Lecture 19: NN & Practical advices

Winter 2018

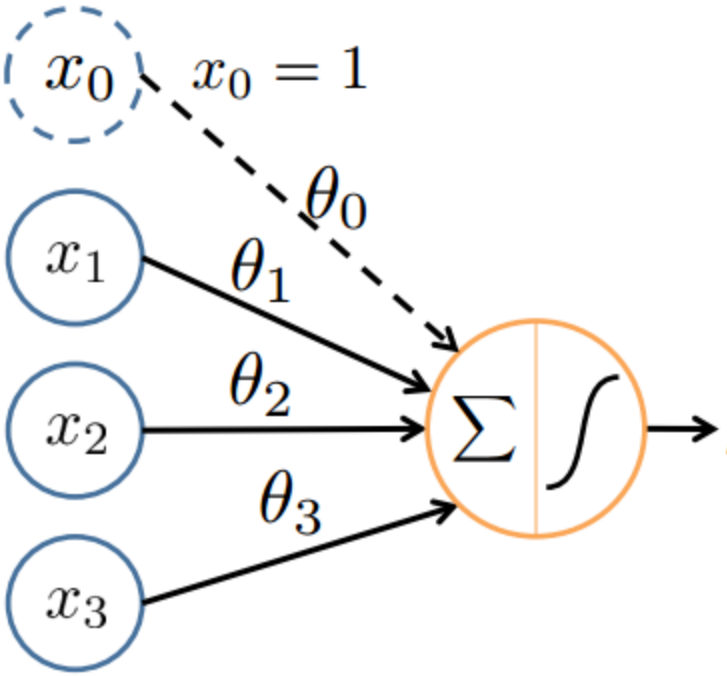
Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikuar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Neural Network

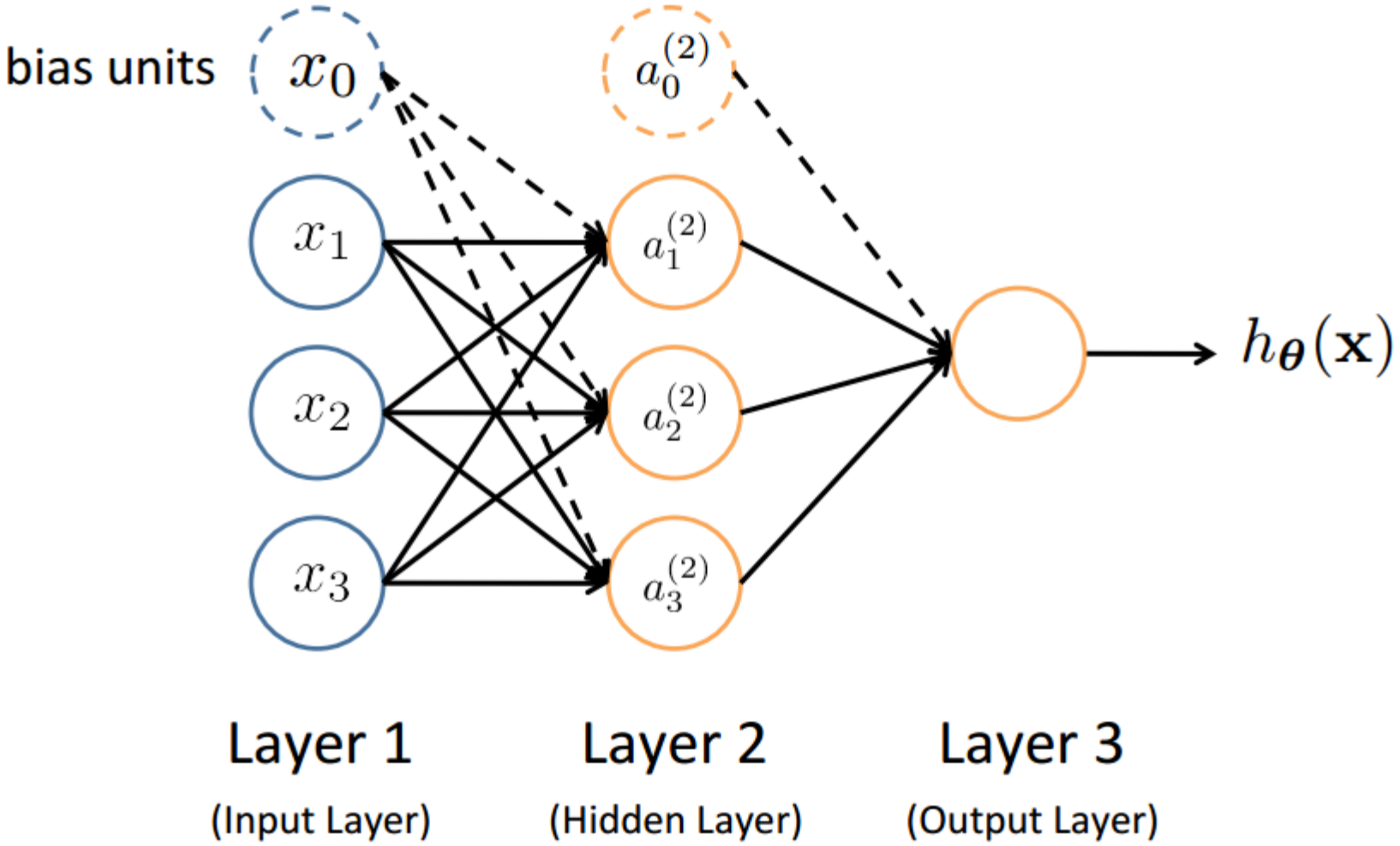


$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}\right) = \frac{1}{1 + e^{-\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}}}$$

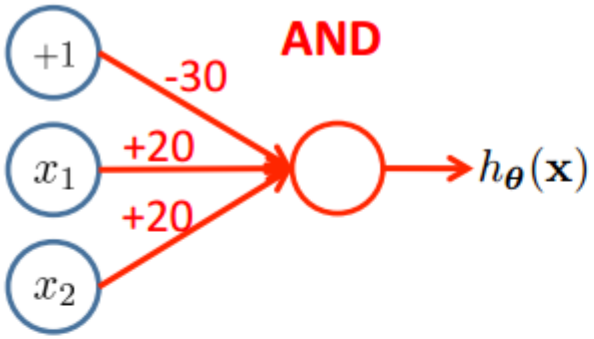
Sigmoid (logistic) activation function: $g(z) = \dfrac{1}{1 + e^{-z}}$

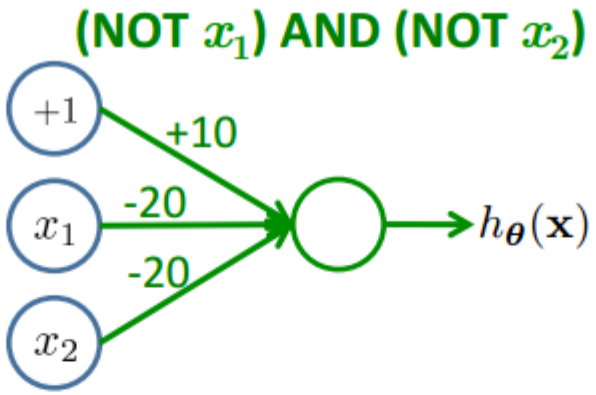
Neural Network (feed forward)

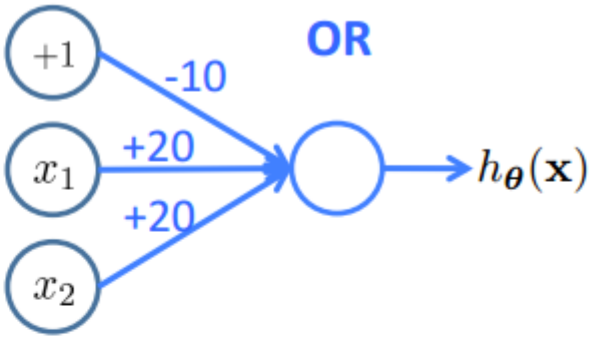


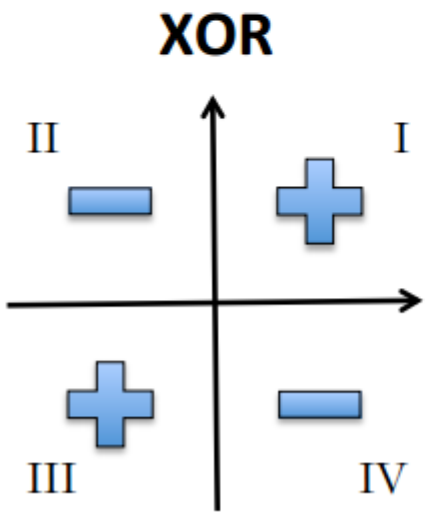
Slide by Andrew Ng

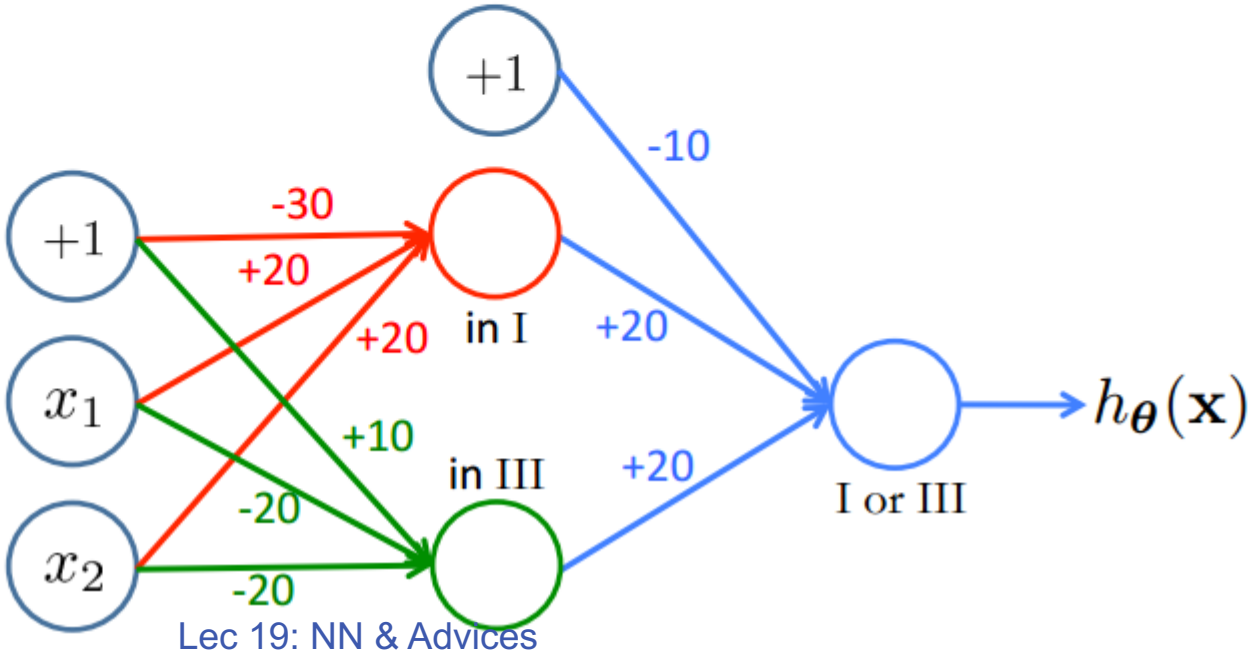
Combining Representations to Create Non-Linear Functions











Based on example by Andrew Ng

Learning as loss minimization

We have a classifier *NN* that is completely defined by its weights

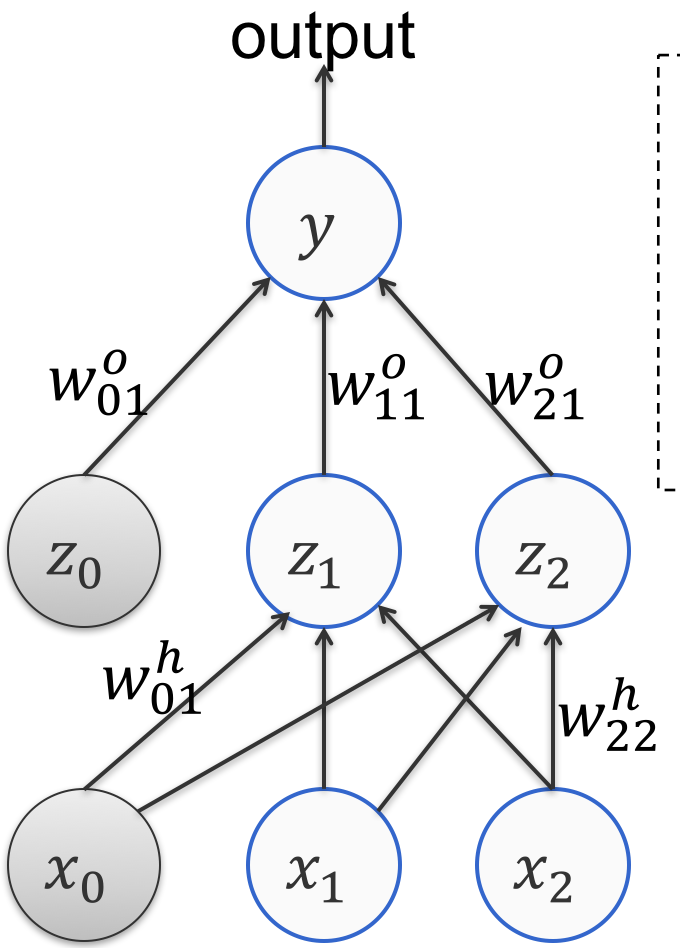
Learn the weights by minimizing a loss L

$$\min_{\boldsymbol{w}} \sum_{i} L(NN(x_i, w), y_i)$$

Perhaps with a *regularizer*

How do we solve the optimization problem?

Back to our running example



Given an input **x**, how is the output predicted

output $y = w_{01}^{o} + w_{11}^{o} z_1 + w_{21}^{o} z_2$

$$z_2 = \sigma(w_{02}^{h} + w_{12}^{h} x_1 + w_{22}^{h} x_2)$$

$$z_1 = \sigma(w_{01}^{h} + w_{11}^{h} x_1 + w_{21}^{h} x_2)$$

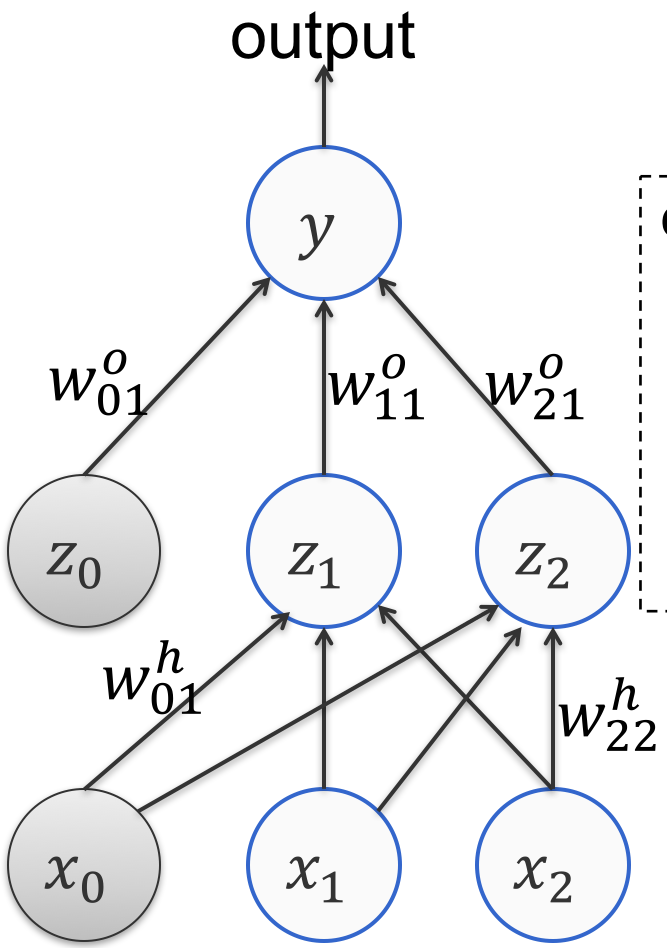
Suppose the true label for this example is a number y^*

We can write the *square loss* for this example as:

$$L = \frac{1}{2}(y - y^*)^2$$

Backpropagation

Applying the chain rule to compute the gradient (And remembering partial computations along the way to speed up things)



$$L = \frac{1}{2}(y - y^*)^2$$

output $y = w_{01}^o + w_{11}^o z_1 + w_{21}^o z_2$

$$z_2 = \sigma(w_{02}^h + w_{12}^h x_1 + w_{22}^h x_2)$$

$$z_1 = \sigma(w_{01}^h + w_{11}^h x_1 + w_{21}^h x_2)$$

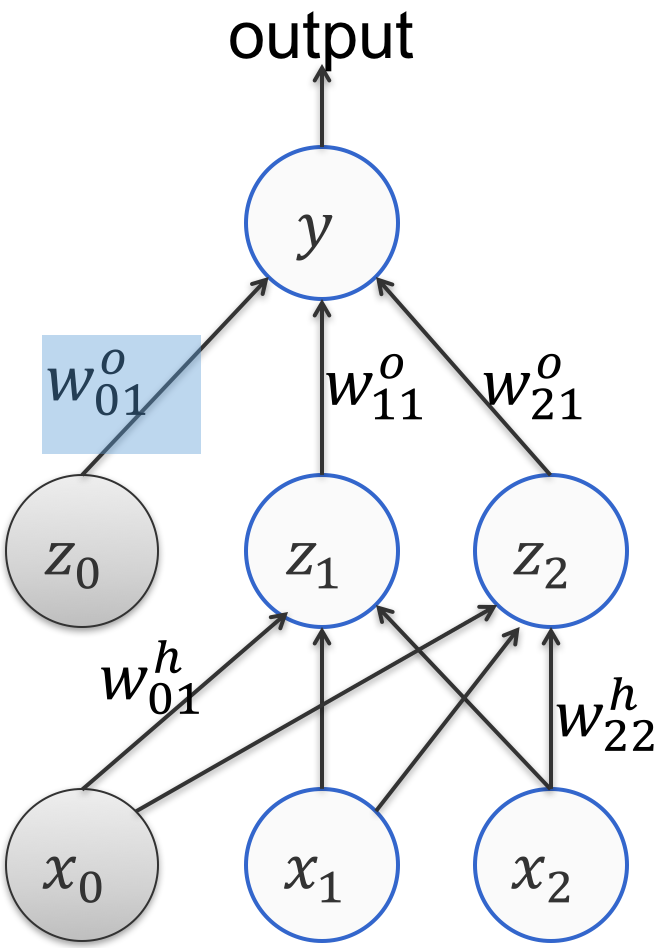
We want to compute

$$\frac{\partial L}{\partial w_{ij}^o} \text{ and } \frac{\partial L}{\partial w_{ij}^h}$$

$$L = \frac{1}{2}(y - y^*)^2$$

output $y = w_{01}^o + w_{11}^o z_1 + w_{21}^o z_2$

Output layer

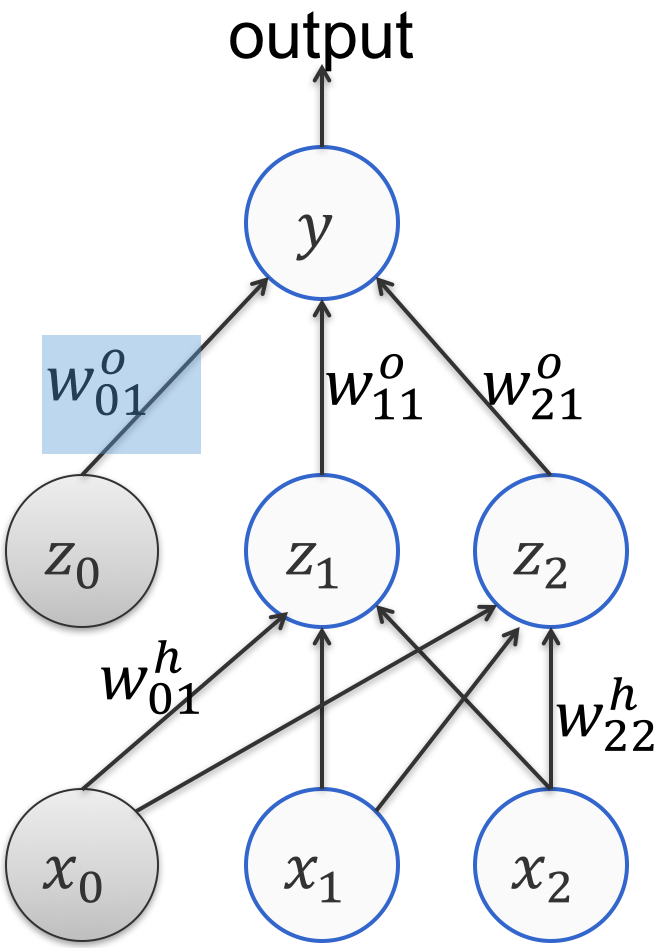


$$\frac{\partial L}{\partial w_{01}^o}$$

$$L = \frac{1}{2}(y - y^*)^2$$

output $y = w_{01}^o + w_{11}^o z_1 + w_{21}^o z_2$

Output layer

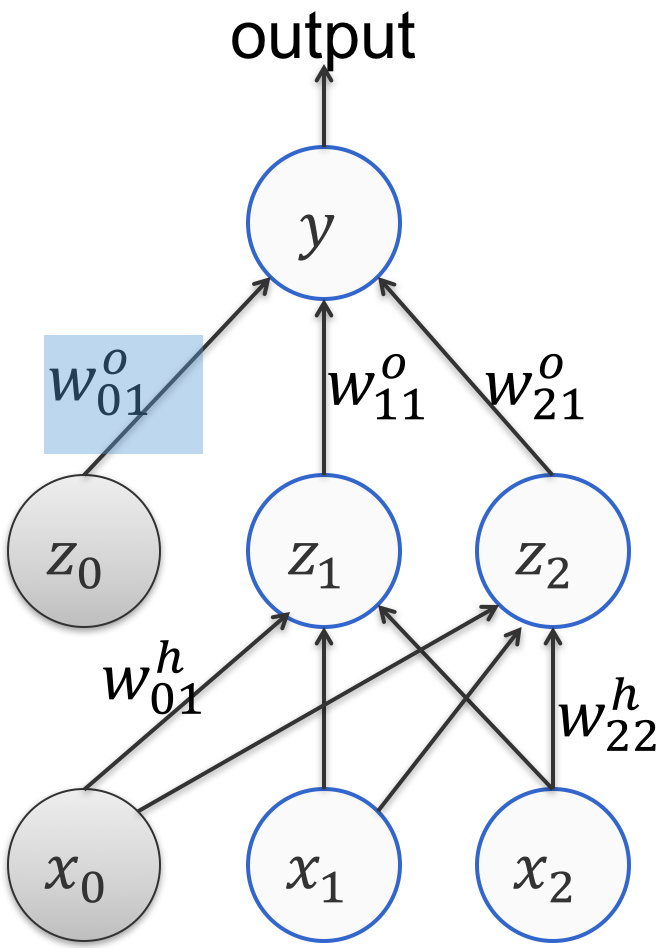


$$\frac{\partial L}{\partial w_{01}^o} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial w_{01}^o}$$

$$L = \frac{1}{2}(y - y^*)^2$$

output $y = w_{01}^o + w_{11}^o z_1 + w_{21}^o z_2$

Output layer



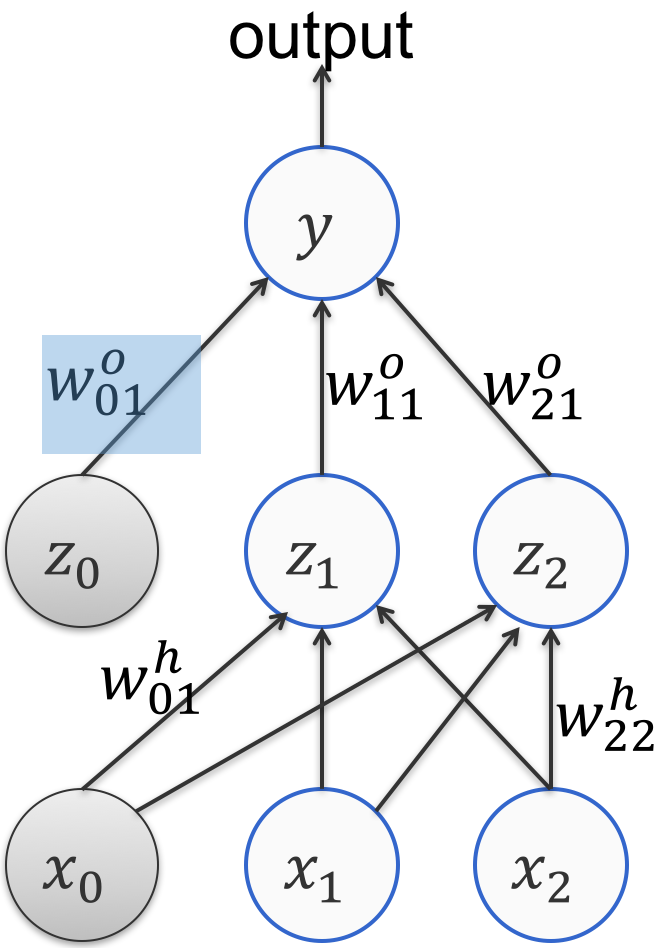
$$\frac{\partial L}{\partial w_{01}^o} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial w_{01}^o}$$

$$\frac{\partial L}{\partial y} = y - y^*$$

$$L = \frac{1}{2}(y - y^*)^2$$

output $y = w_{01}^o + w_{11}^o z_1 + w_{21}^o z_2$

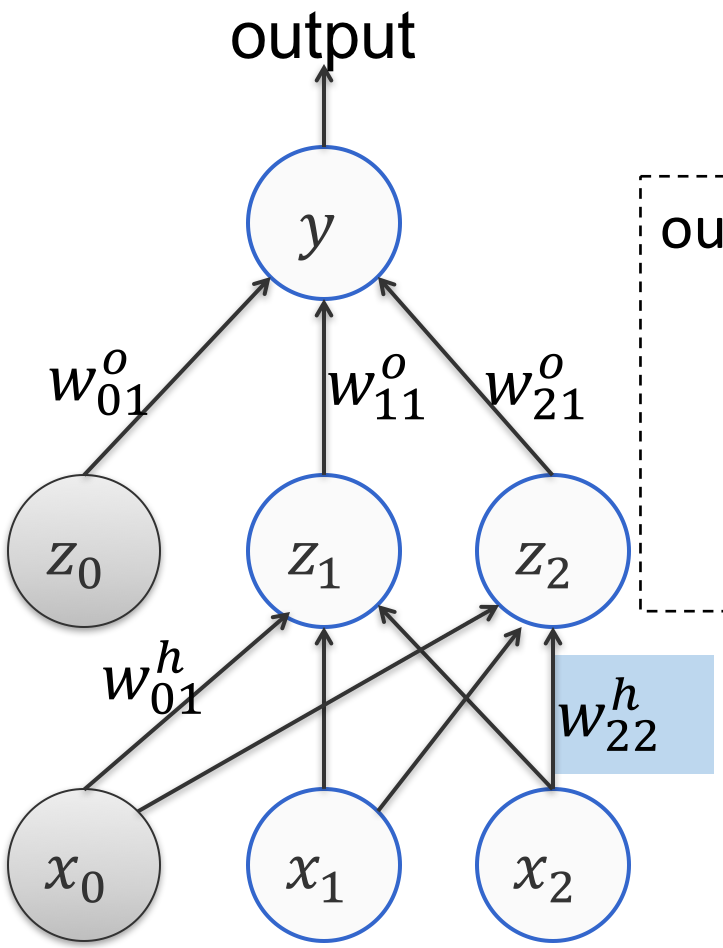
Output layer



$$\frac{\partial L}{\partial w_{01}^o} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial w_{01}^o}$$

$$\frac{\partial L}{\partial y} = y - y^* \qquad \frac{\partial y}{\partial w_{01}^o} = 1$$

Hidden layer derivatives



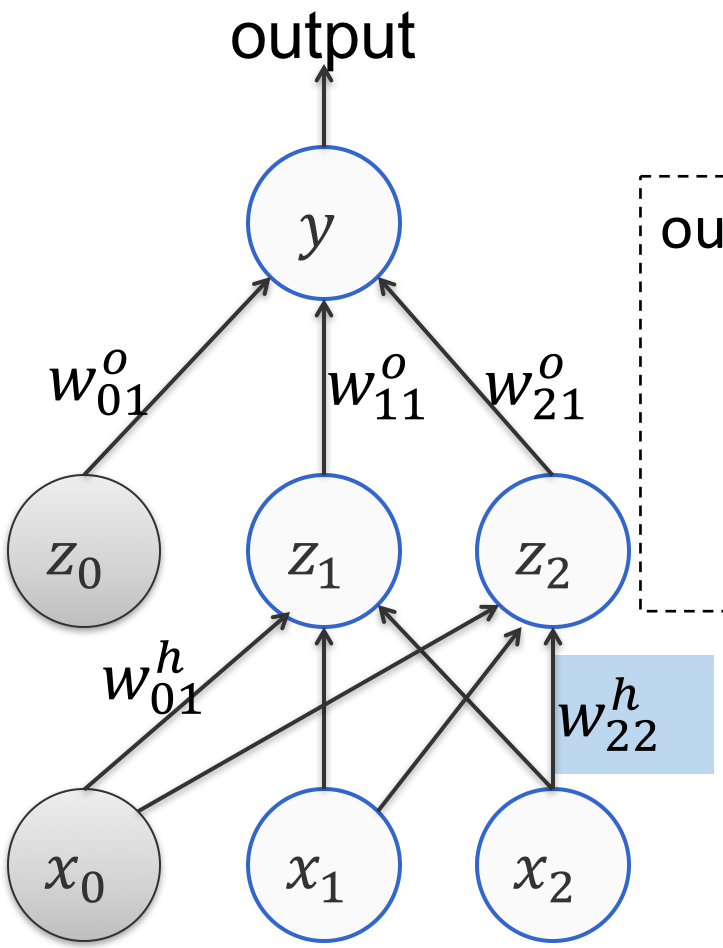
$$L = \frac{1}{2}(y - y^*)^2$$

output $y = w_{01}^{o} + w_{11}^{o} z_1 + w_{21}^{o} z_2$

$$z_2 = \sigma(w_{02}^{h} + w_{12}^{h} x_1 + w_{22}^{h} x_2)$$

$$z_1 = \sigma(w_{01}^{h} + w_{11}^{h} x_1 + w_{21}^{h} x_2)$$

Hidden layer derivatives



$$L = \frac{1}{2}(y - y^*)^2$$

output $y = w_{01}^{o} + w_{11}^{o} z_1 + w_{21}^{o} z_2$

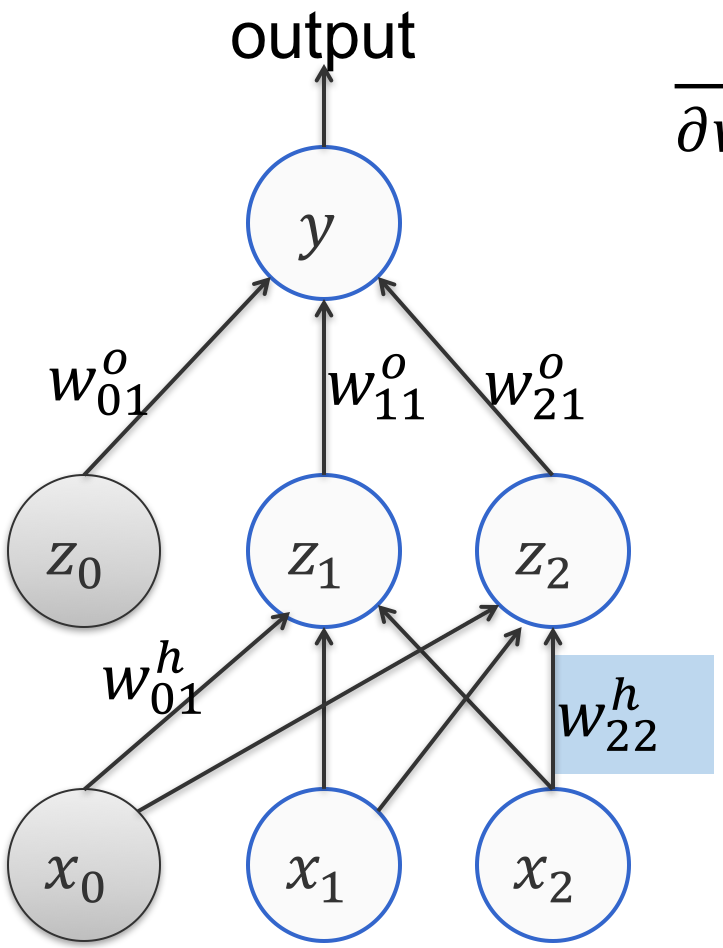
$$z_2 = \sigma(w_{02}^{h} + w_{12}^{h} x_1 + w_{22}^{h} x_2)$$

$$z_1 = \sigma(w_{01}^{h} + w_{11}^{h} x_1 + w_{21}^{h} x_2)$$

We want $\frac{\partial L}{\partial w_{22}^{h}}$

$$L = \frac{1}{2}(y - y^*)^2$$

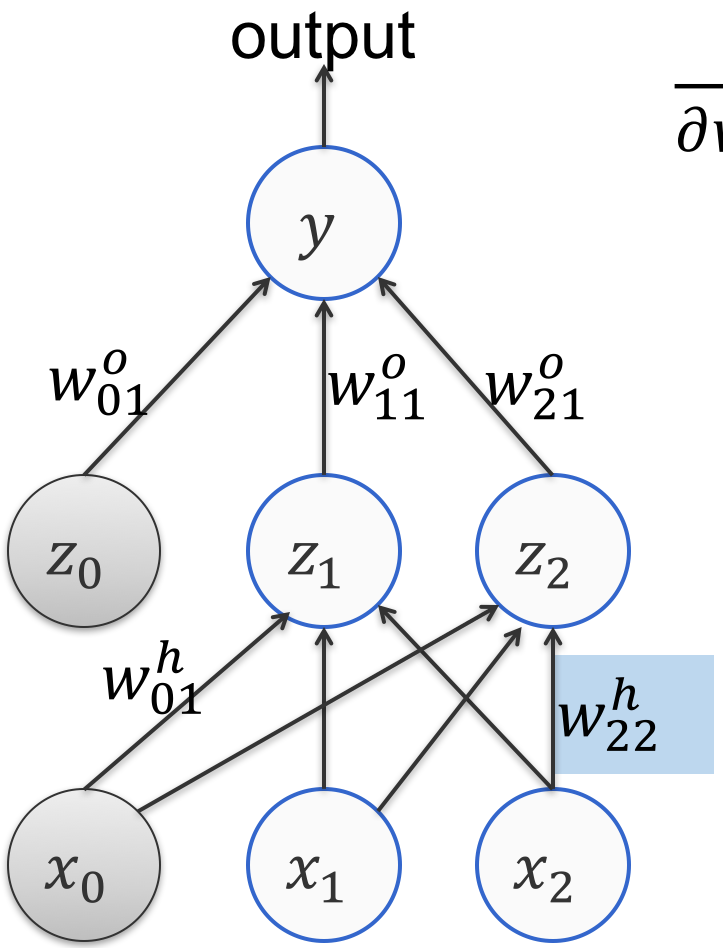
Hidden layer derivatives



$$\frac{\partial L}{\partial w_{22}^h} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial w_{22}^h}$$

$$y = w_{01}^{o} + w_{11}^{o} z_1 + w_{21}^{o} z_2$$

Hidden layer

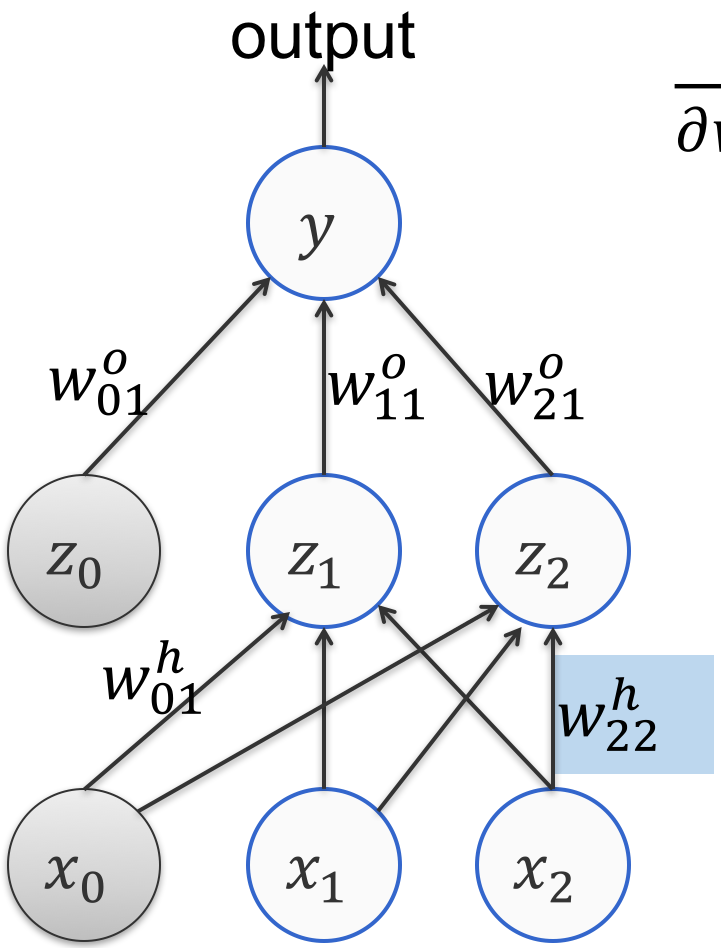


$$\frac{\partial L}{\partial w_{22}^{h}} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial w_{22}^{h}}$$

$$= \frac{\partial L}{\partial y}\frac{\partial}{\partial w_{22}^{h}}\left(w_{01}^{o} + w_{11}^{o} z_1 + w_{21}^{o} z_2\right)$$

$$y = w_{01}^{o} + w_{11}^{o} z_1 + w_{21}^{o} z_2$$

Hidden layer



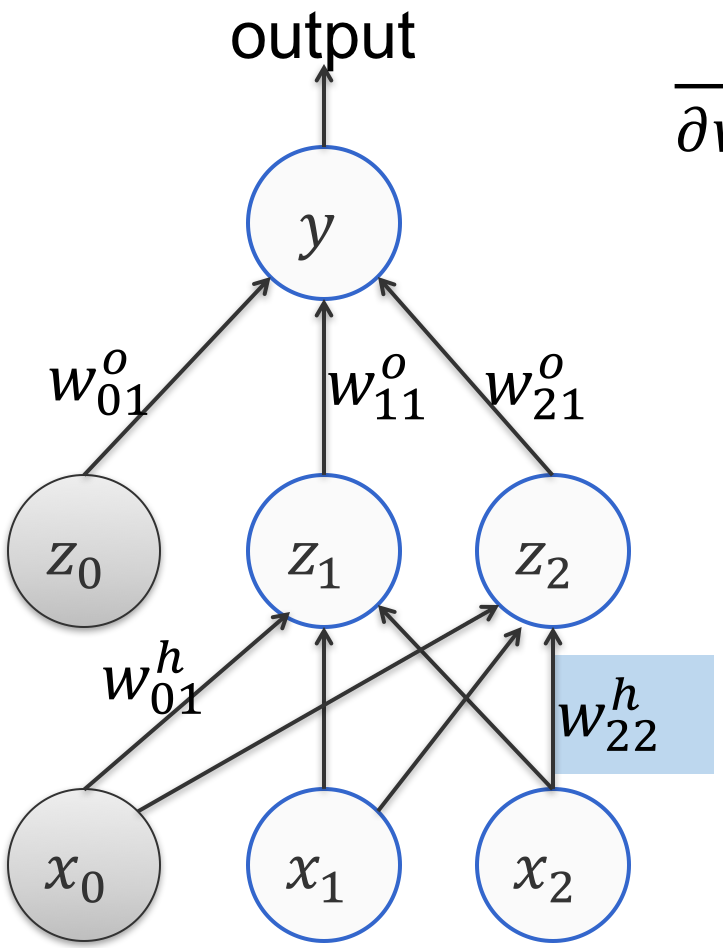
$$\frac{\partial L}{\partial w_{22}^{h}} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial w_{22}^{h}}$$

$$= \frac{\partial L}{\partial y}\frac{\partial}{\partial w_{22}^{h}}\left(w_{01}^{o} + w_{11}^{o} z_1 + w_{21}^{o} z_2\right)$$

$$= \frac{\partial L}{\partial y} w_{21}^{o} \frac{\partial z_2}{\partial w_{22}^{h}}$$

$$z_2 = \sigma(w_{02}^h + w_{12}^h x_1 + w_{22}^h x_2)$$

Hidden layer



$$\frac{\partial L}{\partial w_{22}^h} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial w_{22}^h}$$

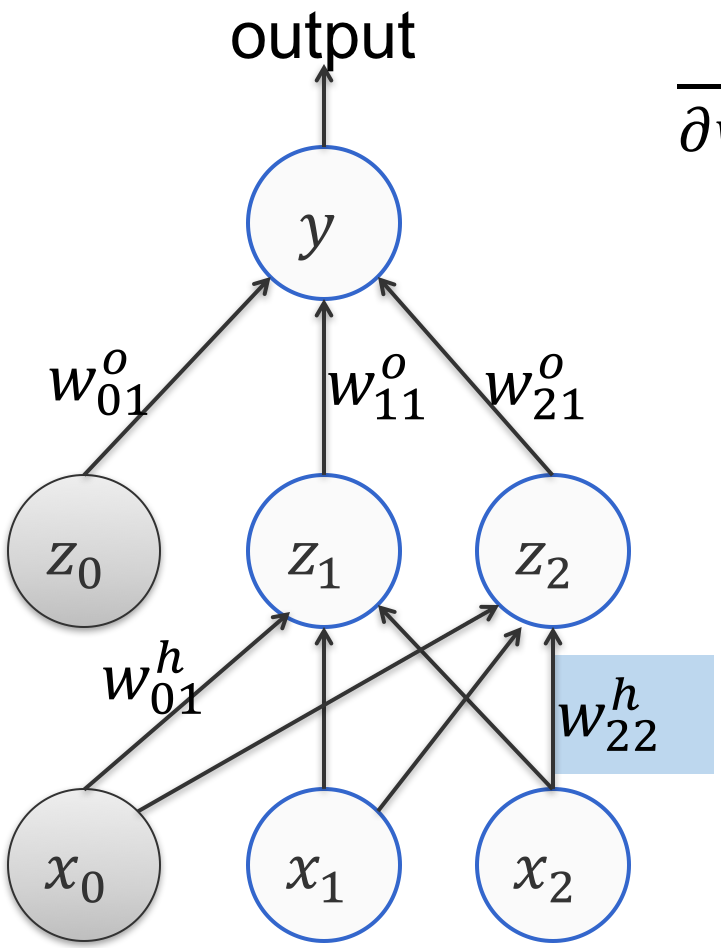
$$= \frac{\partial L}{\partial y}\frac{\partial}{\partial w_{22}^h}(w_{01}^o + w_{11}^o z_1 + w_{21}^o z_2)$$

$$= \frac{\partial L}{\partial y} w_{21}^o \frac{\partial z_2}{\partial w_{22}^h}$$

Hidden layer

$$z_2 = \sigma(w_{02}^h + w_{12}^h x_1 + w_{22}^h x_2)$$

Call this s

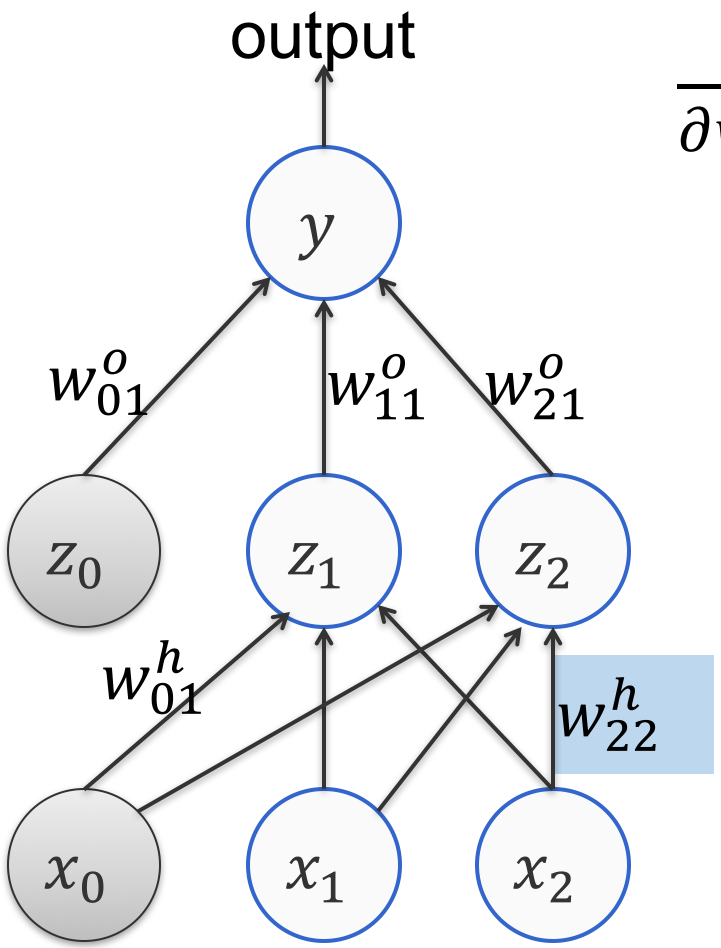


$$\frac{\partial L}{\partial w_{22}^h} = \frac{\partial L}{\partial y} w_{21}^o \frac{\partial z_2}{\partial s} \frac{\partial s}{\partial w_{22}^h}$$ (From previous slide)

$$z_2 = \sigma(w^h_{02} + w^h_{12}x_1 + w^h_{22}x_2)$$

Call this s

Hidden layer



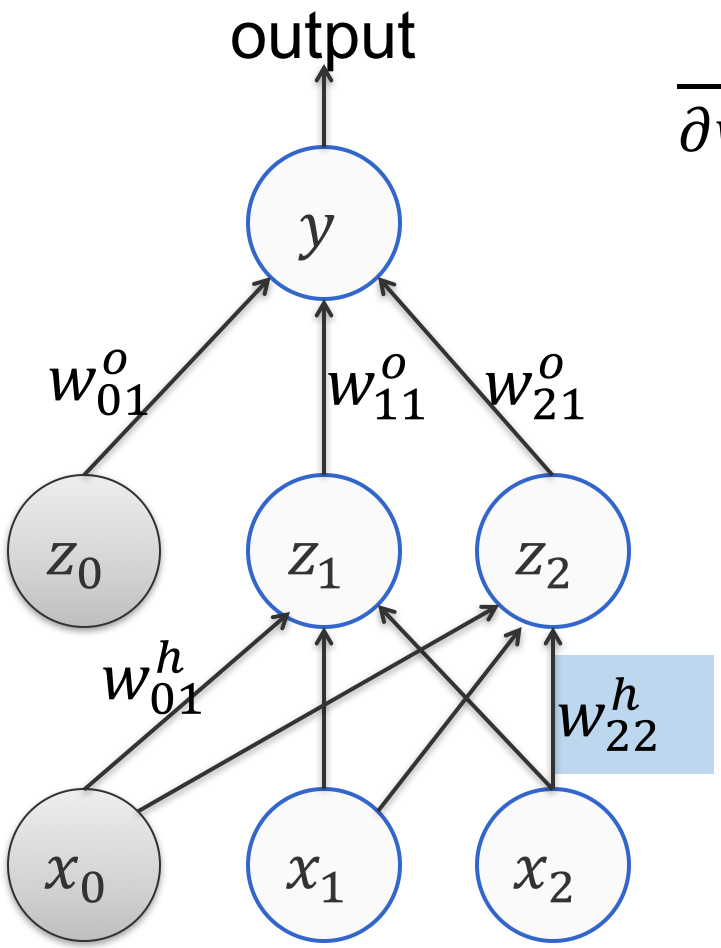
$$\frac{\partial L}{\partial w^h_{22}} = \frac{\partial L}{\partial y} w^o_{21} \frac{\partial z_2}{\partial s} \frac{\partial s}{\partial w^h_{22}}$$

Each of these partial derivatives is easy

$$z_2 = \sigma(w_{02}^h + w_{12}^h x_1 + w_{22}^h x_2)$$

Call this s

Hidden layer



$$\frac{\partial L}{\partial w_{22}^h} = \frac{\partial L}{\partial y} w_{21}^o \frac{\partial z_2}{\partial s} \frac{\partial s}{\partial w_{22}^h}$$

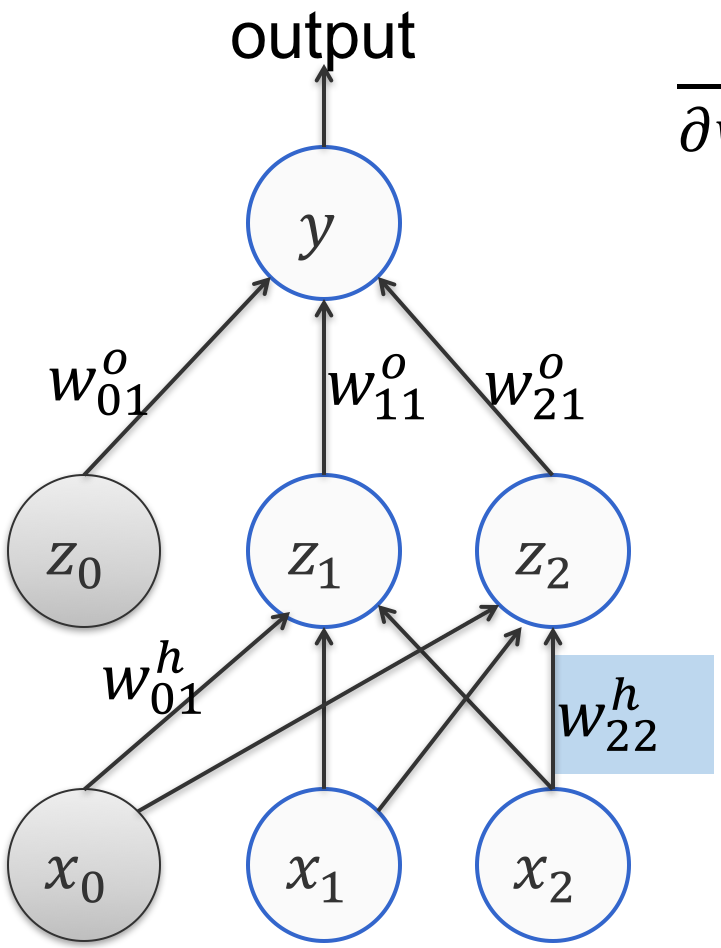
Each of these partial derivatives is easy

$$\frac{\partial L}{\partial y} = y - y^*$$

Hidden layer

$$z_2 = \sigma(w^h_{02} + w^h_{12}x_1 + w^h_{22}x_2)$$

Call this s



$$\frac{\partial L}{\partial w^h_{22}} = \frac{\partial L}{\partial y} w^o_{21} \frac{\partial z_2}{\partial s} \frac{\partial s}{\partial w^h_{22}}$$

Each of these partial derivatives is easy

$$\frac{\partial L}{\partial y} = y - y^*$$

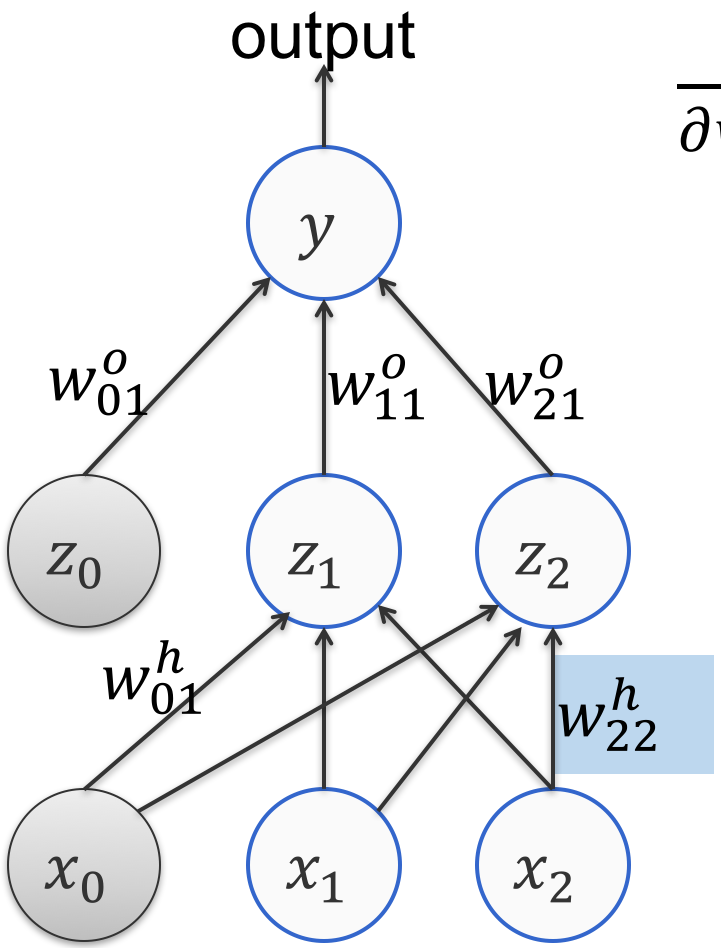
$$\frac{\partial z_2}{\partial s} = z_2(1 - z_2)$$

Why? Because $z_2(s)$ is the logistic function we have already seen

Hidden layer

$$z_2 = \sigma(w_{02}^h + w_{12}^h x_1 + w_{22}^h x_2)$$

Call this s



$$\frac{\partial L}{\partial w_{22}^h} = \frac{\partial L}{\partial y} w_{21}^o \frac{\partial z_2}{\partial s} \frac{\partial s}{\partial w_{22}^h}$$

Each of these partial derivatives is easy

$$\frac{\partial L}{\partial y} = y - y^*$$

$$\frac{\partial z_2}{\partial s} = z_2(1 - z_2)$$

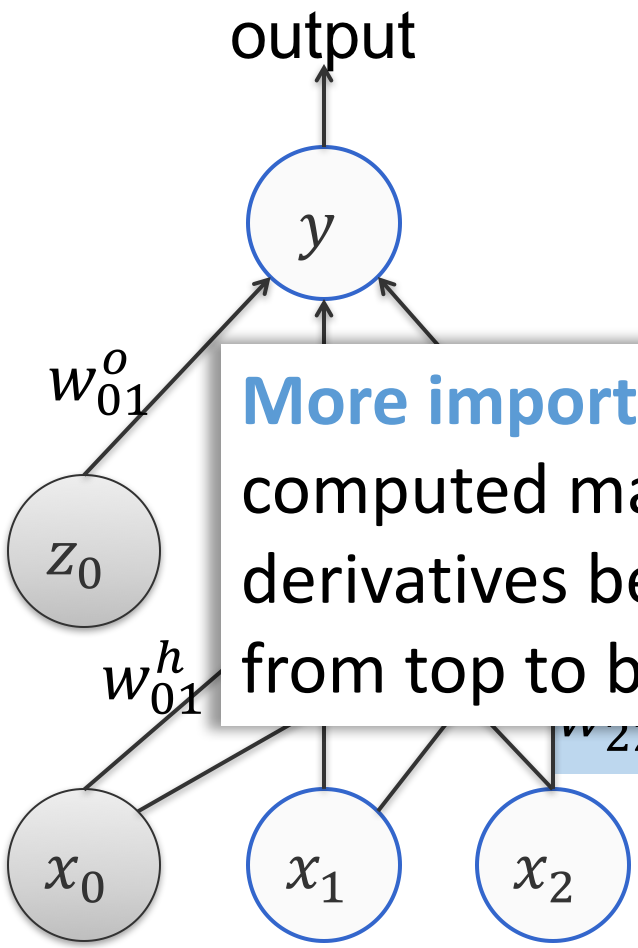
$$\frac{\partial s}{\partial w_{22}^h} = x_2$$

Why? Because $z_2(s)$ is the logistic function we have already seen

$$z_2 = \sigma(w_{02}^h + w_{12}^h x_1 + w_{22}^h x_2)$$

Call this s

Hidden layer



$$\frac{\partial L}{\partial w_{22}^h} = \frac{\partial L}{\partial y} w_{21}^o \frac{\partial z_2}{\partial s} \frac{\partial s}{\partial w_{22}^h}$$

Each of these partial derivatives is easy

More important: We have already computed many of these partial derivatives because we are proceeding from top to bottom (i.e. backwards)

cause $z_2(s)$
istic
we have
een

$$\frac{\partial s}{\partial w_{22}^h} = x_2$$

The Backpropagation Algorithm

The same algorithm works for multiple layers

Repeated application of the chain rule for partial derivatives

- First perform forward pass from inputs to the output
- Compute loss

- From the loss, proceed backwards to compute partial derivatives using the chain rule
- Cache partial derivatives as you compute them
  - Will be used for lower layers

Deep Learning Libraries

Tensorflow

Torch

Theano

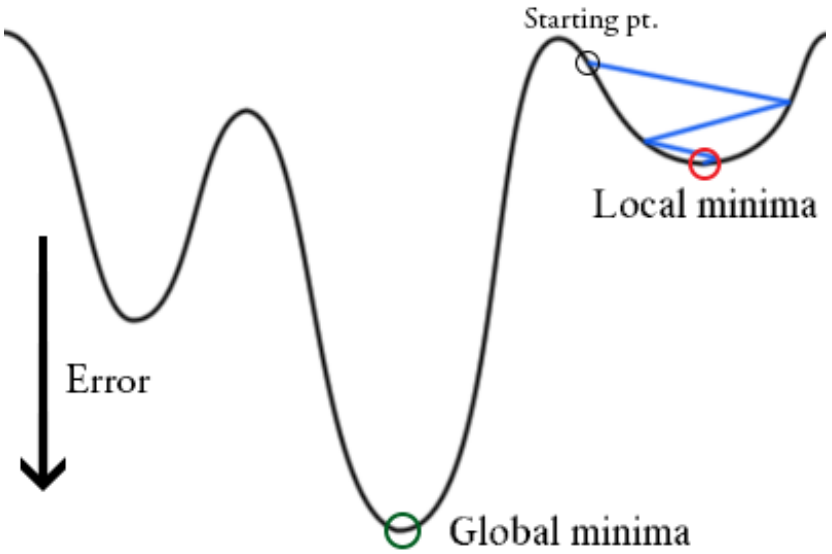
Pytorch

Dynet

They allow automatic differentiation. You can just write the network, get the gradients easily.

(stochastic) Gradient Descent

What happens when function is non-convex ?



Ohh no ! Algorithm gets stuck in local minima

Comments on Training

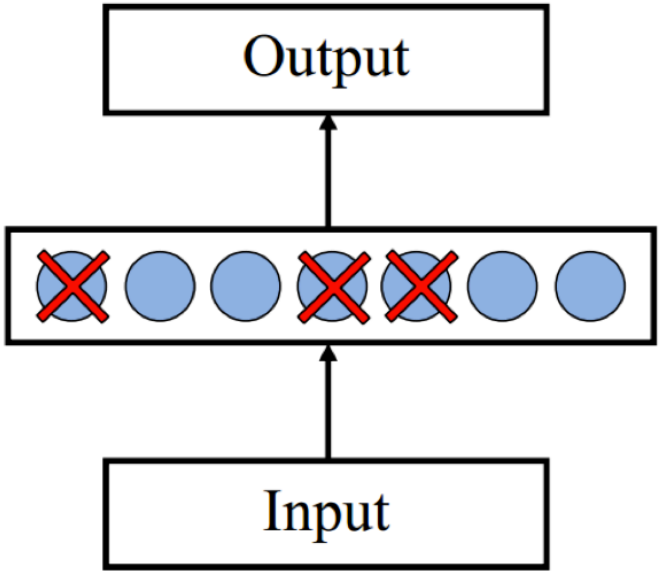
- No guarantee of convergence, may oscillate or reach a local minima
- In practice, many large networks can be trained on large amounts of data for realistic problems
- Termination criteria : Number of epochs; Threshold on training set error; No decrease in error; Increase error on a validation set
- To avoid local minima : several trials with different random initial weights with majority or voting techniques

Over-fitting Prevention

- Running too many epochs may over-train the network and result in overfitting.
- Keep a held out validation set and test accuracy after each epoch
- Randomly dropout some latent states and data

Dropout Training

Proposed by Hinton et al 2012



Each time, decide on whether to delete a hidden unit with some probability **p**

This lecture

- Neural Network
- Recurrent NN
- Convolutional NN

Motivation

Not all problems can be converted into one with fixed length inputs and outputs

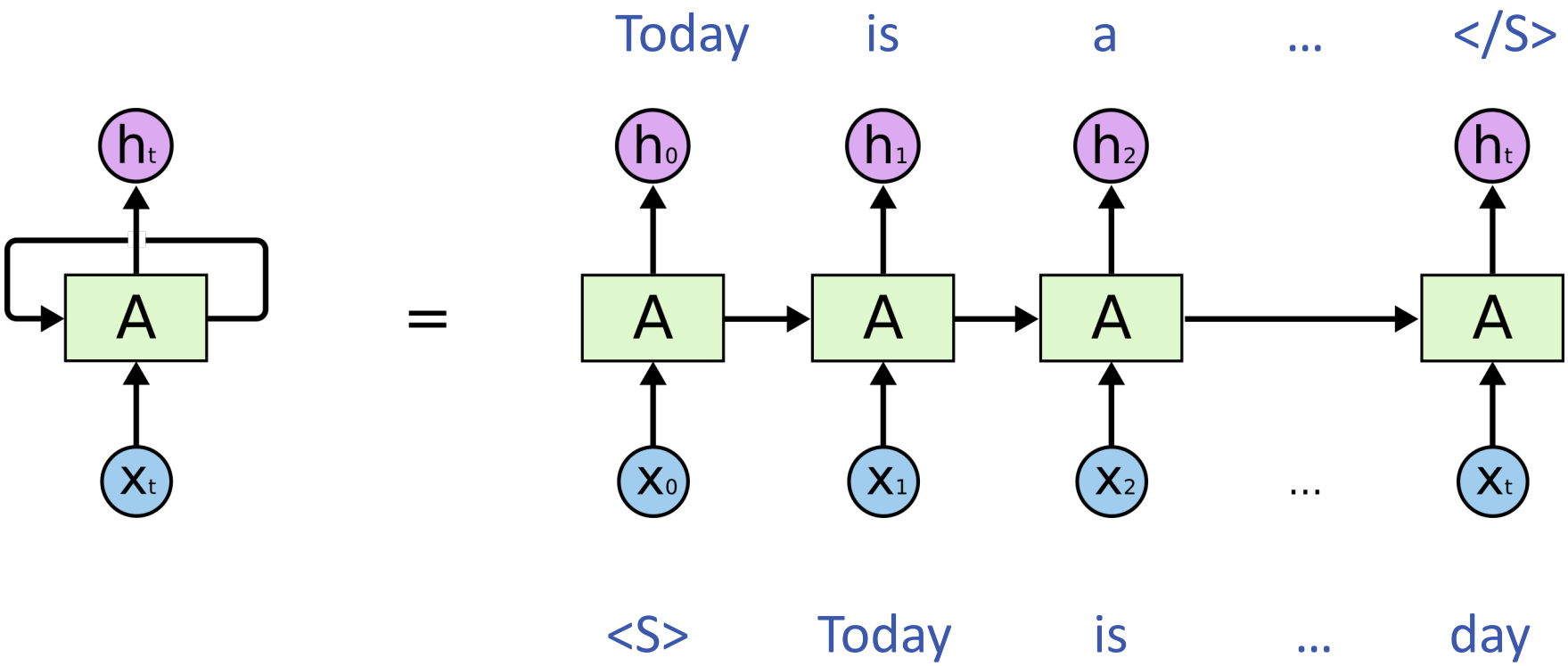
Machine Translation : translating English sentence to other language

Speech Recognition : translating spoken sentence to written sentence

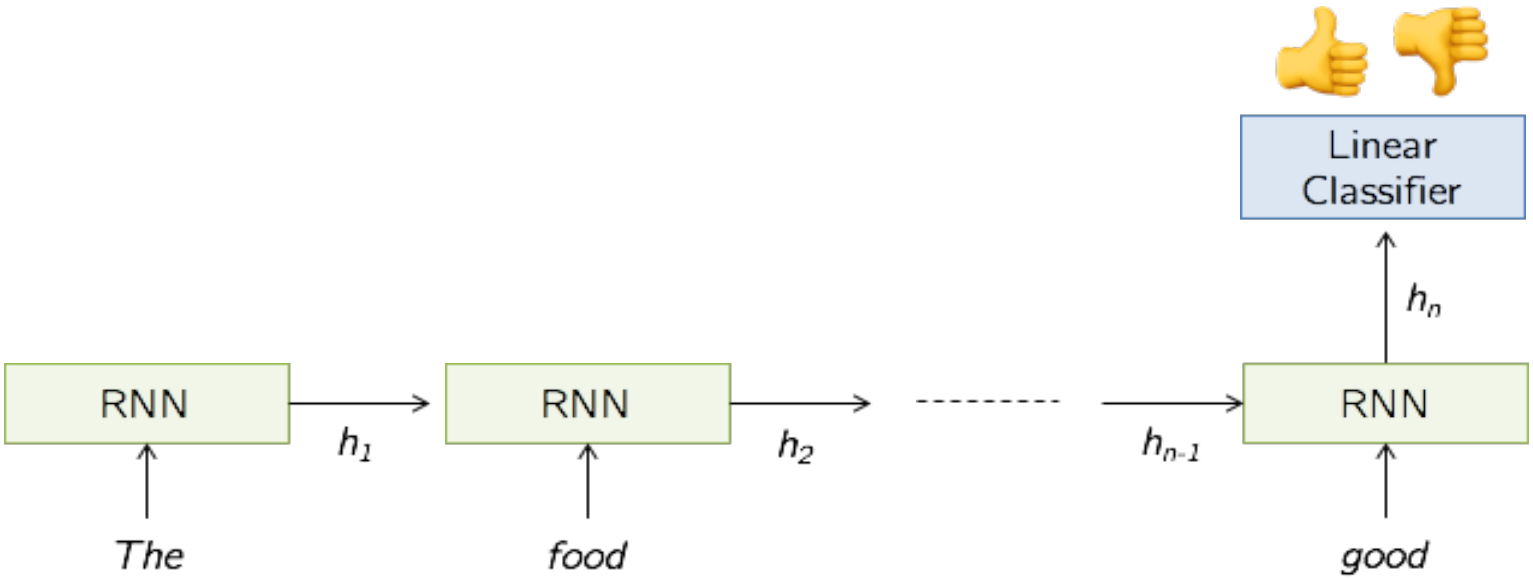
Hard or impossible to choose a large enough window, there can always be a new sentence longer than anything seen

How to deal with input with variant size?

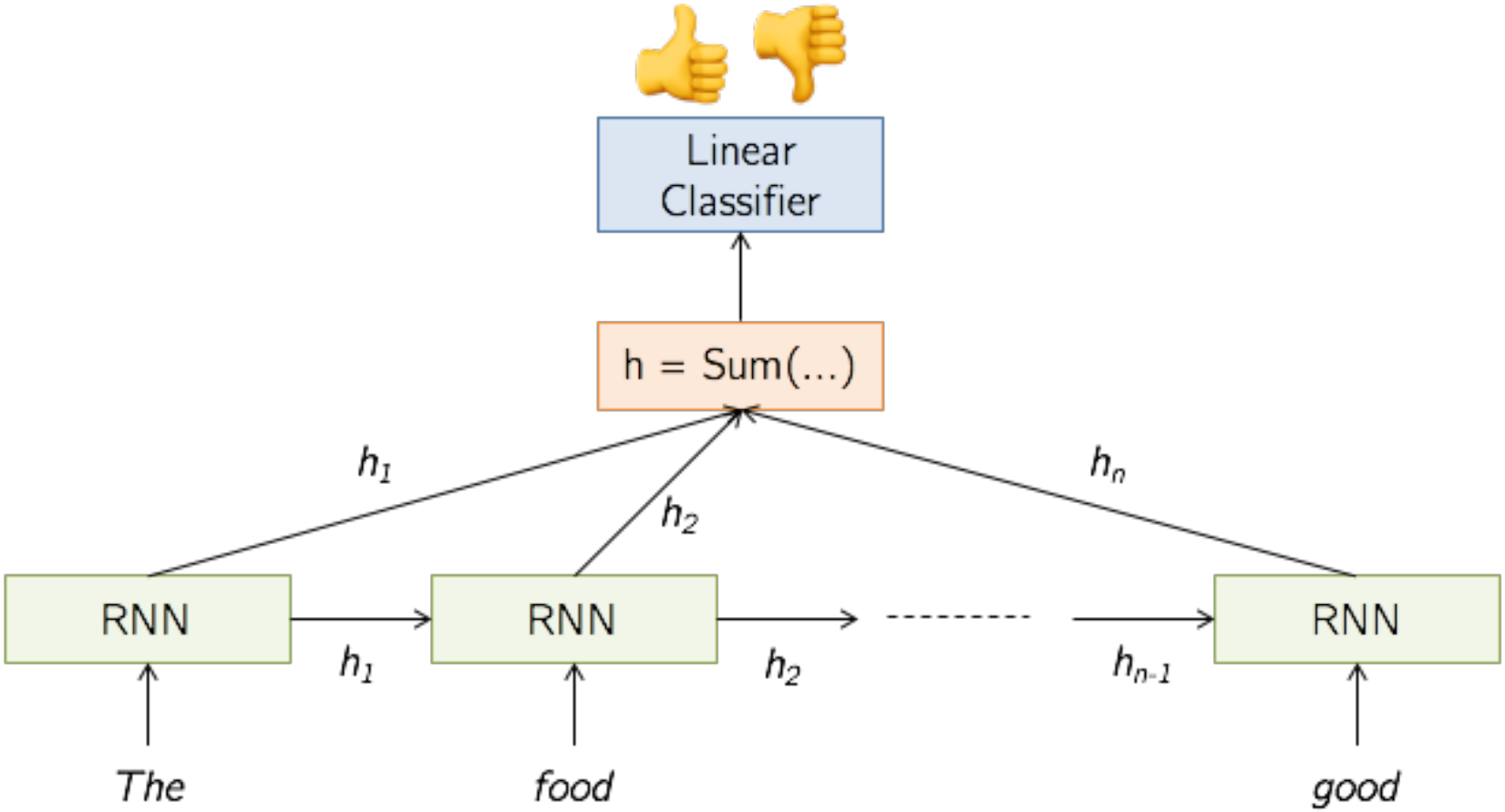
- Use same parameters



Sentiment Classification



Sentiment Classification



http://deeplearning.net/tutorial/lstm.html

Input Output Scenario

Single - Single

Feed-forward network

Single - Multiple

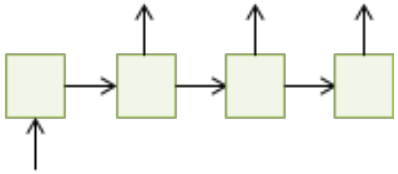
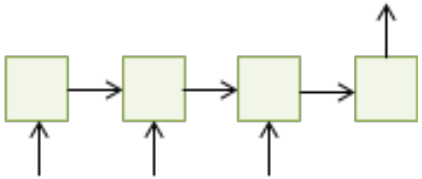


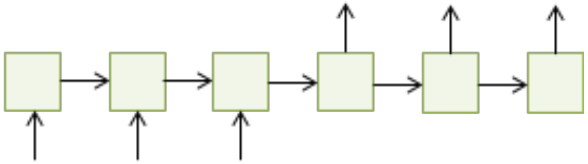
Image captioning

Multiple - Single



Sentiment classification

Multiple - Multiple



Translation

This lecture

- Neural Network
- Recurrent NN
- Convolutional NN

Motivation

Grayscale Image is rectangular matrix of pixel values (intensity values)

How to remove the noise in the photograph ?

This picture and many others taken from slides of Prof. Lana Lazebnik

Moving Average

- Lets replace each pixel with a weighted average of its neighborhood
- The weights are called the filter kernel
- What are the weights for the average of a 3 * 3 neighborhood ?

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

"box filter"

Practice with Linear Filters

Original

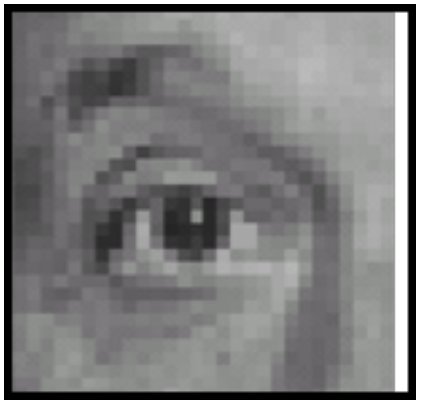
0	0	0
0	1	0
0	0	0

Filtered (No change)

Practice with Linear Filters

Original

0	0	0
0	0	1
0	0	0

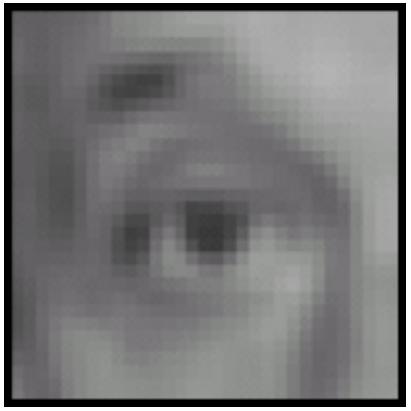


Shifted left by 1 pixel

Practice with Linear Filters

Original

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



Blur (with a box filter)

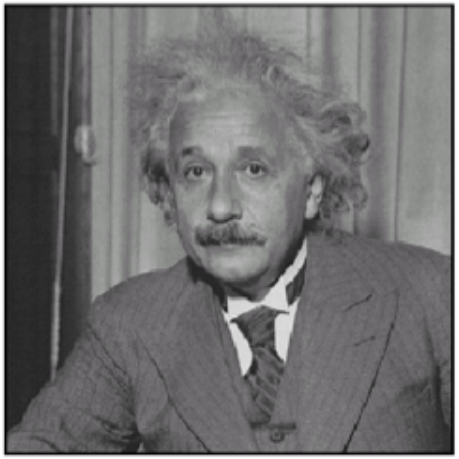
Practice with Linear Filters

Original

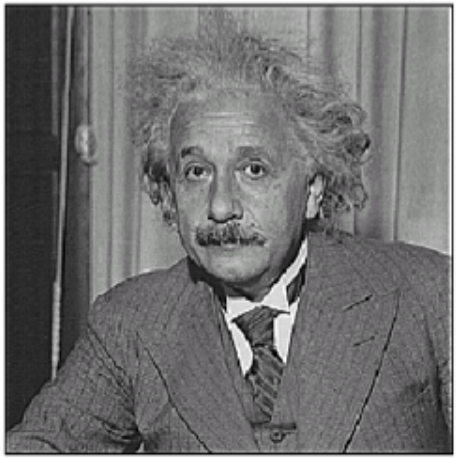
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sharpening filter - increases differences with local average

Sharpening Filter



before



after

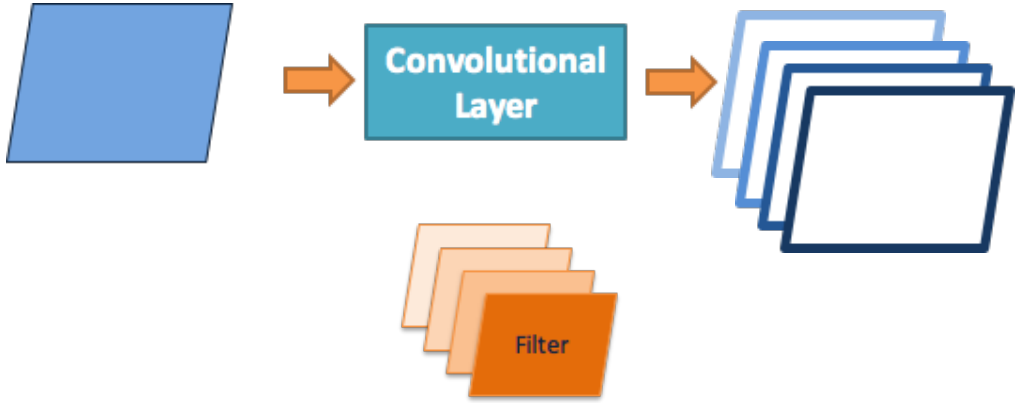
Filters can be used for sharpening, edge detection and extracting many other properties. See https://en.wikipedia.org/wiki/Kernel_(image_processing) for more examples.

Main idea behind Convolutional Neural Networks: **use learned filters**

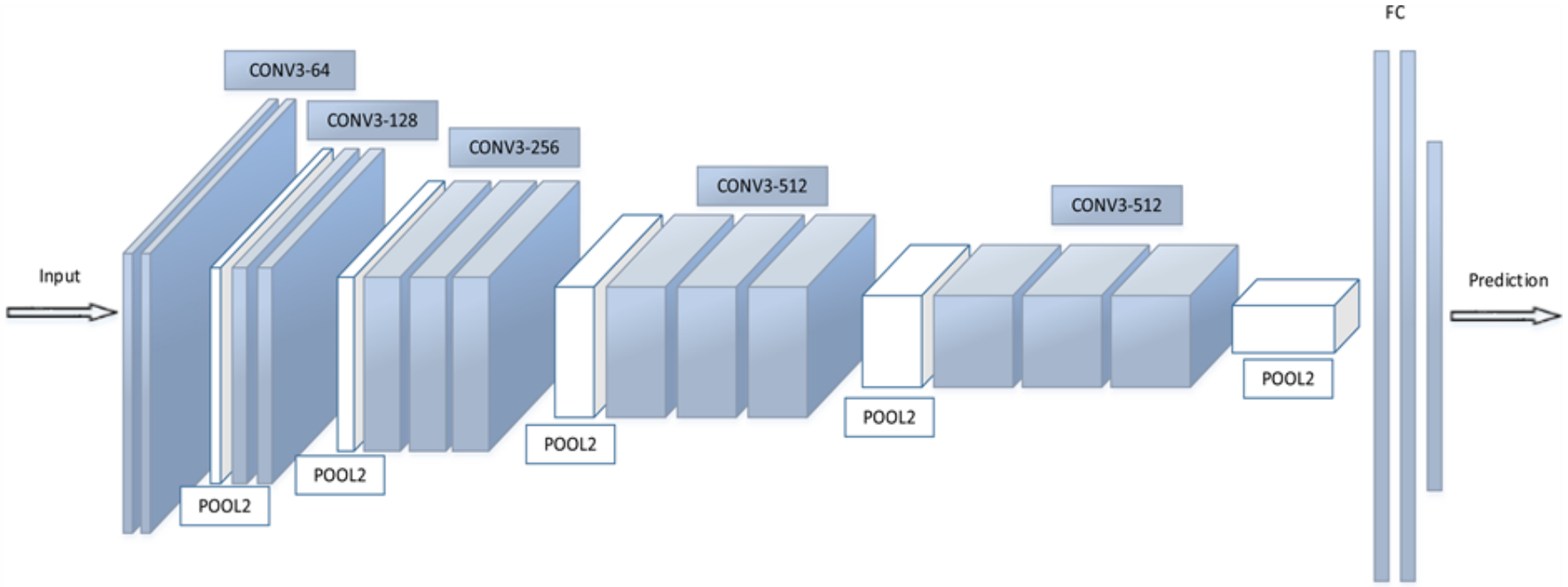
Convolution Layer

You can also use multiple filters on your image, all of them will be learnt

You can add non-linearity at the output of a convolutional layer



An example: VGG

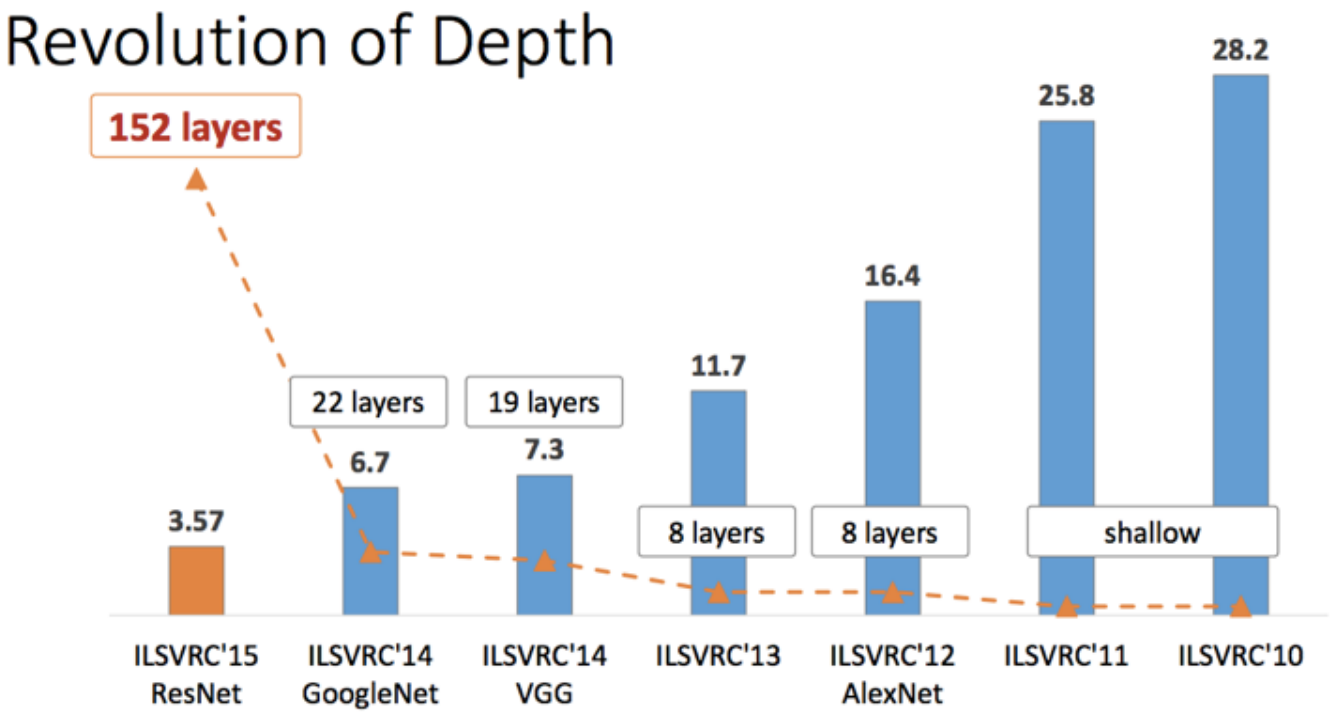


Thickness of each block corresponds to number of channels of the response / output. Height and width correspond to height and width of response / output.

Think of this as stacking several feature extractors, higher stages compute more global, more invariant features

138 million parameters

Object Detection Performance



ImageNet Challenge Top 5 error

Review & Practical advices

ML and the world

- Bias vs Variance
- Diagnostics of your learning algorithm
- Error analysis
- Injecting machine learning into *Your Favorite Task*

Making ML work in the world

Mostly experiential advice

Also based on what other people have said

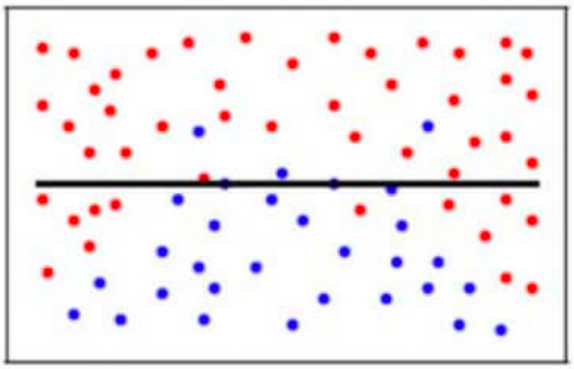
Bias and variance

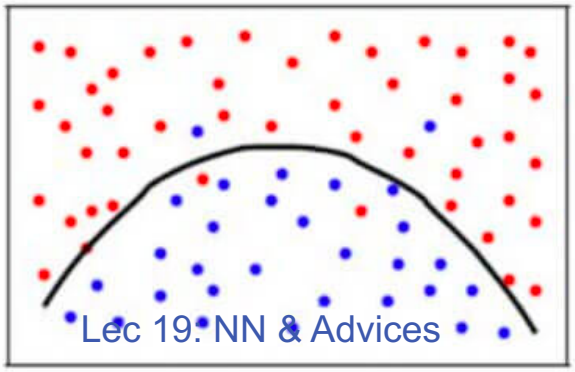
Every learning algorithm requires assumptions about the hypothesis space.

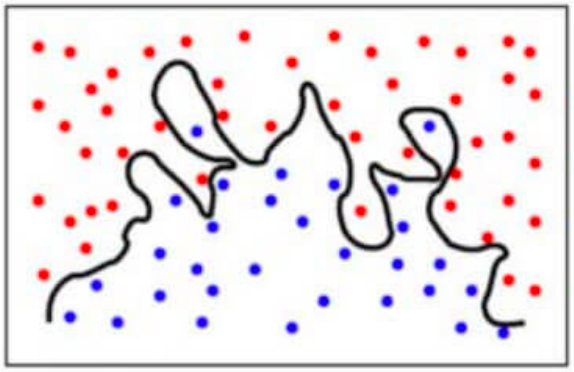
Eg: “My hypothesis space is

- …linear”
- …decision trees with 5 nodes”
- …deep neural network with 12 layers”

Underfitting ⟷ Overfitting







Managing bias and variance

- Ensemble methods can reduce both bias and variance
  - Multiple classifiers are combined
  - Eg: Bagging, boosting
- Decision trees of a fixed depth
  - Increasing depth decreases bias, increases variance
- SVMs
  - Stronger regularization increases bias, decreases variance
- K nearest neighbors
  - Increasing k generally increases bias, reduces variance
- Neural Network
  - Early stopping, dropout

Tune your parameters!!

- Always tune parameters when comparing different settings / algorithms
- Never tune your parameters on the test set

ML and the world

- Bias vs Variance
- Diagnostics of your learning algorithm
- Error analysis
- Injecting machine learning into *Your Favorite Task*

Debugging machine learning

Suppose you train an SVM or a logistic regression classifier for spam detection

You *obviously* follow best practices for finding hyper-parameters (such as cross-validation)

Your classifier is only 75% accurate

What can you do to improve it?

Different ways to improve your model

More training data

Features

1. Use more features
2. Use fewer features
3. Use other features

Better training

1. Run for more iterations
2. Use a different algorithm
3. Use a different classifier
4. Play with regularization

Different ways to improve your model

More training data

Features

1. Use more features
2. Use fewer features
3. Use other features

Tedious!

And prone to errors, dependence on luck

Let us try to make this process more methodical

Better training

1. Run for more iterations
2. Use a different algorithm
3. Use a different classifier
4. Play with regularization

First, diagnostics

Easier to fix a problem if you know where it is

Some possible problems:

1. Over-fitting (high variance)
2. Under-fitting (high bias)
3. Your learning does not converge
4. Are you measuring the right thing?

Detecting over or under fitting

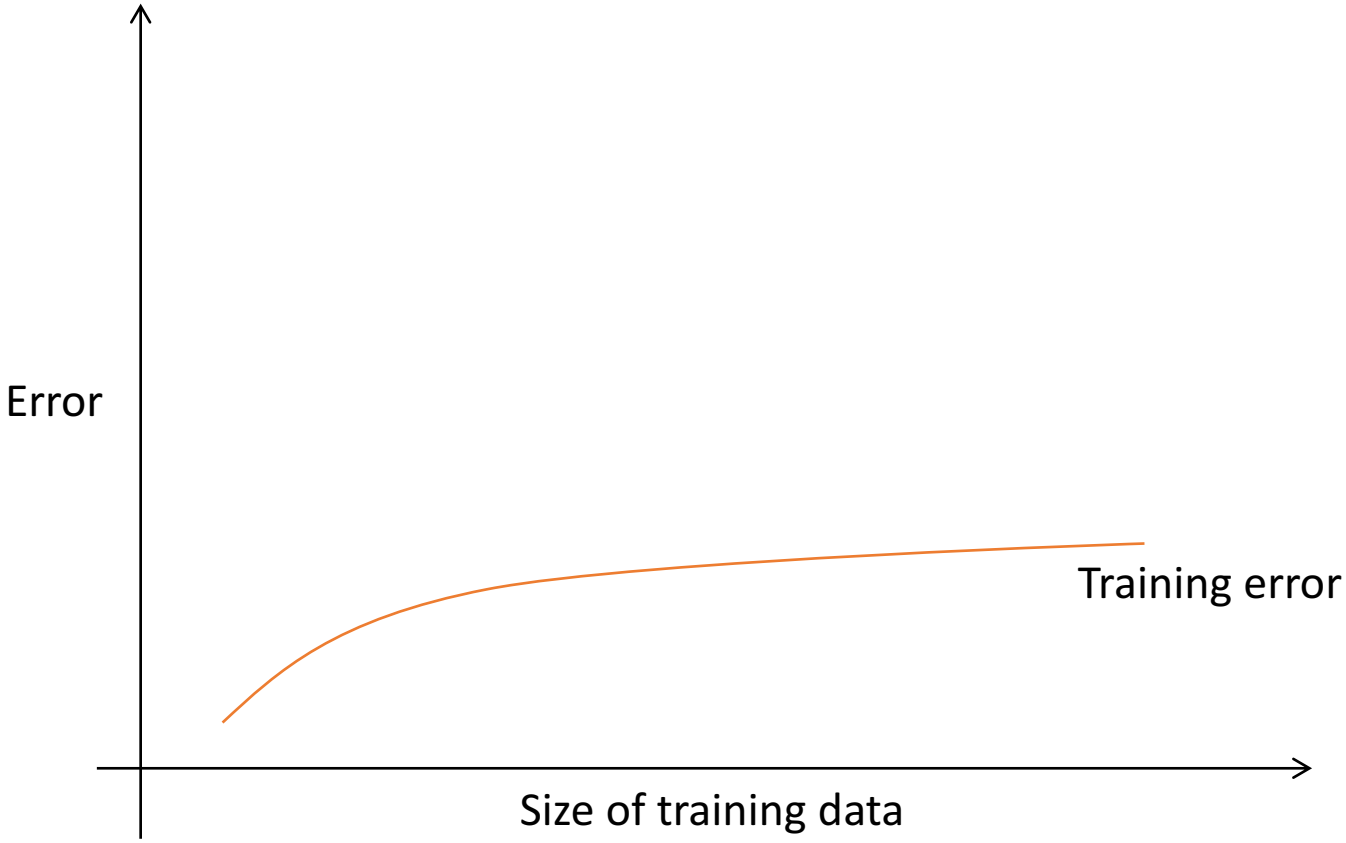
Over-fitting: The training accuracy is much higher than the test accuracy

- The model explains the training set very well, but poor generalization

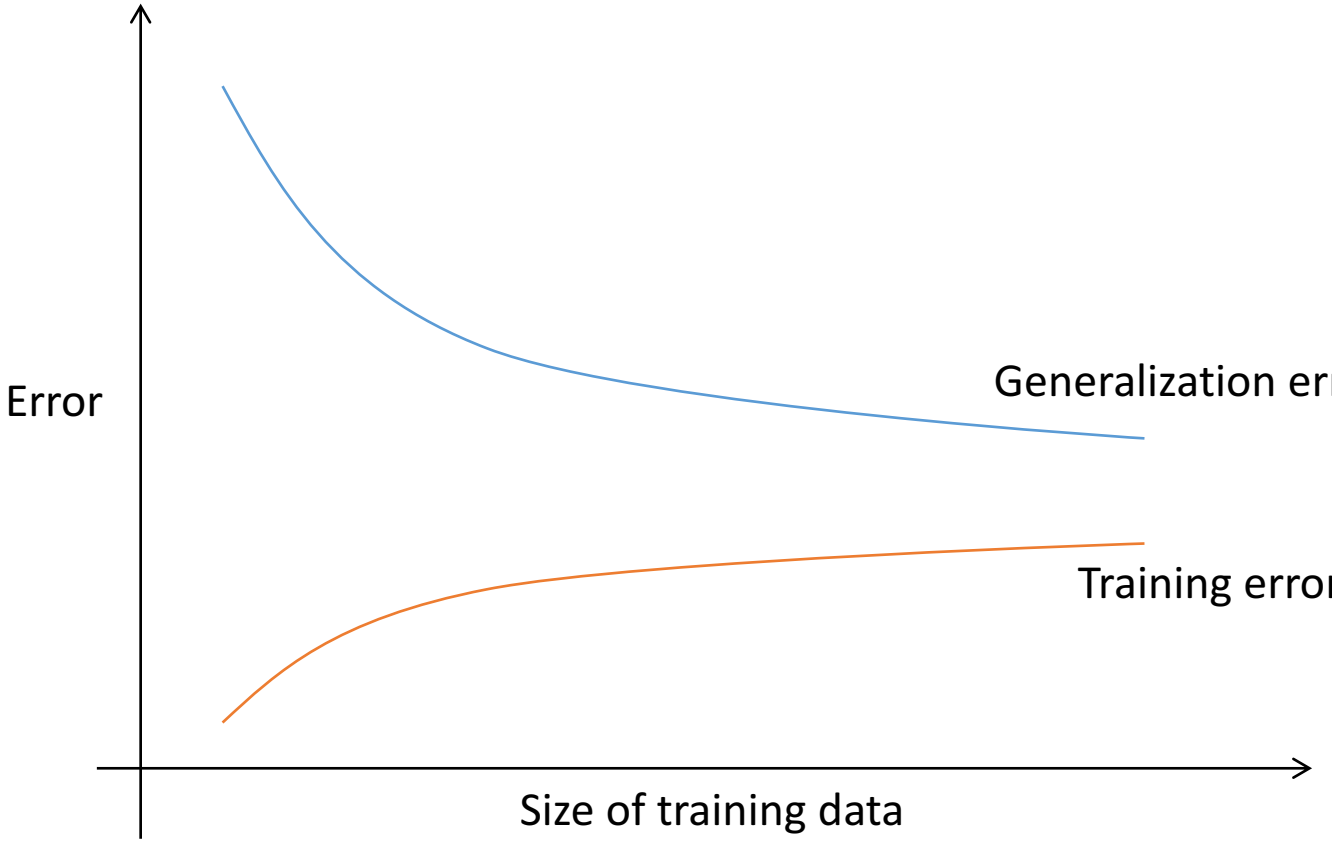
Under-fitting: Both accuracies are unacceptably low

- The model can not represent the concept well enough

Detecting high variance using learning curves

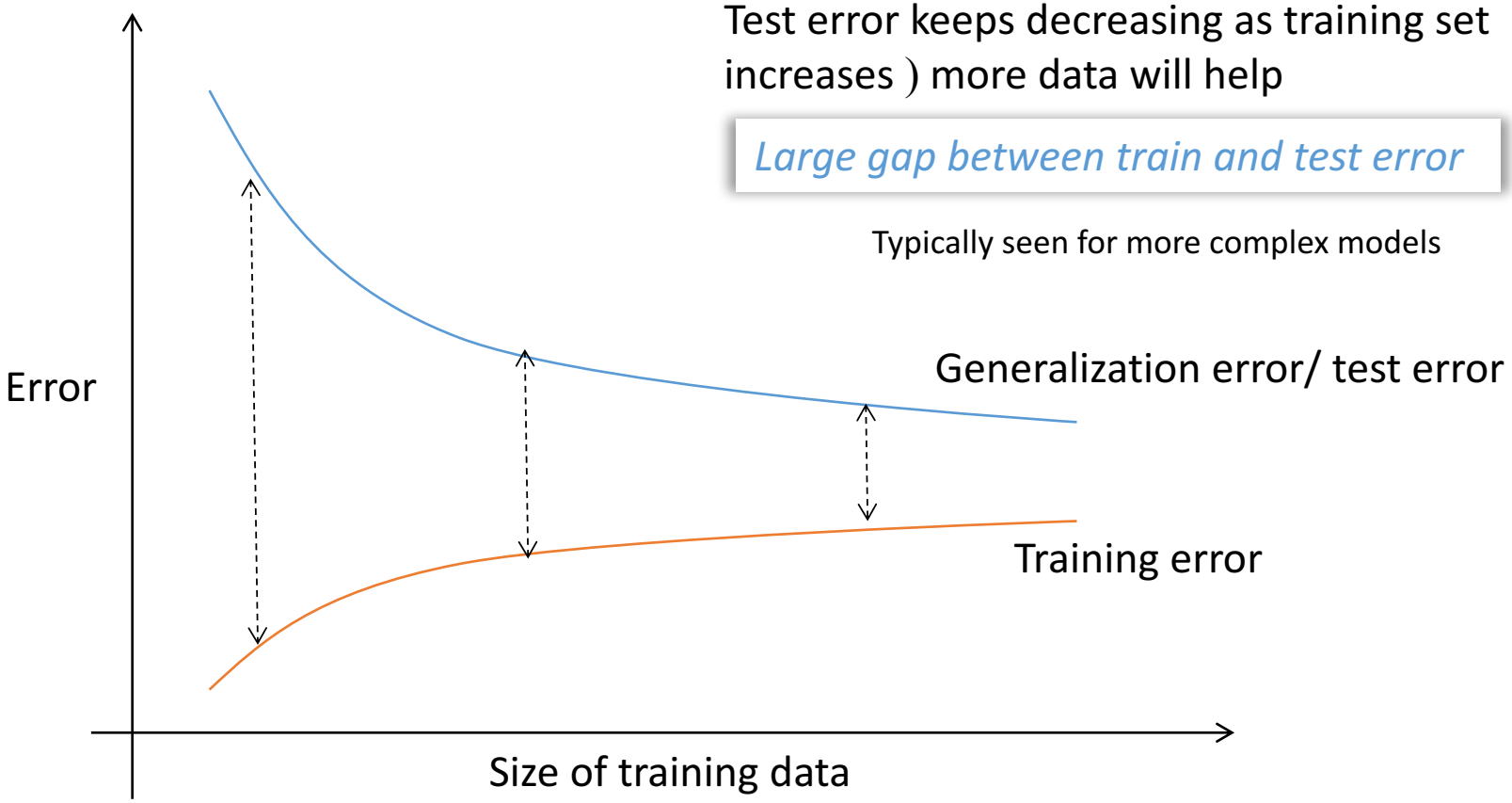


Detecting high variance using learning curves



Do we need more data?

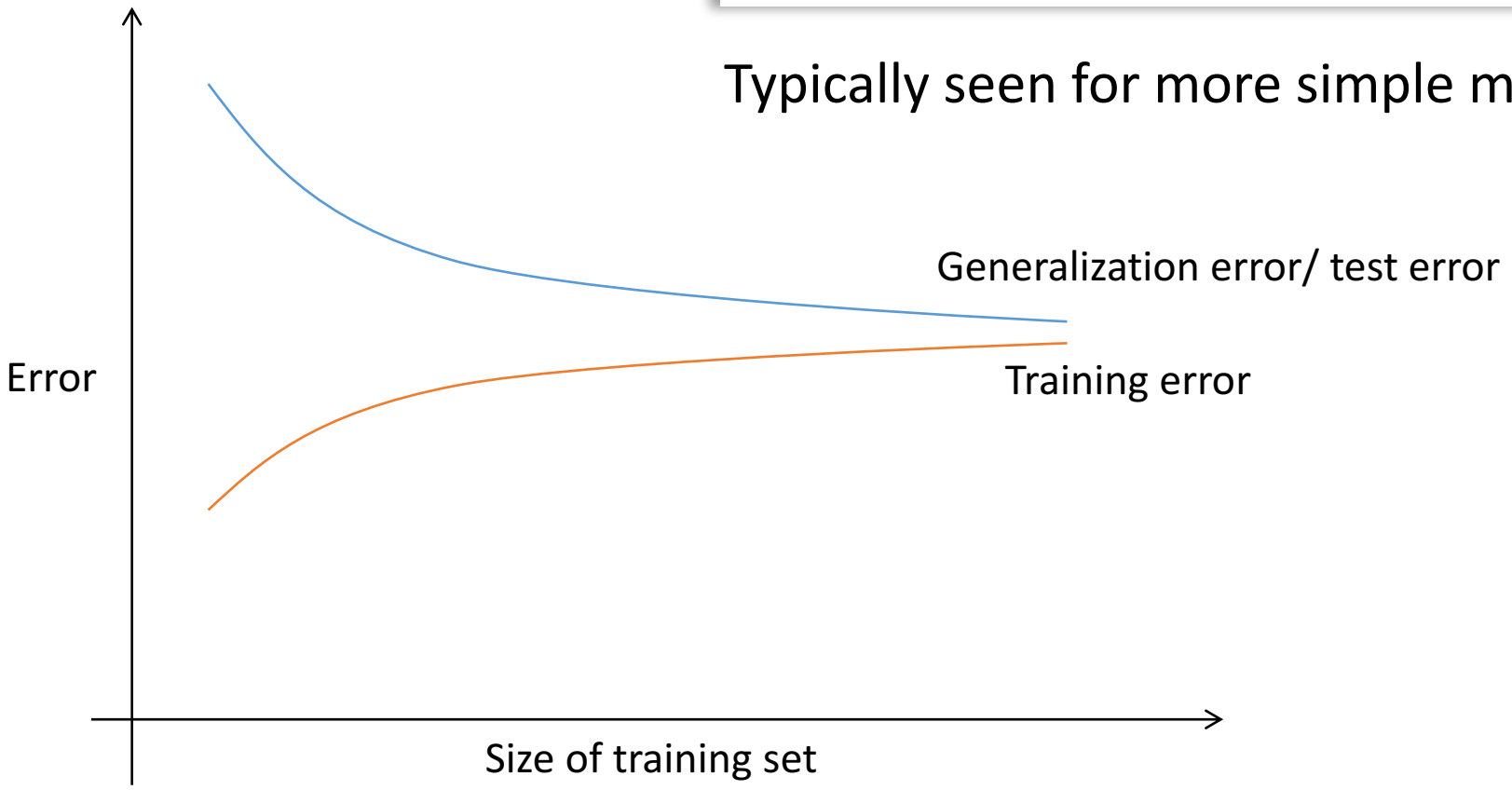
Detecting high variance using learning curves



Detecting high bias using learning curves

Both train and test error are unacceptable
(But the model seems to converge)

Typically seen for more simple models



Different ways to improve your model

More training data

Features

1. Use more features
2. Use fewer features
3. Use other features

Better training

1. Run for more iterations
2. Use a different algorithm
3. Use a different classifier
4. Play with regularization

Different ways to improve your model

More training data

Helps with over-fitting

Features

1. Use more features — Helps with under-fitting
2. Use fewer features — Helps with over-fitting
3. Use other features — Helps with over-fitting or under-fitting

Better training

1. Run for more iterations
2. Use a different algorithm — Could help with over-fitting / and under-fitting
3. Use a different classifier
4. Play with regularization

Diagnostics

Easier to fix a problem if you know where it is

Some possible problems:

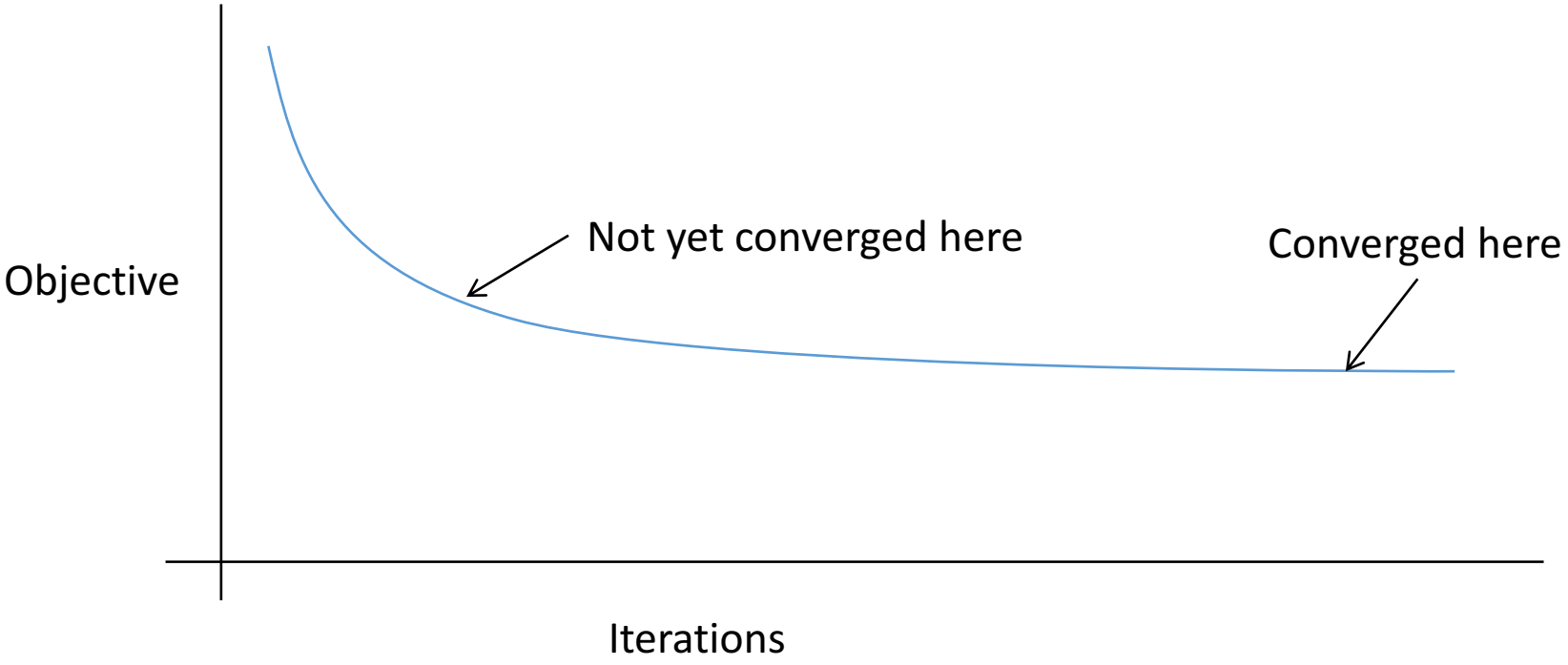
- ✓ Over-fitting (high variance)
- ✓ Under-fitting (high bias)

3. Your learning does not converge

4. Are you measuring the right thing?

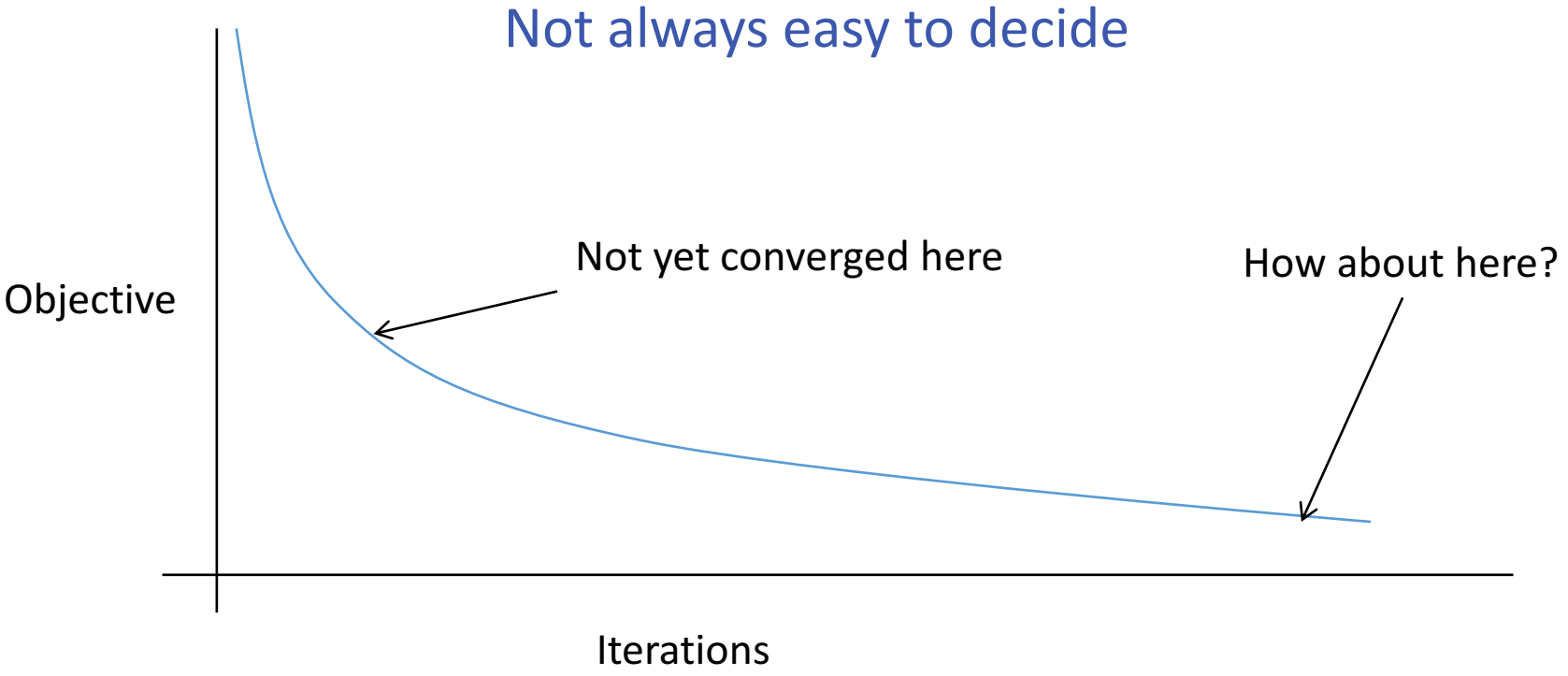
Does your learning algorithm converge?

If learning is framed as an optimization problem, track the objective



Does your learning algorithm converge?

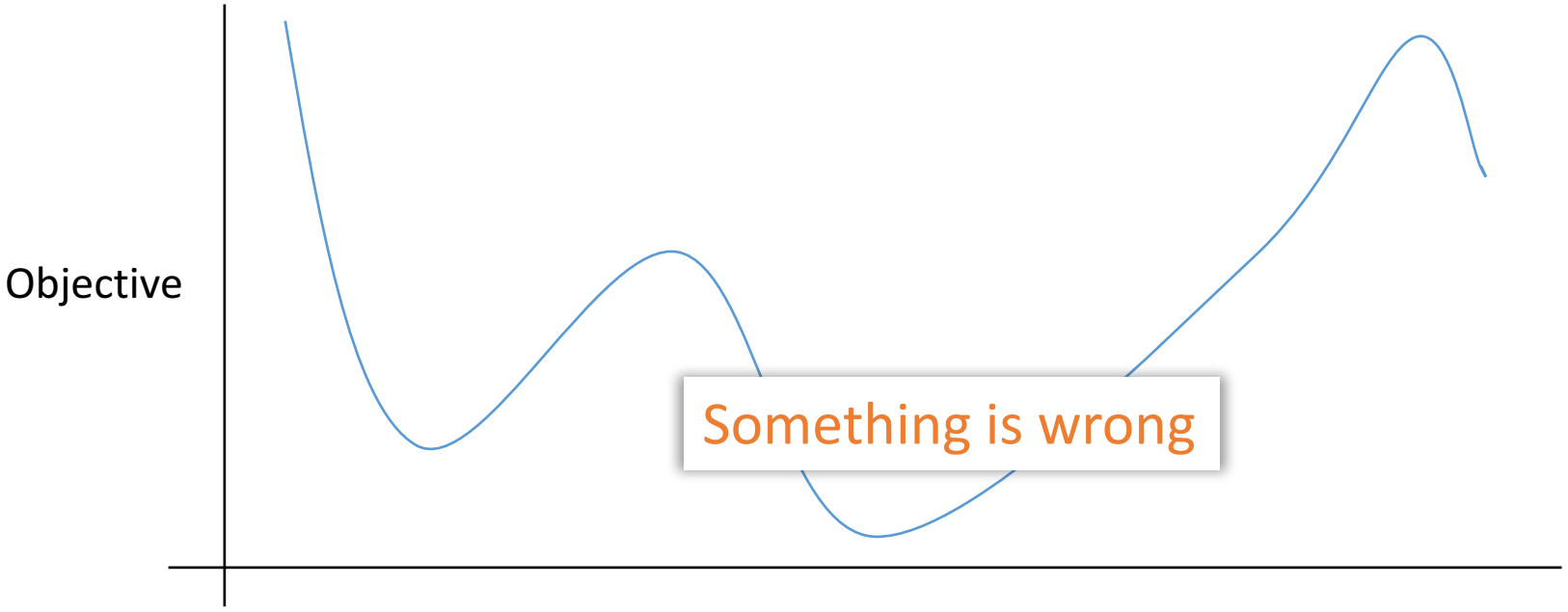
If learning is framed as an optimization problem, track the objective



Does your learning algorithm converge?

If learning is framed as an optimization problem, track the objective

Check gradient!



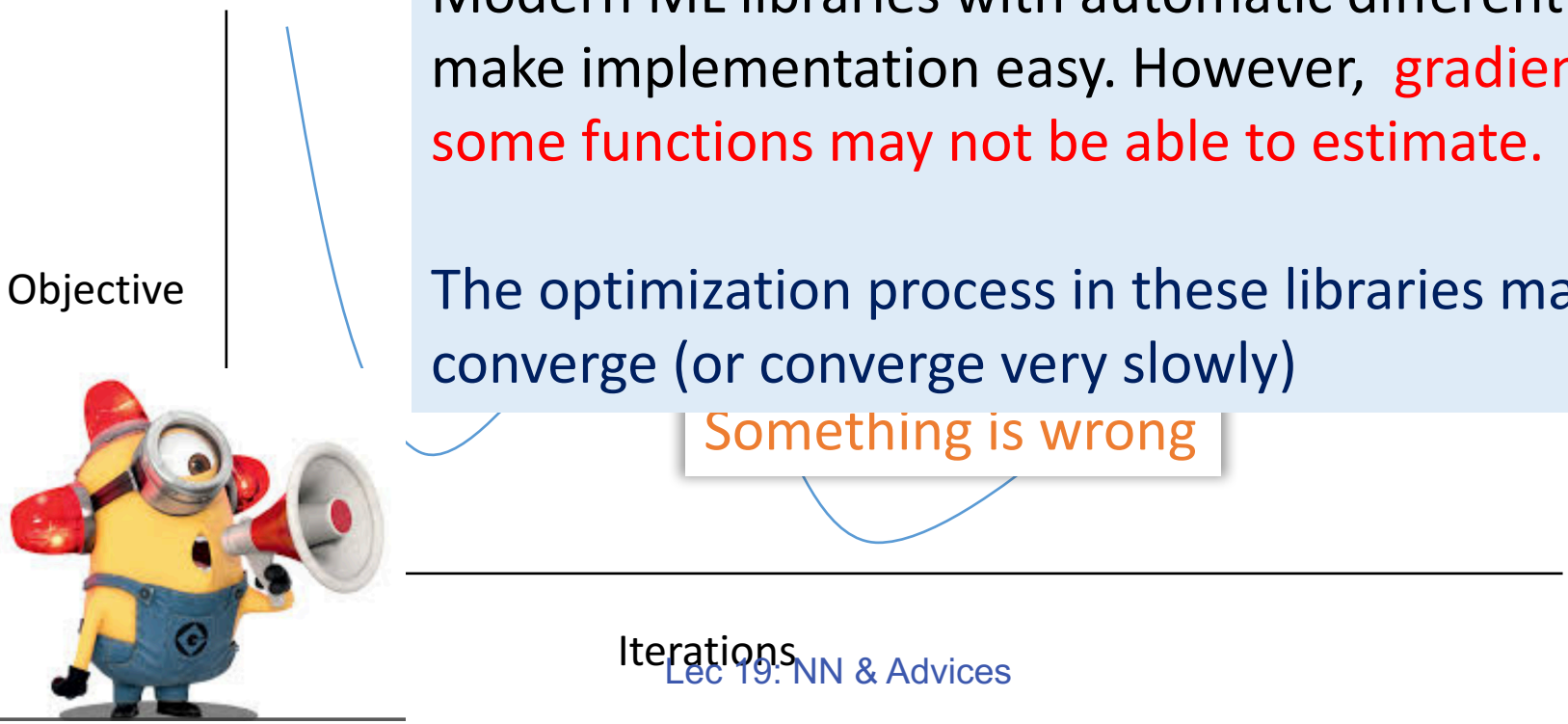
Does your learning algorithm converge?

If learning is framed as an optimization problem, track the objective

Check gradient!

Modern ML libraries with automatic differentiation make implementation easy. However, gradient of some functions may not be able to estimate.

The optimization process in these libraries may not converge (or converge very slowly)



Diagnostics

Easier to fix a problem if you know where it is

Some possible problems:

- ✓ Over-fitting (high variance)
- ✓ Under-fitting (high bias)
- ✓ Your learning does not converge
- ❖ Are you measuring the right thing?

What to measure

- Accuracy of prediction is the most common measurement
- If your data set is unbalanced, accuracy may be misleading
  - 1000 positive examples, 1 negative example
  - A classifier that always predicts positive will get 99.9% accuracy. Has it really learned anything?

- Unbalanced labels → measure label specific precision, recall and F-measure

ML and the world

- Bias vs Variance
- Diagnostics of your learning algorithm
- Error analysis
- Injecting machine learning into *Your Favorite Task*

Machine Learning in this class

Machine Learning in context

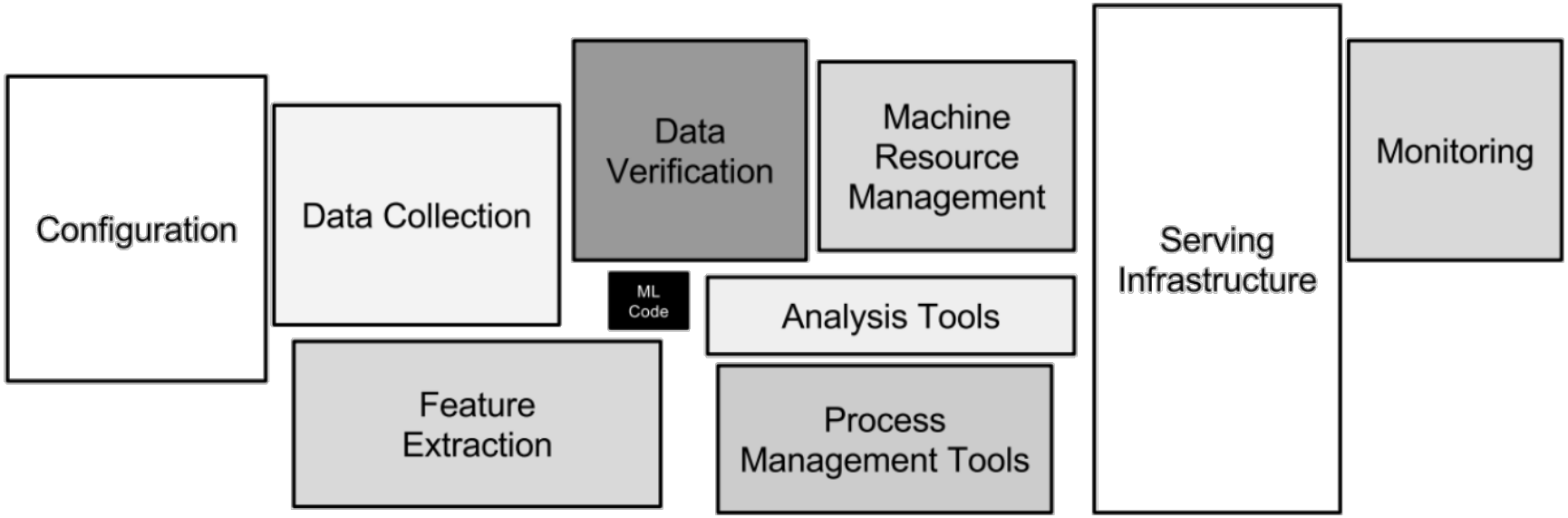


Figure from [Sculley, et al NIPS 2015]

Error Analysis

Generally machine learning plays a small role in a larger application

- Pre-processing
- Feature extraction (possibly by other ML based methods)
- Data transformations

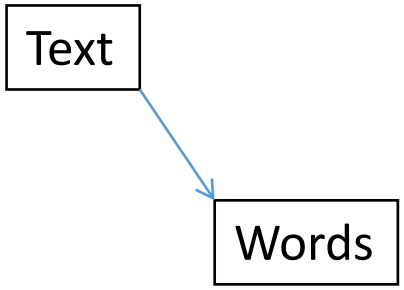
How much do each of these contribute to the error?

Error analysis tries to explain why a system is not performing perfectly

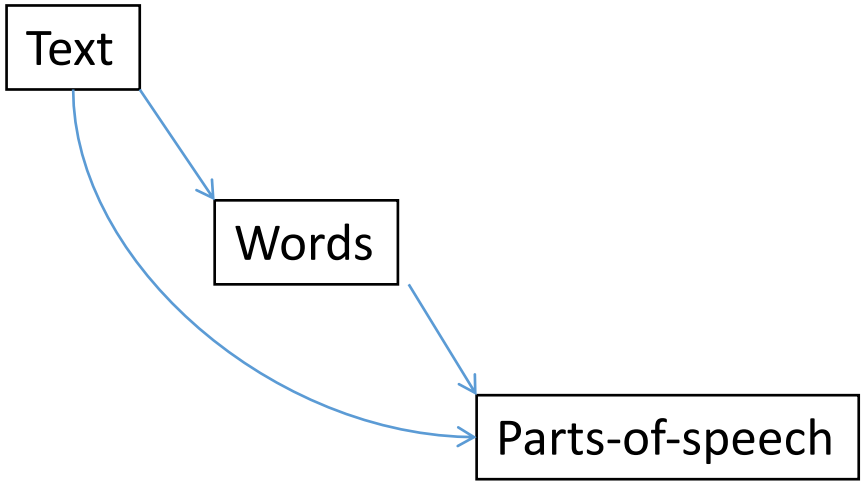
Example: A typical text processing pipeline

Example: A typical text processing pipeline

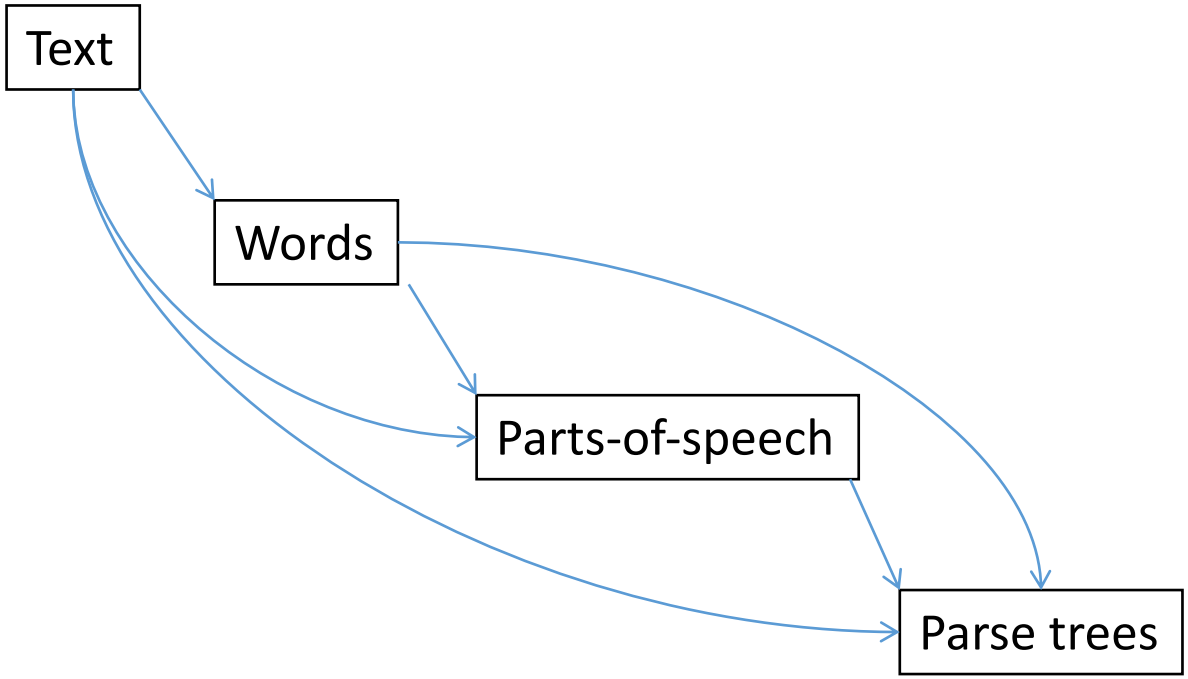
Example: A typical text processing pipeline



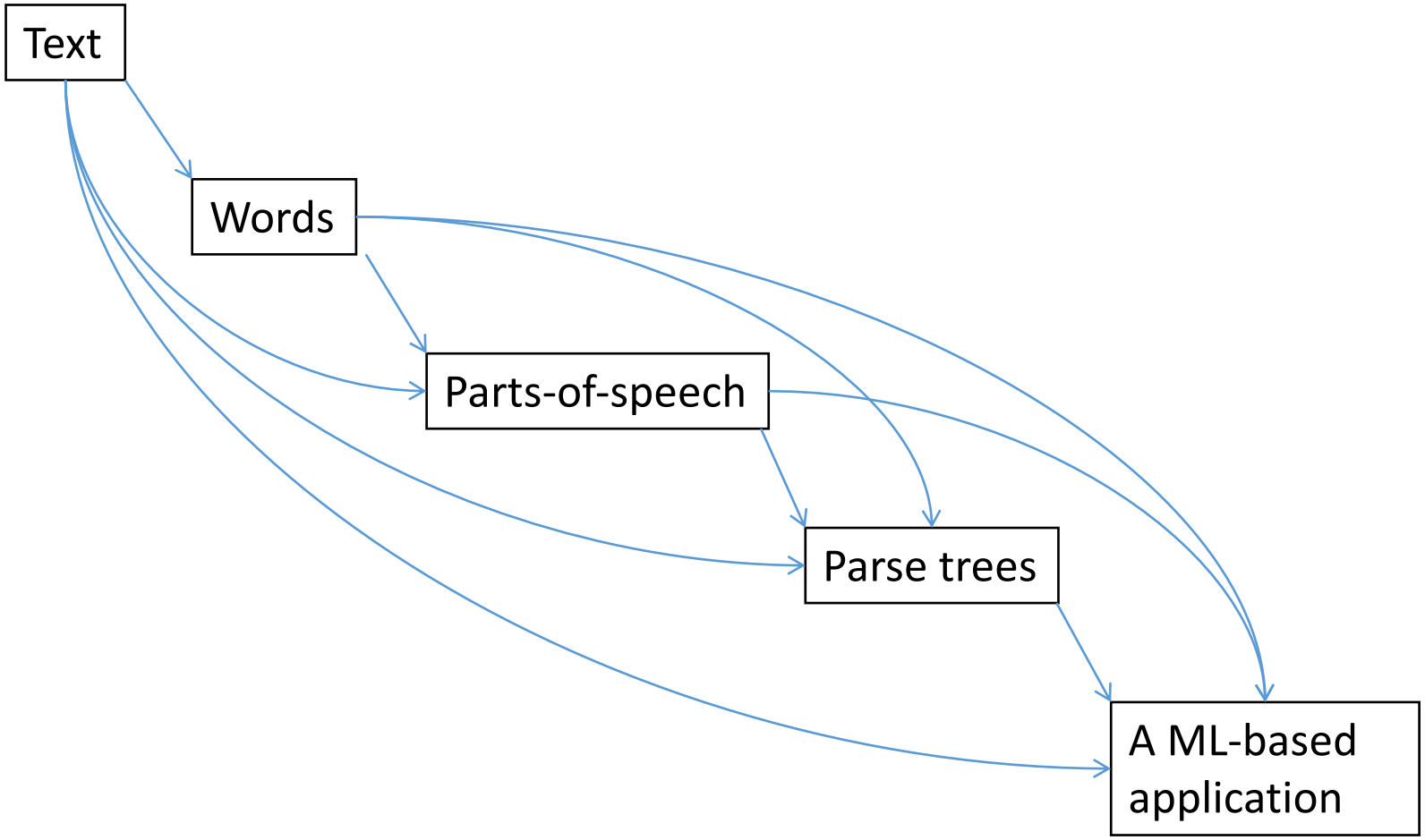
Example: A typical text processing pipeline



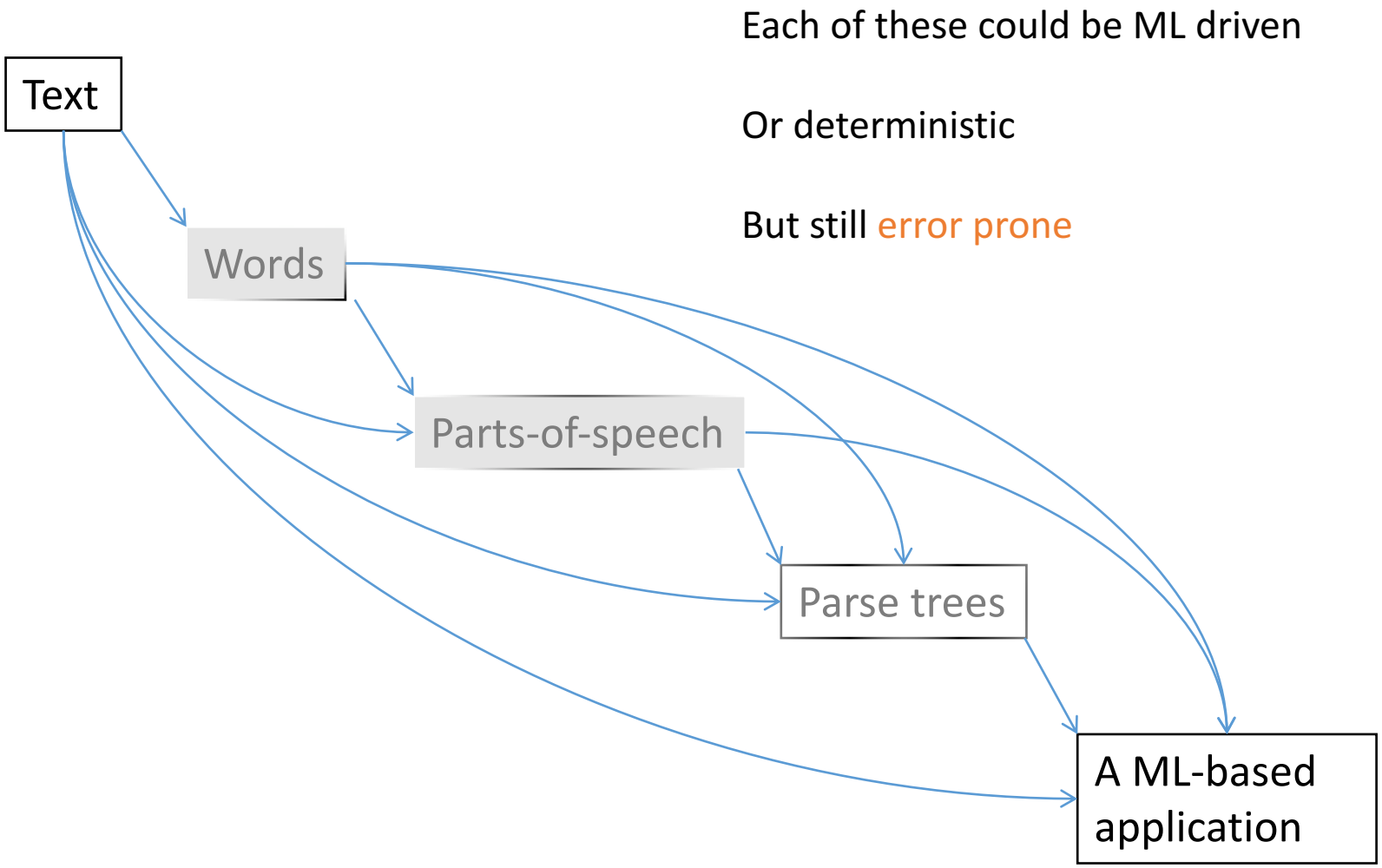
Example: A typical text processing pipeline



Example: A typical text processing pipeline



Example: A typical text processing pipeline

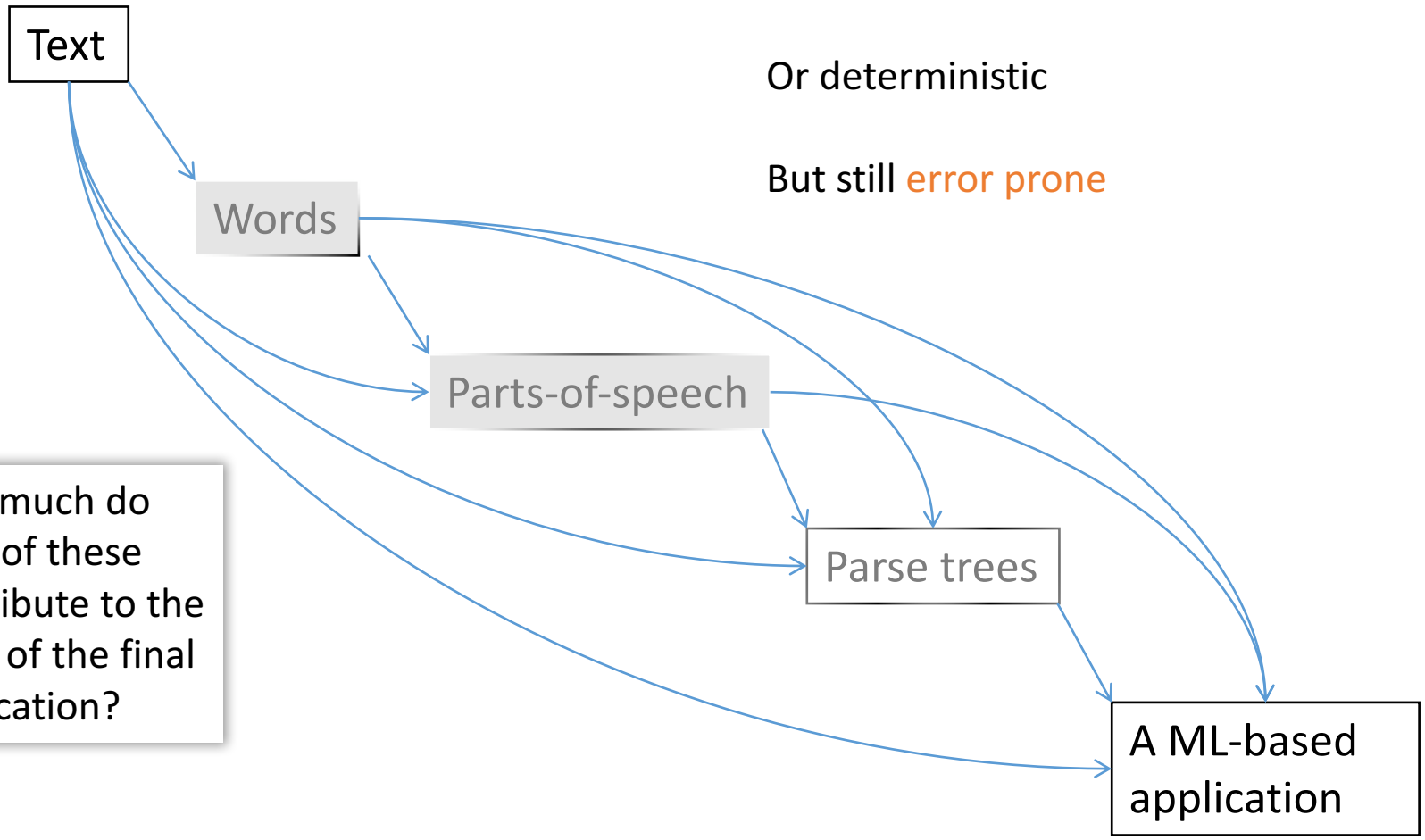


Example: A typical text processing pipeline

Each of these could be ML driven

Or deterministic

But still error prone

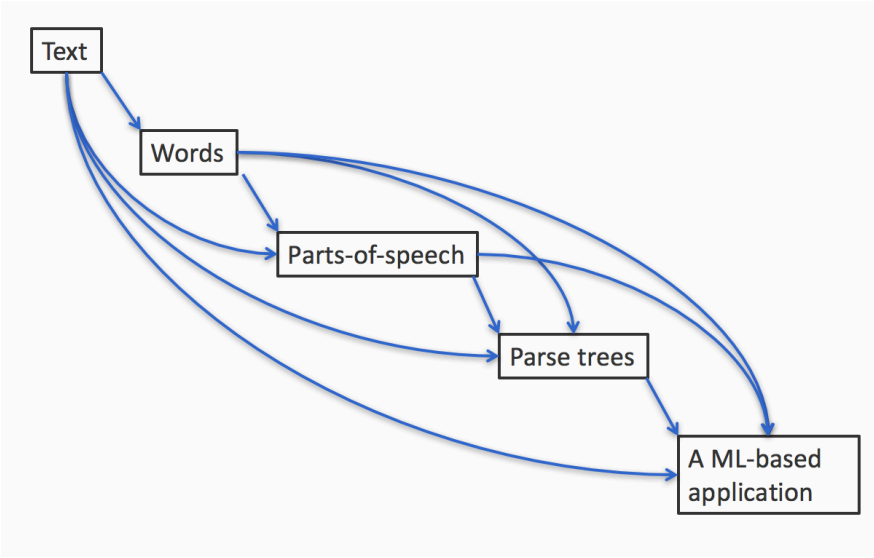


How much do each of these contribute to the error of the final application?

Tracking errors in a complex system

Plug in the ground truth for the intermediate components and see how much the accuracy of the final system changes

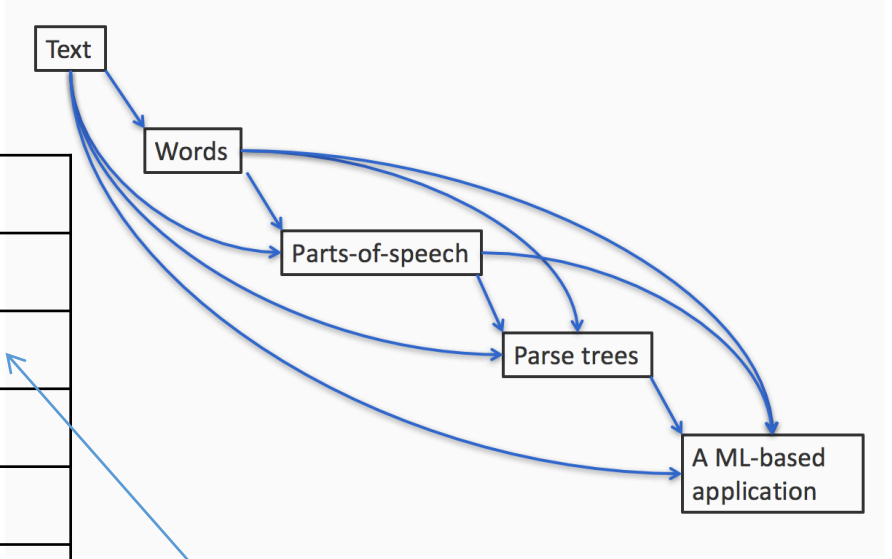
System	Accuracy
End-to-end predicted	55%
With ground truth words	60%
+ ground truth parts-of-speech	84 %
+ ground truth parse trees	89 %
+ ground truth final output	100 %



Tracking errors in a complex system

Plug in the ground truth for the intermediate components and see how much the accuracy of the final system changes

System	Accuracy
End-to-end predicted	55%
With ground truth words	60%
+ ground truth parts-of-speech	84 %
+ ground truth parse trees	89 %
+ ground truth final output	100 %



Error in the part-of-speech component hurts the most

Ablative study

Explaining difference between the performance between a strong model and a much weaker one (a baseline)

Usually seen with features

- Evaluate simpler systems that progressively use fewer and fewer features to see which features give the highest boost

It is not enough to have a classifier that works; it is useful to know why it works.

Helps interpret predictions, diagnose errors and can provide an audit trail

ML and the world

- Bias vs Variance
- Diagnostics of your learning algorithm
- Error analysis
- Injecting machine learning into *Your Favorite Task*

Classifying fish

Say you want to build a classifier that identifies whether a real physical fish is salmon or tuna

How do you go about this?

Classifying fish

Say you want to build a classifier that identifies whether a real physical fish is salmon or tuna

How do you go about this?

The slow approach

1. Carefully identify features, get the best data, the software architecture, maybe design a new learning algorithm
2. Implement it and hope it works

Advantage: Perhaps a better approach, maybe even a new learning algorithm. Research.

Classifying fish

Say you want to build a classifier that identifies whether a real physical fish is salmon or tuna

How do you go about this?

The slow approach

1. Carefully identify features, get the best data, the software architecture, maybe design a new learning algorithm
2. Implement it and hope it works

Advantage: Perhaps a better approach, maybe even a new learning algorithm. Research.

The hacker's approach

1. First implement something
2. Use diagnostics to iteratively make it better

Advantage: Faster release, will have a solution for your problem quicker

What to watch out for

- Do you have the right evaluation metric?
  - And does your loss function reflect it?

- Beware of contamination:

  Ensure that your training data is not contaminated with the test set
  - Do not see your test set either. You may inadvertently contaminate the model
  - Beware of contaminating your features with the label!
  - (Be suspicious of perfect predictors)

What to watch out for

- Be aware of bias vs. variance tradeoff (or over-fitting vs. under-fitting)
- Be aware that intuitions may not work in high dimensions
  - No proof by picture
  - Curse of dimensionality
- A theoretical guarantee may only be theoretical
  - May make invalid assumptions (eg: if the data is separable)
  - May only be legitimate with infinite data (eg: estimating probabilities)
  - Experiments on real data are equally important

Big data is not enough

But more data is always better

- Cleaner data is even better

Remember that learning is impossible without some bias that simplifies the search

- Otherwise, no generalization

Learning requires knowledge to guide the learner

- *Machine learning is not a magic wand*

Miscellaneous advice

- Learn simpler models first
  - If nothing, at least they form a baseline that you can improve upon
- Ensembles seem to work better
- Think about whether your problem is learnable at all
  - Learning = generalization

A retrospective look at the course

Learning = generalization

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Tom Mitchell (1999)

We saw different "models"

Or: what kind of a function should a learner learn

- Linear classifiers
- Decision trees
- Non-linear classifiers, feature transformations, neural networks
- Ensembles of classifiers

Different learning protocols

- Supervised learning
  - A *teacher* supplies a collection of examples with labels
  - The *learner* has to learn to label new examples using this data
- Unsupervised learning
  - No *teacher*, *learner* has only unlabeled examples
  - Data mining

- We did not see
  - Reinforcement learning
    - Learning from interaction
  - Semi-supervised learning
    - *Learner* has access to both labeled and unlabeled examples

The theory of machine learning

Mathematically defining learning

- Online learning
- Probably Approximately Correct (PAC) Learning
- Bayesian learning

What we saw

1. A broad theoretical and practical understanding of machine learning paradigms and algorithms

2. Ability to implement learning algorithms

3. Identify where machine learning can be applied and make the most appropriate decisions (about algorithms, models, supervision, etc)

Thank you

- Please log in MyUCLA for course survey