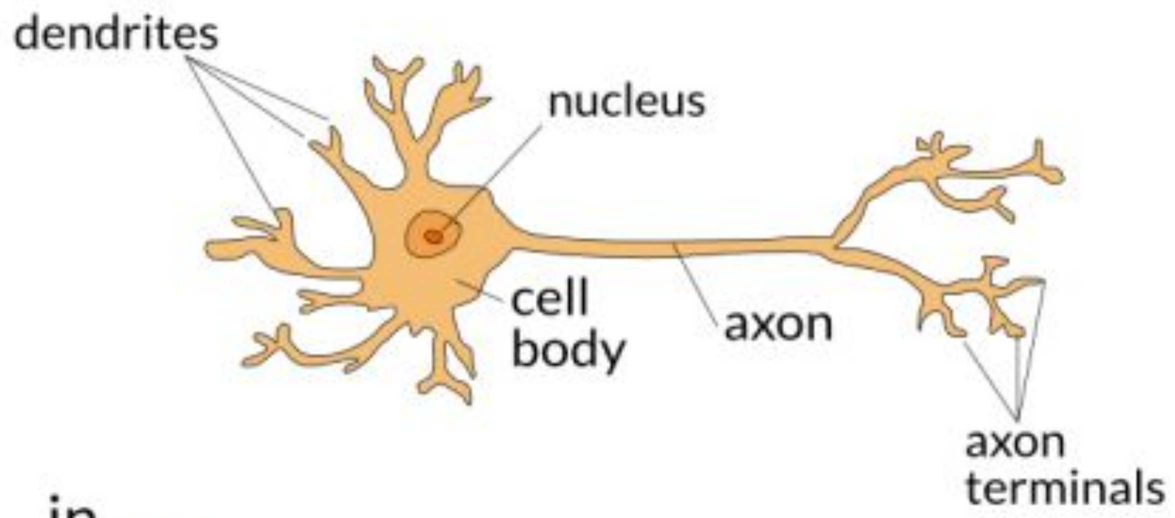


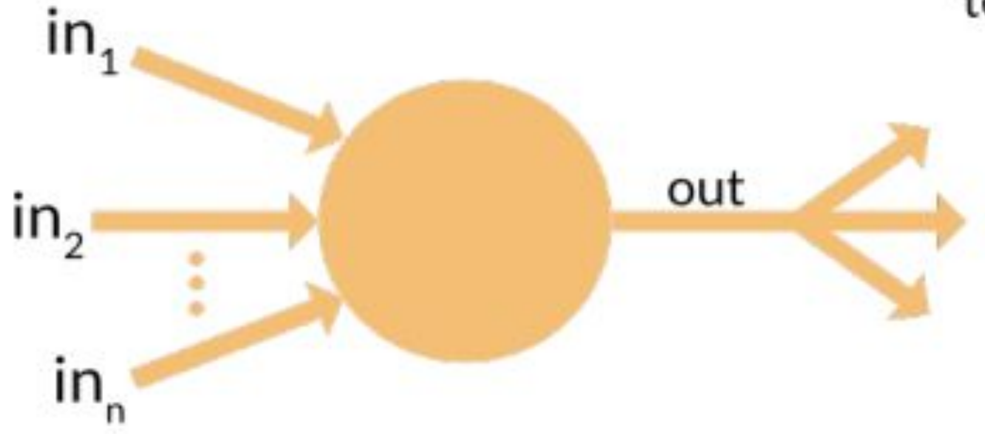
Santiago Ramón y Cajal: single Purkinje cell

from the cat cerebellar cortex

https://commons.wikimedia.org/w/index.php?curid=10650411

Neurons

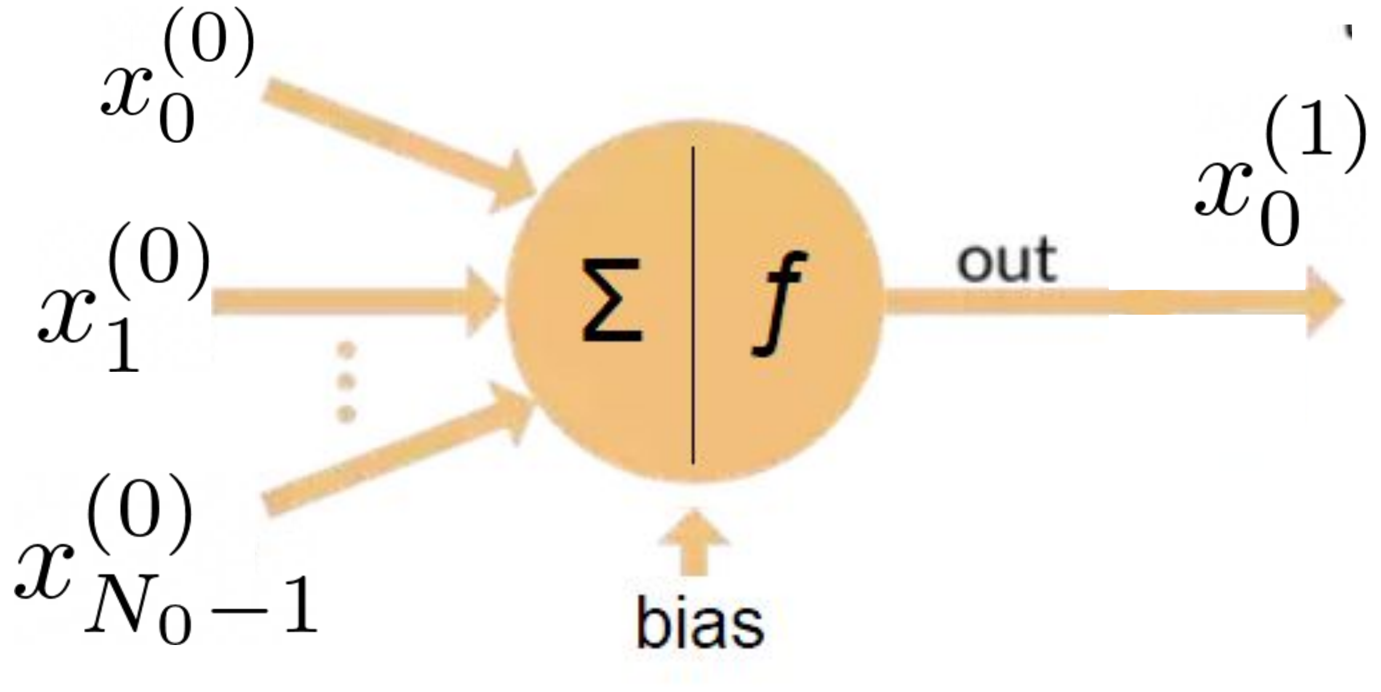




Wu (UCLA)

Individual Computing Elements

Linear transformation + a simple non-linearity

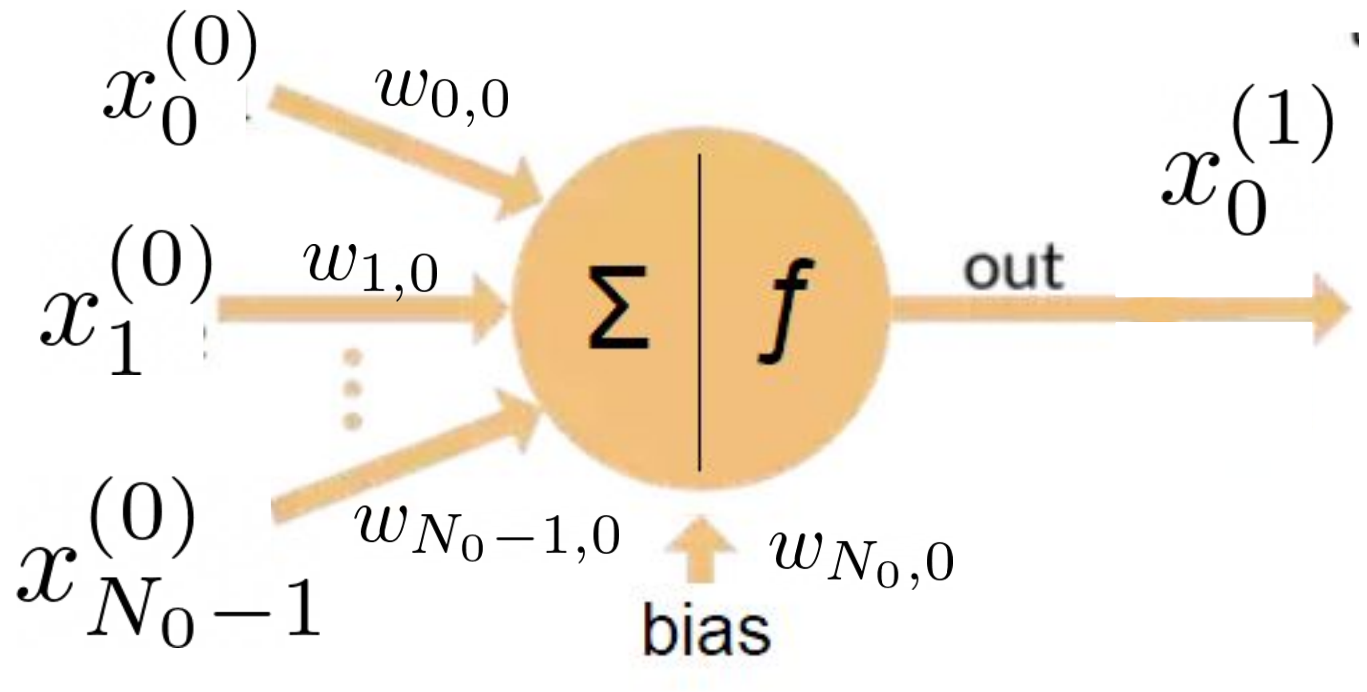


Wu (UCLA)

Layer 0

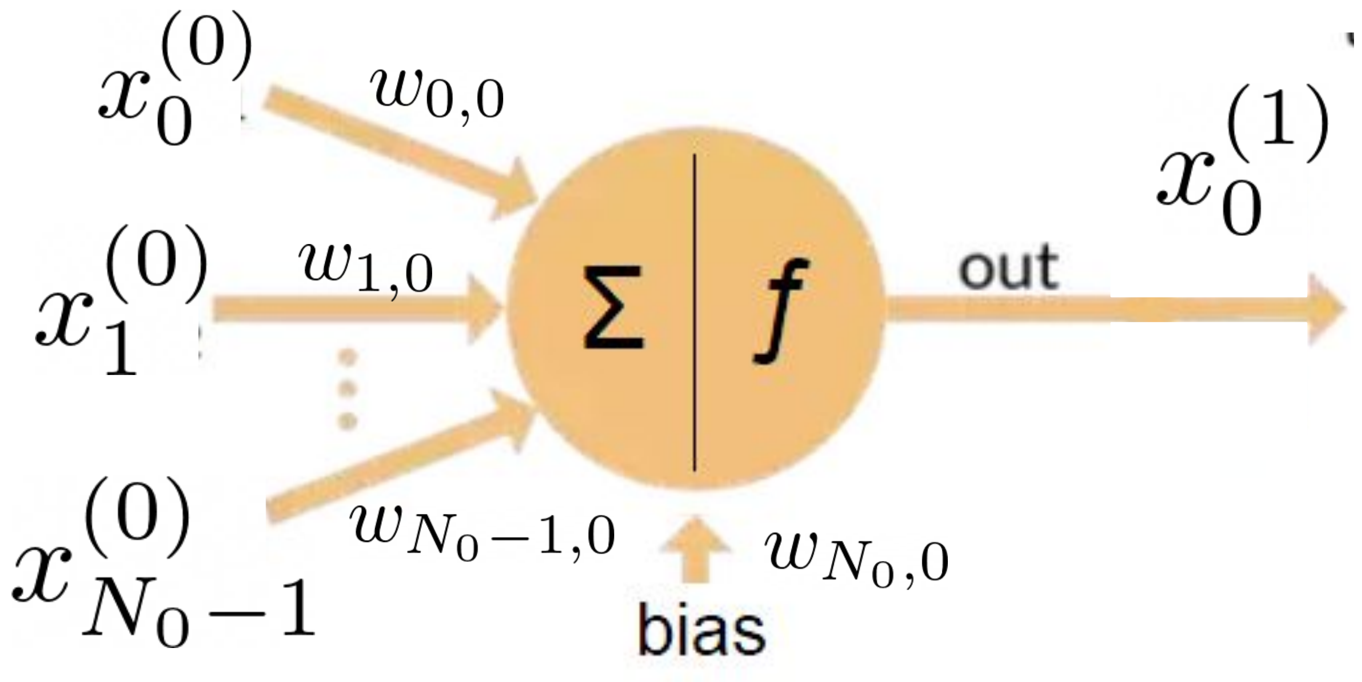
Layer 1

Individual Computing Elements



Wu (UCLA)

Individual Computing Elements



Wu (UCLA)

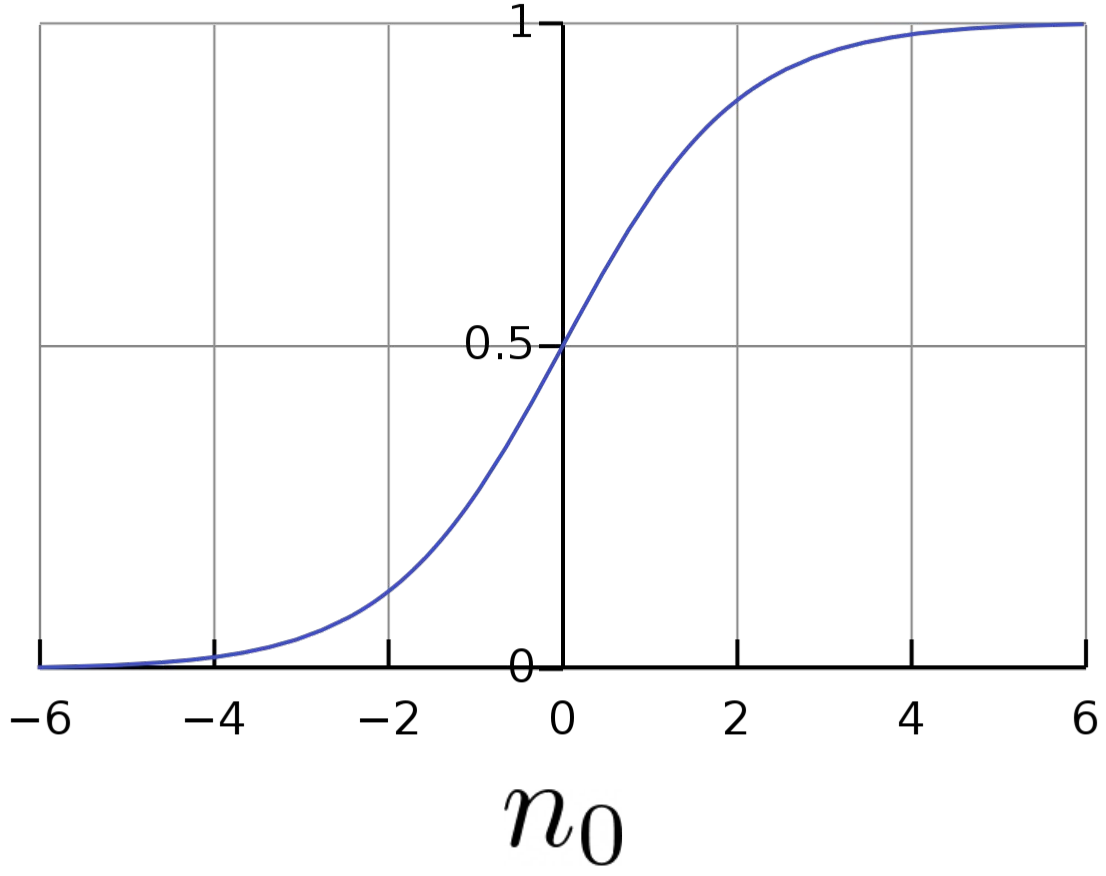
$$n_0^{(1)} = \sum_{i=0}^{N_0-1} w_{i,0} \times x_i^{(0)} + w_{N_0,0} \qquad x_0^{(1)} = f\left(n_0^{(1)}\right)$$

Many Non-Linearities are Possible

Sigmoid or Logistic function:

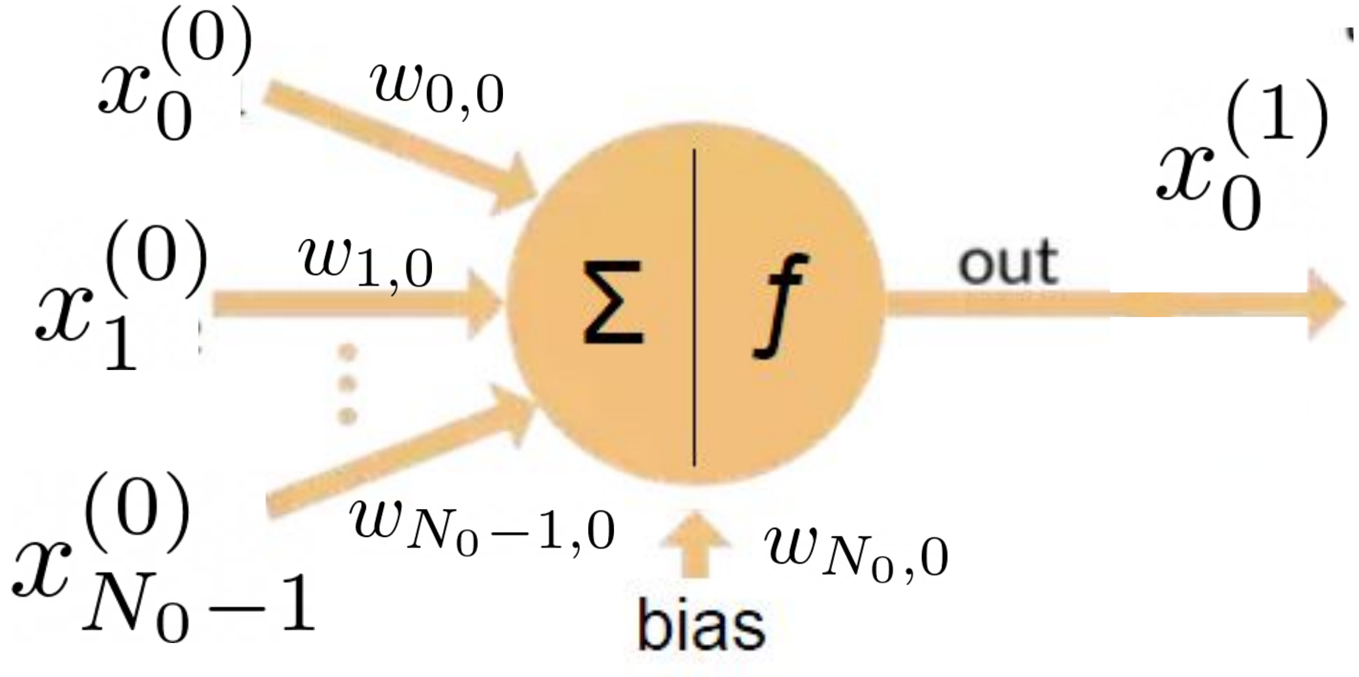
$$x_0 = f(n_0)$$

$$= \frac{1}{1 + e^{-n_0}}$$



https://en.wikipedia.org/wiki/Sigmoid_function

Individual Computing Elements

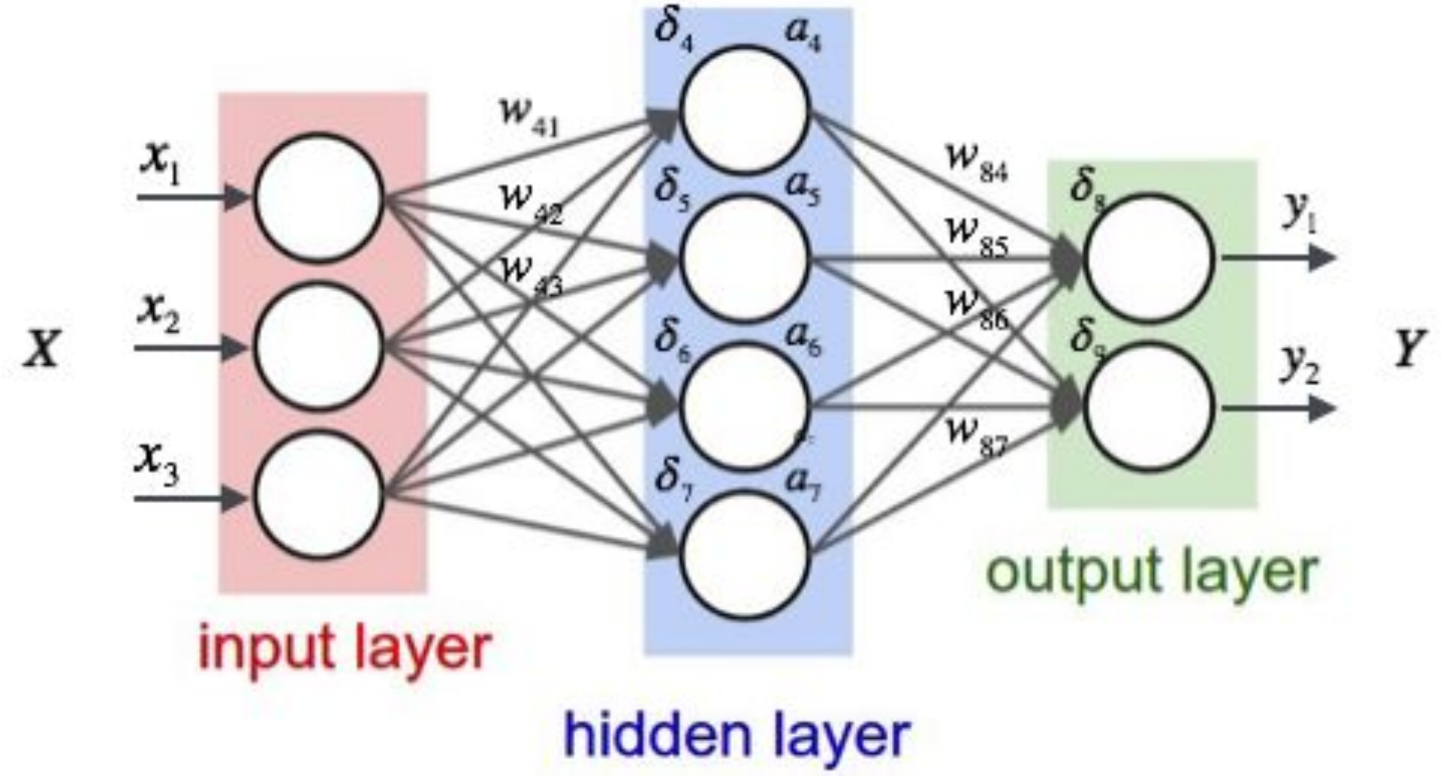


Wu (UCLA)

$$n_0^{(1)} = \sum_{i=0}^{N_0-1} w_{i,0} \times x_i^{(0)} + w_{N_0,0} \qquad x_0^{(1)} = f\left(n_0^{(1)}\right)$$

Networks of Neurons

- Parameters are tunable for both the hidden and output layers
- Hidden layer becomes our feature detectors



Valkov (2017)

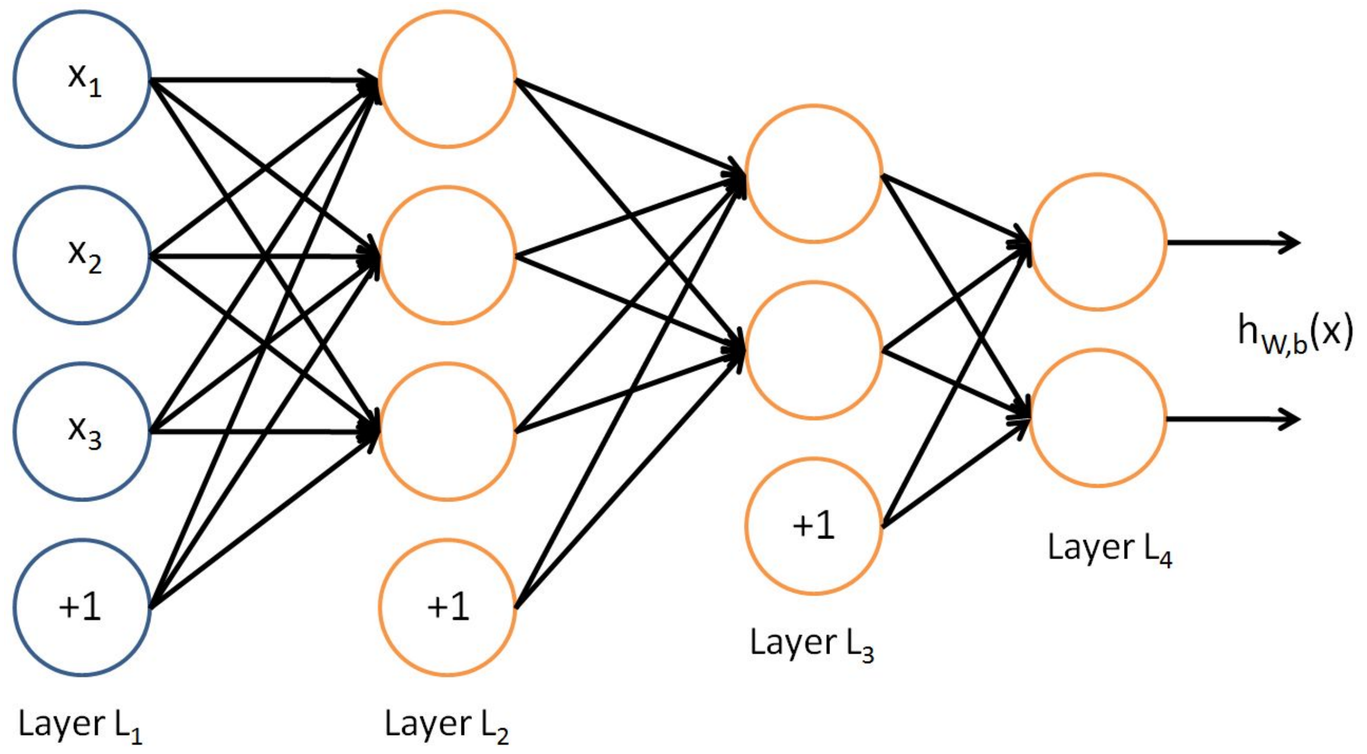
Networks of Neurons

Features are abstractions

- True if these are hand coded or if they are learned!
- Does it make sense to construct abstractions of abstractions?

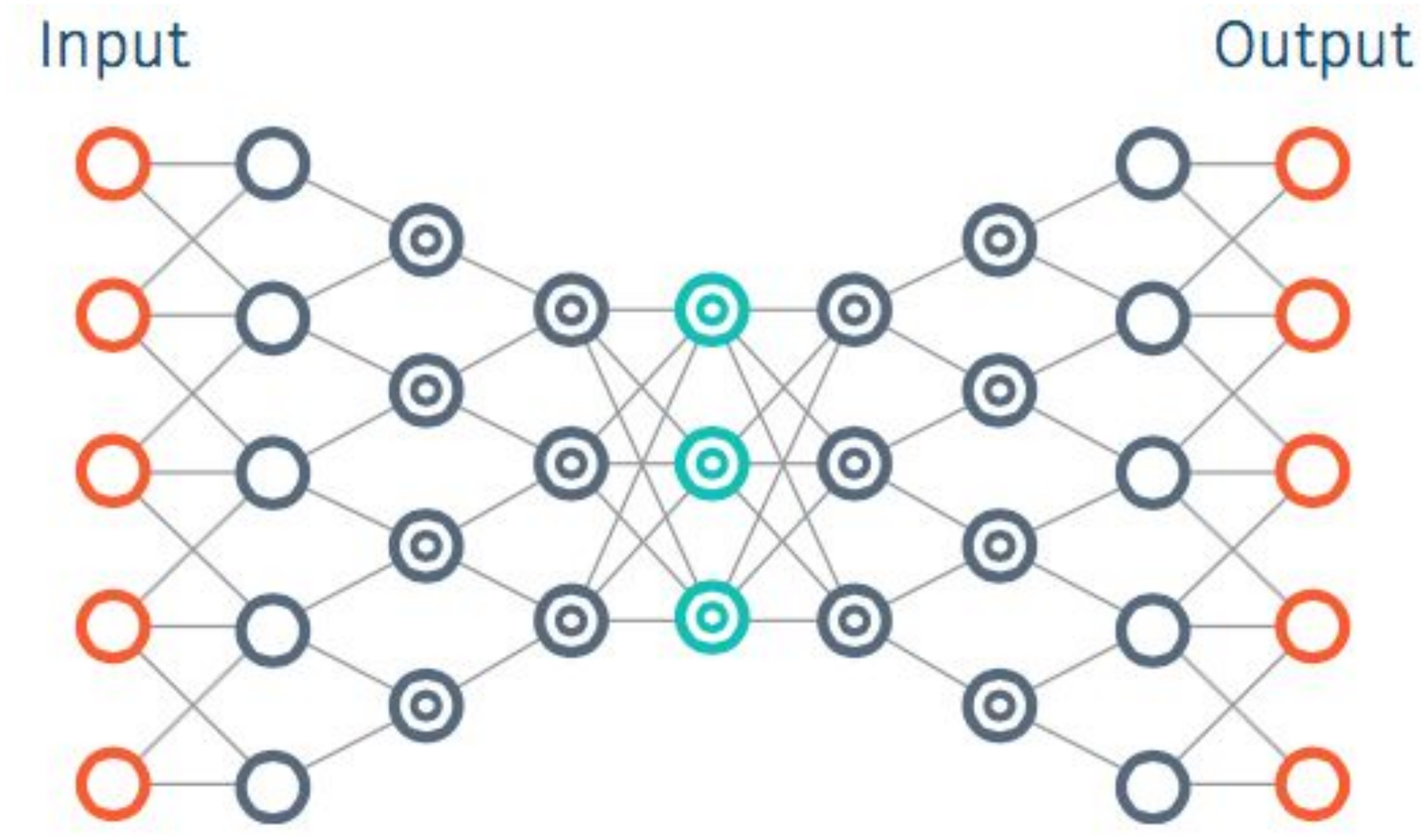
Deeper Networks

Abstractions of abstractions: multiple hidden layers

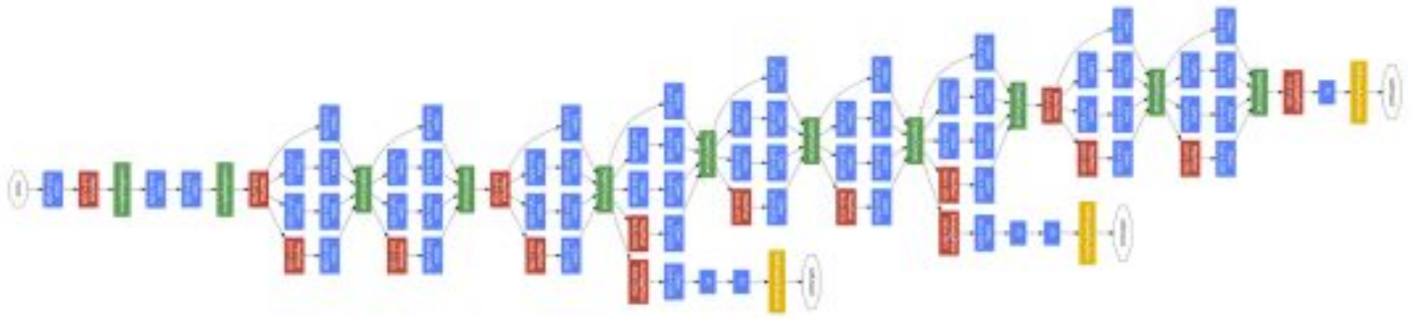


http://deeplearning.stanford.edu/

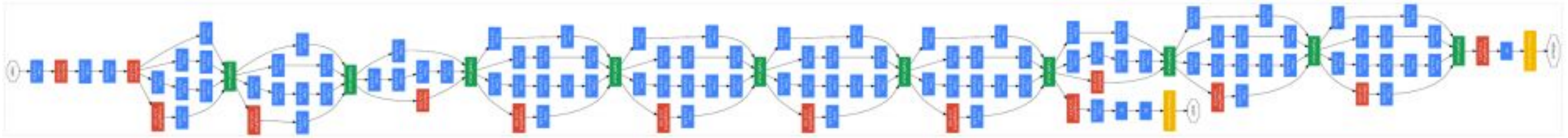
Even Deeper …



Jou (2019)



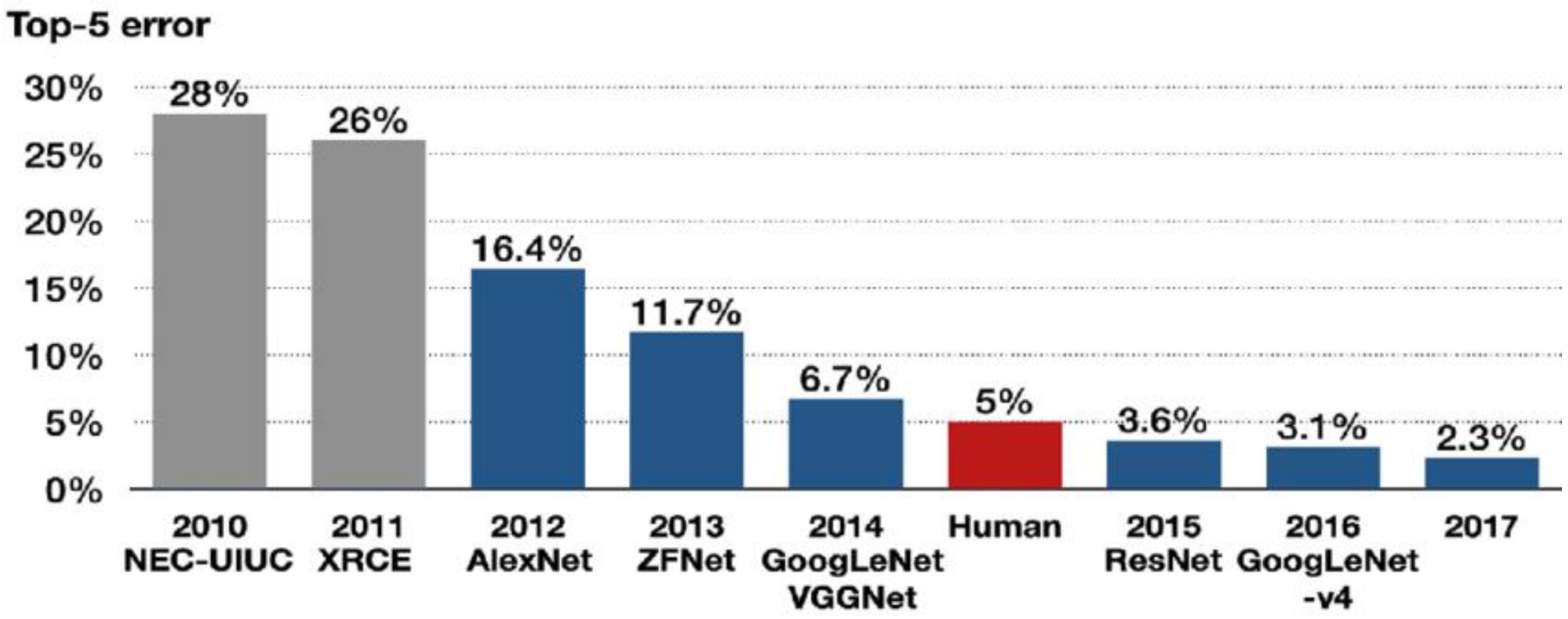
[1]Inception 5 (GoogLeNet)



Inception 7a

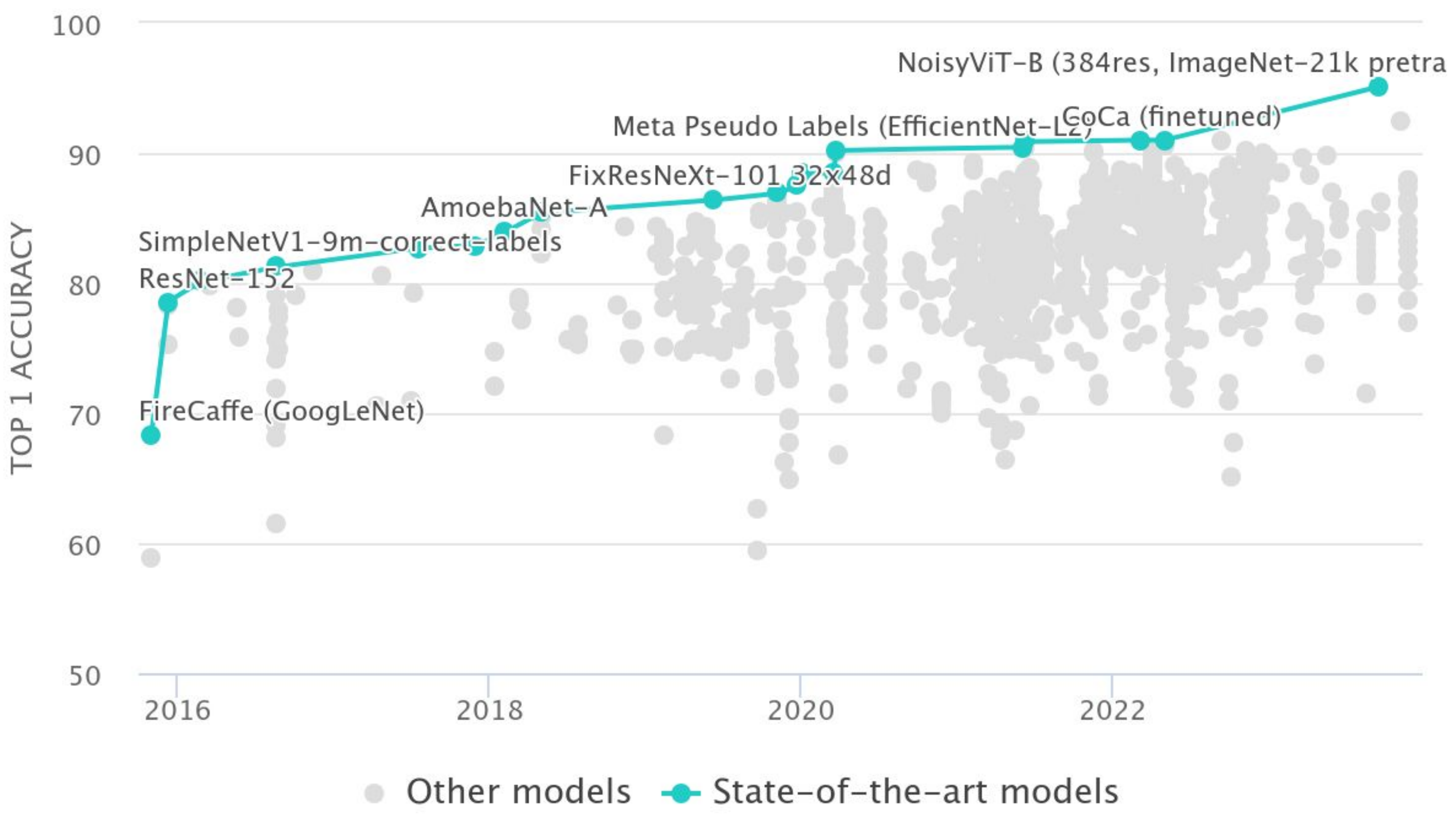
[1]Going Deeper with Convolutions, [C. Szegedy et al, CVPR 2015]

Imagenet Top-5 Error by Year



Alyafeai et al. (2019)

Imagenet Top-1 Accuracy



https://paperswithcode.com/sota/image-classification-on-imagenet

Single Images are Big

- E.g., 1024x768x3 variables
- 3D/4D data are even bigger
- The parameters necessary to detect a feature in the image are the same no matter where that feature is located
    - Edges, corners, basic shapes, eyes, cats…
- We can take advantage of this structure to simplify our models
    - Key tool: Convolutional Neural Networks (CNNs)

Dealing with Sequential Inputs

- Timeseries data
- Language / written text
- Biological “strings” (DNA/RNA/amino acids)

Dealing with Sequential Inputs

- Locality of items in the sequence is often important
  - Groups of nucleotides ultimately encode a single amino acid
  - A sequence of words forms a phrase
- But: how we process one part of a sequence could be very similar to how we handle other parts of the sequence
- CNNs and Recurrent NNs attempt to capitalize on this structure

Transformers & Generative Models

- Large Language Models: Given some context, generate a sequence of words as output
  - Context + what has already been generated -> compute a probability distribution over the next possible word
  - Sample a word from this distribution
  - Repeat
- ChatGPT, …

Agents Acting in the World

Given the current state of the world (or an agent’s view of it), what is the next best action to take?

- How do we encode state/situation?
- How do we measure “best”?

Agents Acting in the World

Measuring the outcome of an action:

- Sometimes, we know immediately what the outcome is and can evaluate this outcome
- For many interesting problems, the outcome is dependent on a long history of actions
- This requires an agent to attempt many different sequences of actions to infer what the “best” is

Agents Acting in the World

worldchesspieces.com

analyticsindiamag.com

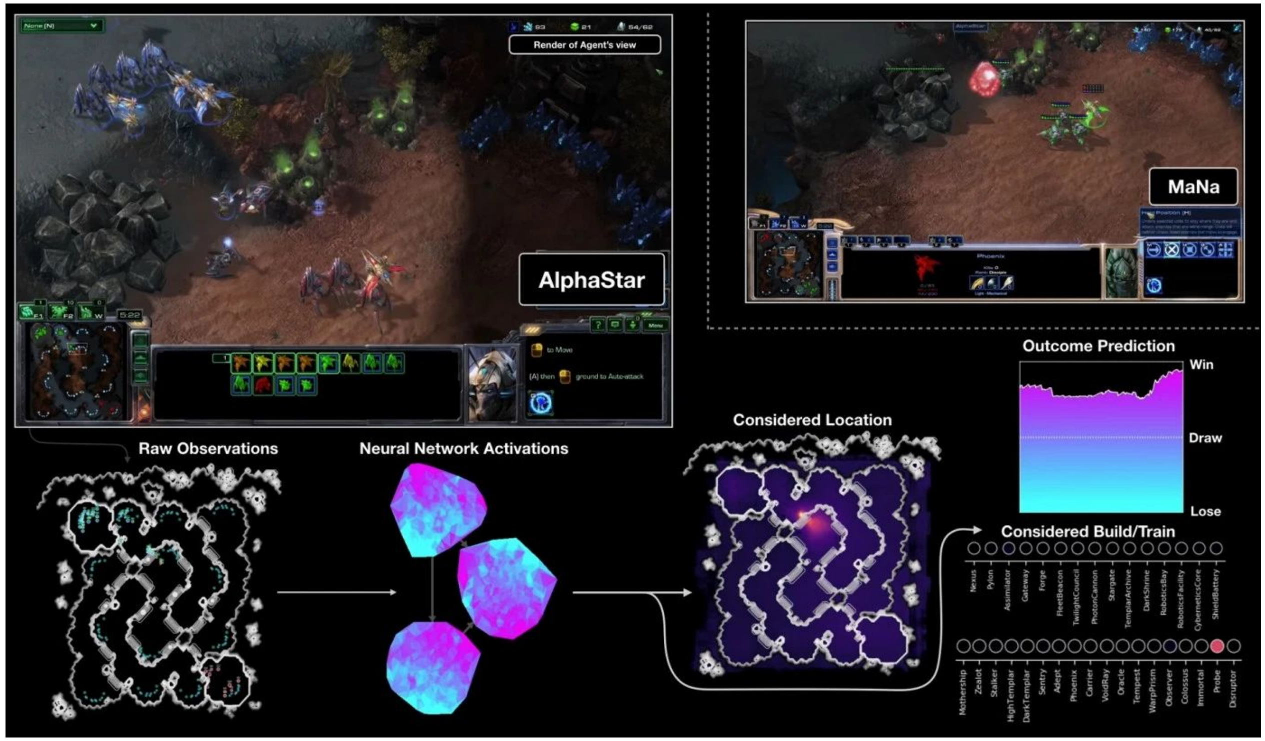
Agents Acting in the World

- Taking the history of states and actions into account in making action decisions is just as complex (perhaps more so) as taking all of the pixels into account in an image classification problem
- Getting a handle on this from a learning perspective requires:
  - Lots of data
  - Plenty of abstraction

Agents Acting in the World

Combining image processing with action: even more complex problem

ten Pas (2018)



techcrunch.com

(Some) Challenges of Deep Learning

- Having enough data
  - And being able to store it!
- Having the right data
  - Sampling from the true distribution
  - Stationarity of the underlying distribution
- Vanishing/exploding gradient problems
- Being able to explain what a learned model is doing

Deep Learning Progress has Accelerated

Confluence of:

- Availability of **a lot** of data
- Computational and data handling hardware
- Easy-to-use computational tools (e.g., python, Tensorflow, Keras, Pytorch, Jax…)
  - Automatic computation of gradients!
- Key algorithmic insights, e.g.:
  - Addressing vanishing gradient issues
  - Parameter sharing
  - Transfer learning
  - Transformers

Our Topics

- Backpropagation
- Model Evaluation Process: metrics, cross-validation, statistics, addressing the multiple comparisons problem
- Tools: TensorFlow, Keras, Weights and Biases
- Supercomputer use (SLURM), GPUs, efficient data handling
- Convolutional Neural Networks
- Recurrent Neural Networks
- Transformers
- Generative models: GANs and diffusion models
- Semantic segmentation
- Probabilistic Neural Networks

What I am assuming about you…

- Statistics and hypothesis testing
- Linear algebra and calculus
- Experience with machine learning
  - Including some multi-layer neural networks and backpropagation
- Programming skills
- Able to jump into Python, including the “Object-Orientedness” of it
- Know or can learn Unix command-line tools (very quickly)

Resources

- Course web page: http://symbiotic-computing.org/fagg_html/classes/aml
- Text: Simon J.D. Prince (2023) Understanding Deep Learning, ISBN-13: 978-0262048644, MIT Press
- Web resources: documentation, tutorials, papers (linked from the schedule or announced on Canvas)

Communication Resources

- Email
- GatherTown (invite link is on the class syllabus page)
    - Office hours and other meetings (look for the “Machine Learning” room)
- Slack (invite link is on the class syllabus)

Computing Environment

Setting up a ML environment can be a challenge…

- Tools you need: Python 3.12, Tensorflow 2.16, graphviz, Keras. And -Jupyter is helpful, too
  - Getting your own GPU set up can be a bear
  - Jupyter on Schooner (see the main class web page for instructions)

- We are setting up access to OSCER (OU supercomputer), which will have all of these configured for you
  - Common storage of data and code
  - Ability to run many compute jobs at once

- Weights and Biases (http://wandb.ai): exp visualization and tracking

Grading

- Homework assignments (9): 85%
- In-class exercises: 15%

Homework

- Supercomputer & Keras basics
- Deep networks
- Regularization and hyper-parameter searches
- Convolutional neural networks
- Semantic Segmentation
- Recurrent neural networks & Transformers
- Diffusion networks
- Probabilistic neural networks

Proper Academic Conduct

Homework assignments are to be done on your own

- No communication of large code solutions
- Do not copy code off the net
- Do not copy code from LLMs

Fair game (allowed):

- Any code that I release
- Using the net or LLMs to understand APIs or for small samples of code
- Sharing small bits of code with each other

Collaborative Note Taking

- Some material will be in slide form (I will post the slides)
- But: I will say lots around the slides and do a lot of work on the board
  - It is important for you to capture all of this information
- One possible approach: shared google drive with notes
  - One or two people are responsible for getting the key ideas down
  - Others can fill in details

For Next Time

- Next time: Neural Network basics
- For those needing help in getting into Python: there is a link from Canvas to a set of videos of mine that talk about the key features of Python (assuming prior programming experience)