# CS 190I
# Deep Learning
# Model Evaluation

Lei Li (leili@cs)
UCSB

Acknowledgement: Slides borrowed from Bhiksha Raj's 11485 and Mu Li & Alex Smola's 157 courses on Deep Learning, with modification

# Recap

- Compute the gradient through Back-propagation algorithm
  - with forward pass and backward pass
  - backward pass is application of chain rule

# Forward "Pass"

- Input: $D$ dimensional vector $\mathbf{x} = [x_j, \;\; j = 1 \ldots D]$
- Set:
  - $D_0 = D,$ is the width of the 0th (input) layer
  - $y_j^{(0)} = x_j, \;\; j = 1 \ldots D;$ $\quad y_0^{(k=1 \ldots N)} = x_0 = 1$
- For layer $k = 1 \ldots N$
  - For $j = 1 \ldots D_k$
    - $z_j^{(k)} = \sum_{i=0}^{D_{k-1}} w_{i,j}^{(k)} y_i^{(k-1)}$
    - $y_j^{(k)} = f_k\left(z_j^{(k)}\right)$

$D_k$ is the size of the kth layer

- Output:
  - $Y = y_j^{(N)}, \;\; j = 1..D_N$

# Backward Pass

- Output layer $(N)$ :
  - For $i = 1 \ldots D_N$
    - $$\frac{\partial \ell}{\partial z_i^{(N)}} = f_N'(z_i^{(N)}) \frac{\partial \ell}{\partial \hat{y}_i^{(N)}}$$
    - $$\frac{\partial \ell}{\partial w_{ij}^{(N)}} = y_i^{(N-1)} \frac{\partial \ell}{\partial z_j^{(N)}} \text{ for each j}$$

Called "Backpropagation" because the derivative of the loss is propagated "backwards" through the network

- For layer $k = N - 1 \; downto$

Very analogous to the forward pass:

  - For $i = 1 \ldots D_k$
    - $$\frac{\partial \ell}{\partial y_i^{(k-1)}} = \sum_j w_{ij}^{(k)} \frac{\partial \ell}{\partial z_j^{(k)}}$$ ← Backward weighted combination of next layer
    - $$\frac{\partial \ell}{\partial z_i^{(k)}} = f_k'(z_i^{(k)}) \frac{\partial \ell}{\partial y_i^{(k)}}$$ ← Backward equivalent of activation
    - $$\frac{\partial \ell}{\partial w_{ij}^{(k)}} = y_i^{(k-1)} \frac{\partial \ell}{\partial z_j^{(k)}} \text{ for each j}$$

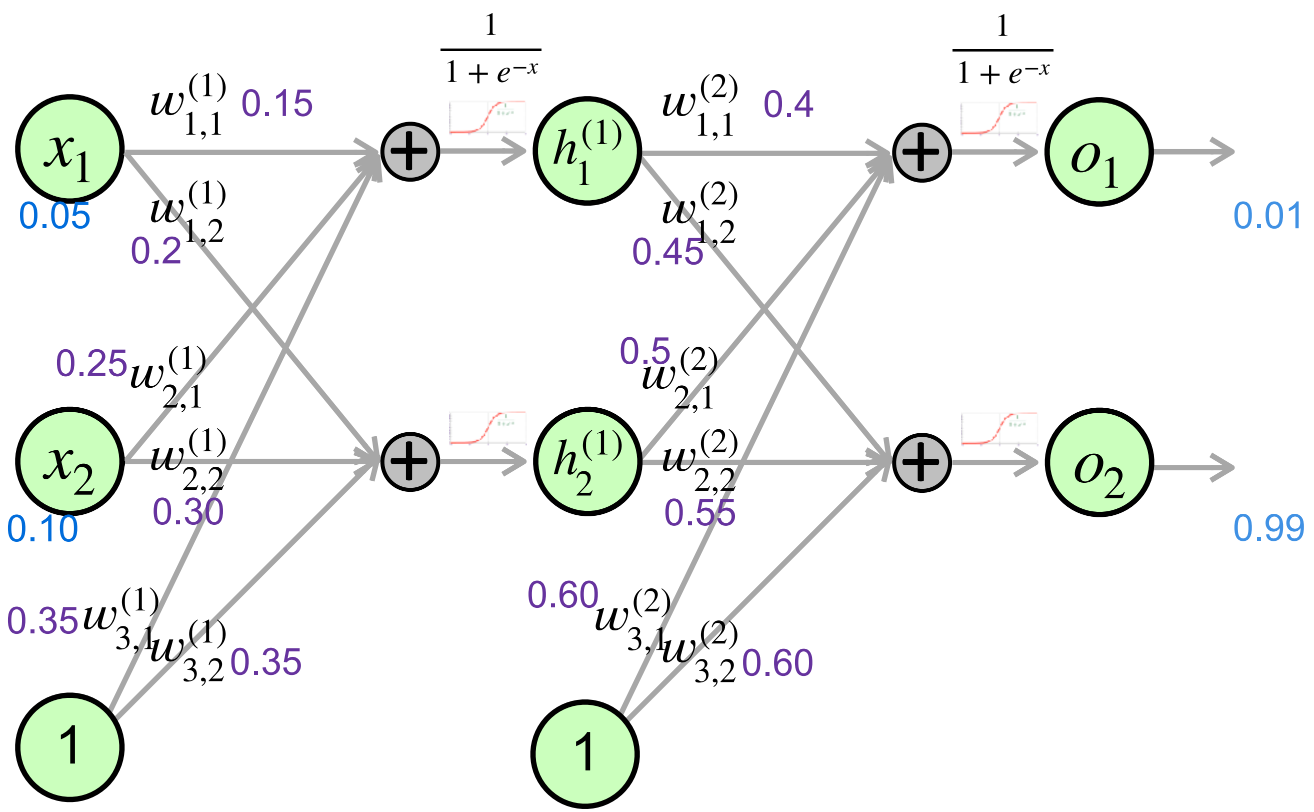
# Gradient Descent for FFN

learning rate eta.

1. set initial parameter $\theta \leftarrow \theta_0$
2. for epoch = 1 to maxEpoch or until converge:
3.   for each data (x, y) in D:
4.     compute forward y_hat = f(x; $\theta$)
5.     compute gradient $g = \frac{\partial \text{err}(y_{hat}, y)}{\partial \theta}$ using backpropagation
6.     total_g += g
7.   update $\theta$ = $\theta$ − eta * total_g / num_sample

# Quiz (on Edstem)

Calculate all gradients, using MSE $\frac{1}{2}|y - o|_2^2$



# Model Evaluation

# Risk

- Generalization error: The expected risk is the average risk (loss) over the entire (x, y) data space

$$R(\theta) = E_{\langle x,y\rangle \in P}\left[\ell(y, f(x;\theta))\right] = \int \ell(y, f(x;\theta))dP(x,y)$$

# The general learning framework: Empirical Risk Minimization (ERM)

- Ideally, we want to minimize the expected risk
  - but, unknown data distribution …
- Instead, given a training set of empirical data $D = \{(x_n, y_n)\}_{n=1}^{N}$
- Minimize the empirical risk over training data

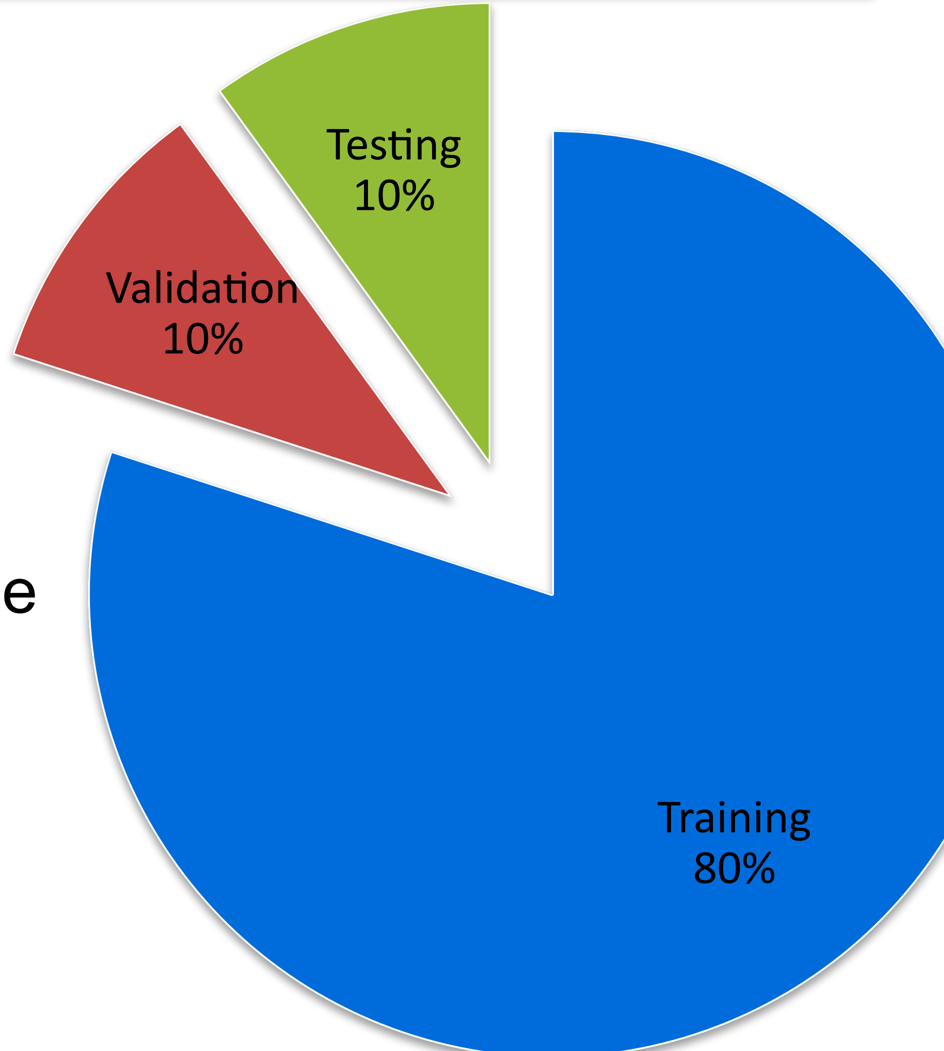
$$\hat{\theta} \leftarrow \arg\min_{\theta} L(\theta) = \frac{1}{N} \sum_{n} \ell(y_n, f(x_n; \theta))$$

# Training and Generalization

- Training error (=empirical risk): model prediction error on the training data
- Generalization error (= expected risk): model error on new unseen data over full population
- Example: practice a GRE exam with past exams
  - Doing well on past exams (training error) doesn't guarantee a good score on the future exam (generalization error)
  - Student A gets 0 error on past exams by rote learning
  - Student B understands the reasons for given answers

# Validation Dataset and Test Dataset

- Validation dataset: a dataset used to evaluate the model performance
  - E.g. Take out 50% of the training data
  - Should not be mixed with the training data (#1 mistake)
- Test dataset: a dataset can be used once, e.g.
  - A future exam
  - The house sale price I bided
  - Dataset used in private leaderboard in Kaggle

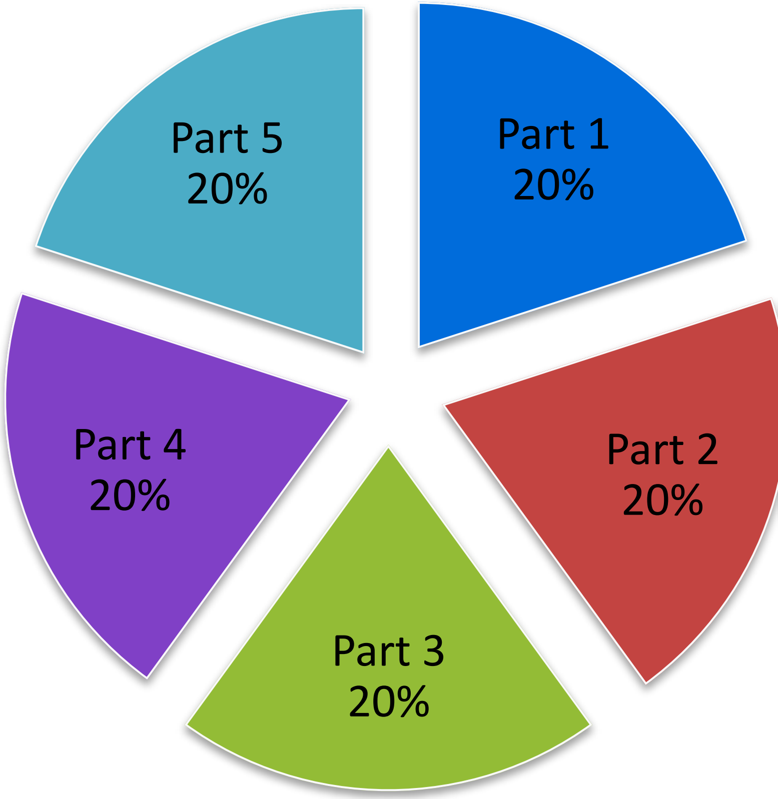


# Model Inference

- After train a model
- Given an input data x
- to compute the prediction for output y
- For regression:
  - just model output
- For classification:
  - $\hat{y} = \arg\max_i f(x)_i$
- Need to do inference for validation and testing

# K-fold Cross-Validation

- Useful when insufficient data
- Algorithm:
  - Partition the training data into K parts
  - For i = 1, …, K
    - Use the i-th part as the validation set, the rest for training
    - Train the model using training set, and evaluate the performance on validation set.
  - Report the averaged the K validation errors
- Popular choices: K = 5 or 10



# Underfitting
# Overfitting


Image credit: hackernoon.com

# Underfitting and Overfitting

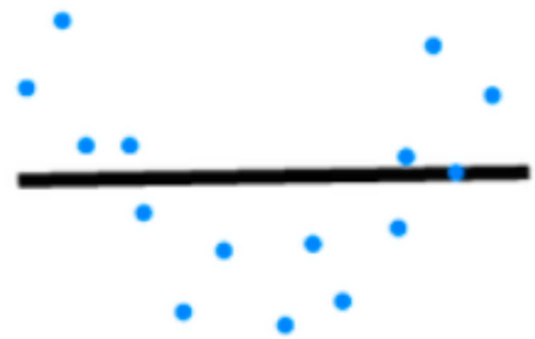
**Data complexity**

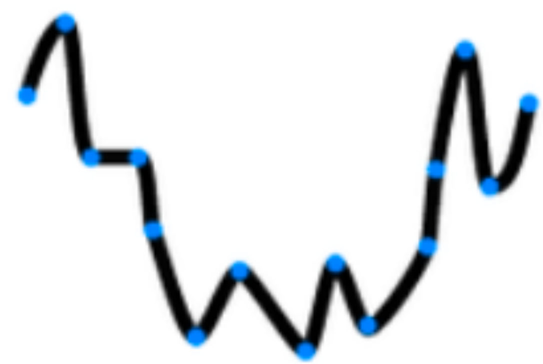
**Model capacity**

| | Simple | Complex |
|---|---|---|
| Low | ok | Underfitting |
| High | Overfitting | ok |

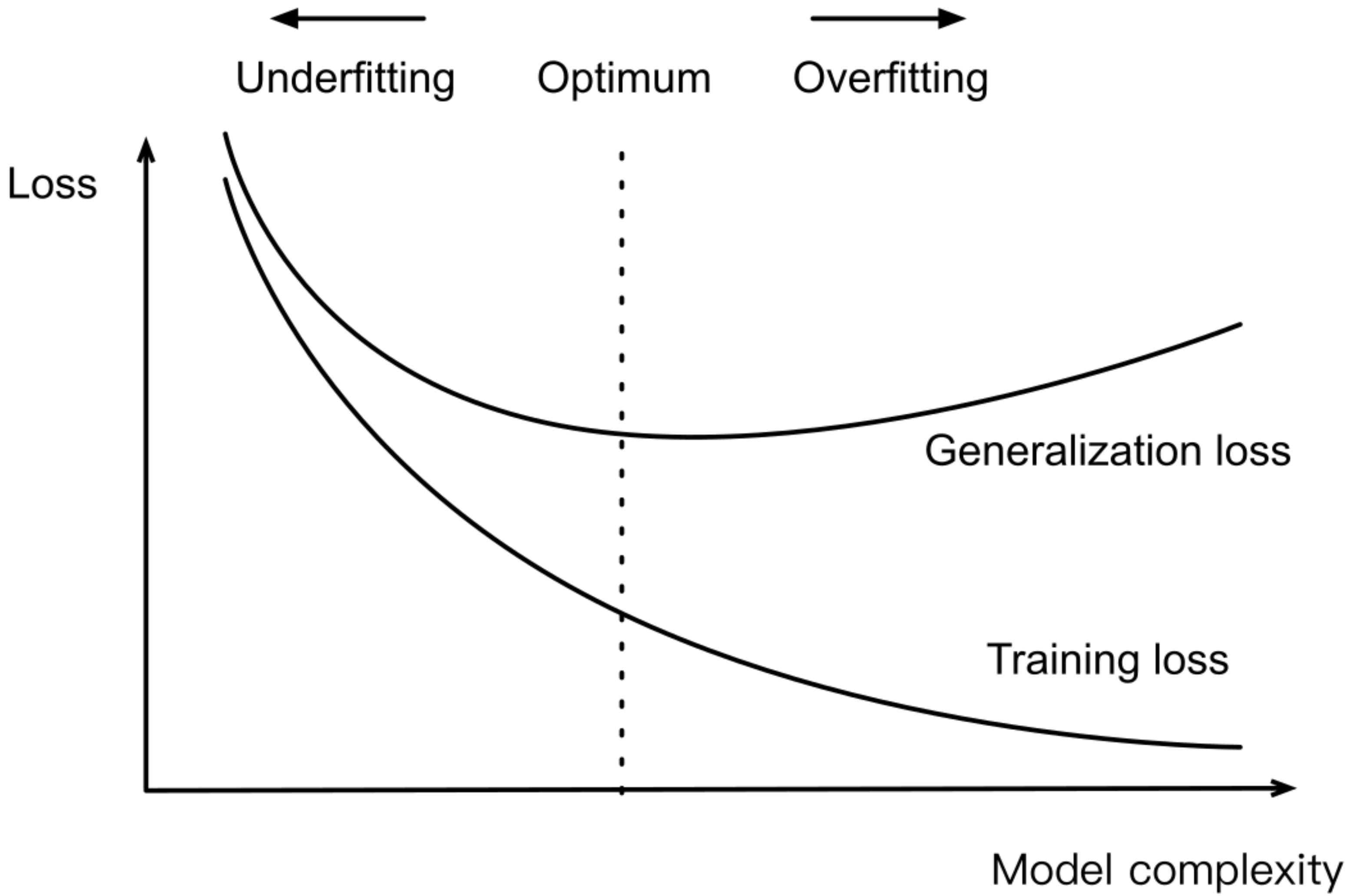
# Model Capacity

- The ability to fit variety of functions
- Low capacity models struggles to fit training set
  - Underfitting
- High capacity models can memorize the training set
  - Overfitting





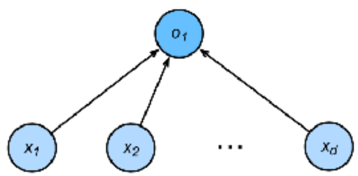
# Influence of Model Complexity



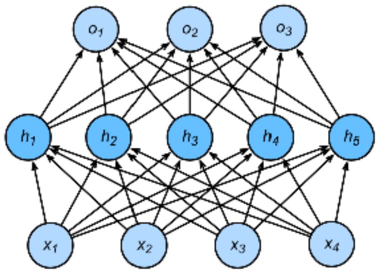
# Estimate Model Capacity

- It's hard to compare complexity between different algorithms
  - e.g. tree vs neural network
- Given an algorithm family, two main factors matter:
  - The number of parameters
  - The values taken by each parameter

$d + 1$




$(d + 1)m + (m + 1)k$



# VC Dimension

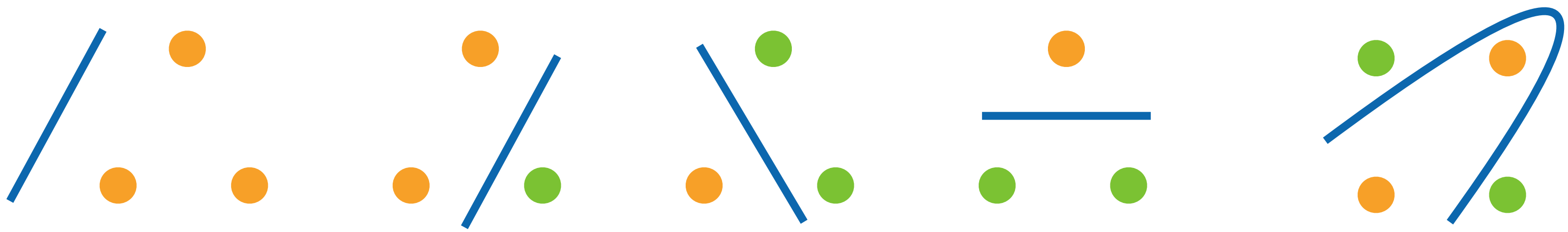
- A center topic in Statistic Learning Theory
- For a classification model, it's the size of the largest dataset, no matter how we assign labels, there exist a model to classify them perfectly

Vladimir **V**apnik

Alexey **C**hervonenkis

# VC-Dimension for Classifiers

- 2-D logistic regression: VCdim = 3
  - Can classify any 3 points, but not 4 points (xor)



- Logistic Regression with *N* parameters: VCdim = $N$
- Some Multilayer Perceptrons: VCdim = $O(N \log_2(N))$

# Usefulness of VC-Dimension

- Provides theoretical insights why a model works
  - Bound the gap between training error and generalization error
- Rarely used in practice with deep learning
  - The bounds are too loose
  - Difficulty to compute VC-dimension for deep neural networks
- Same for other statistic learning theory tools

# Data Complexity

- Multiple factors matters
  - # of examples
  - # of features in each example
  - temporal/spacial structure
  - diversity/coverage

# Recap

- Model evaluation
  - Empirical risk minimization
  - training, validation, testing
  - Cross validation
- Under-fitting
  - model cannot fit the data well
  - => increase model complexity
- Overfitting
  - Model fits well on training data, but does not perform well on testing data
  - => regularization (next lecture)

# Next Up

- Regularization
- Convolutional Neural Networks
- Visual perception:
  - Image classification
  - Object recognition
  - Face detection