

CS273B: Deep learning for Genomics and Biomedicine

Lecture 1: Machine learning 101

09/25/2017

Anshul Kundaje, James Zou

Course Outline

Course overview

- Deep learning + computational biology = exciting!
- This class is **projects focused** research class:
best way to learn and state-of-the-art very limited
- **Prerequisites**: first course in machine learning and statistics. We will cover the relevant biology and neural networks.

Course overview

Deep Learning

Supervised learning: feedforward, convnets and RNN/LSTM.

Unsupervised/representation learning: autoencoders, VAE, GANs.

General techniques: optimization, regularization, TensorFlow.

Course overview

Deep Learning

Supervised learning: feedforward, convnets and RNN/LSTM.

Unsupervised/representation learning: autoencoders, VAE, GANs.

General techniques: optimization, regularization, TensorFlow.

Biomedicine

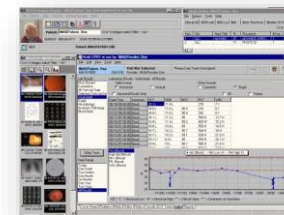
Genomics: regulatory genomics, population genetics.



Protein structure/drug discovery



Medical records and imaging



Course schedule

Check Canvas for latest and resources.

Date	Topic	Papers	Recitation topic	Assignment
9/25	Overview, Intro to neural networks and genomics.	1. Deep Learning http://www.nature.com/nature/journal/v521/n7553/full/nature14539.html 2. Neural Nets and Deep learning primer http://neuralnetworksanddeeplearning.com/		
9/27	Genomics + ConvNets	1. Deep learning for computational biology http://msb.embopress.org/content/12/7/878 2. DeepSEA: Predicting effects of noncoding variants with deep learning-based sequence model http://www.nature.com/nbt/journal/v33/n8/full/nbt.3300.html 3. Denoising genome-wide histone ChIP-seq with convolutional neural networks https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btx243	Deep learning primer	
10/2	Recurrent NN			
10/4	TensorFlow + Azure tutorial		Genomics primer	Projects released on 10/6.
10/9	Optimization and regularization			
10/11	Autoencoders + representation learning		DragonNN primer	Select projects.
10/16	Generative models	GANs for Biological Image Synthesis https://arxiv.org/abs/1708.04692		
10/18	Paper presentations	1. Basenji: Sequential regulatory activity prediction across chromosomes with convolutional neural networks https://www.biorxiv.org/content/early/2017/07/10/161851 2. FIDDLE: An integrative deep learning framework for functional genomic data inference https://www.biorxiv.org/content/early/2016/10/17/081380 3. DeepCpG: accurate prediction of single-cell DNA methylation states using deep learning https://genomebiology.biomedcentral.com/articles/10.1186/s13059-017-1189-z	Proteins primer	
10/23	Interpretation of black-box models	1. The Mythos of Model Interpretability https://arxiv.org/abs/1606.03490 2. DeepLIFT: Learning Important Features Through Propagating Activation Differences https://arxiv.org/abs/1704.02685v1 3. Deep Motif Dashboard: Visualizing and Understanding Genomic Sequences Using Deep Neural Networks https://arxiv.org/abs/1608.03644		

10/25	Paper presentations	1. Attend and Predict: Understanding Gene Regulation by Selective Attention on Chromatin https://arxiv.org/abs/1708.00339 2. Deep learning for population genetics http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004845 3. Integrative deep models for alternative splicing https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btx268	Deep learning review	
10/30	Proposal presentations			3 minute proposal presentations
11/1	Guest lecture: Google Brain team	1. DeepVariant: Creating a universal SNP and small indel variant caller with deep neural networks https://www.biorxiv.org/content/early/2016/12/21/092890 2. Chiron: Translating nanopore raw signal directly into nucleotide sequence using deep learning https://www.biorxiv.org/content/early/2017/09/12/179531 3. Adaptive Somatic Mutations Calls with Deep Learning and Semi-Simulated Data https://www.biorxiv.org/content/early/2016/10/04/079087		
11/6	Drug discovery + protein structure	1. Deep learning for computational chemistry http://onlinelibrary.wiley.com/doi/10.1002/jcc.24764/abstract 2. MoleculeNet 3. One shot learning drug discovery		
11/8	Paper presentations	1. Accurate De Novo Prediction of Protein Contact Map by Ultra-Deep Learning Model http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1005324 2. De novo peptide sequencing by deep learning http://www.pnas.org/content/114/31/8247 3. Molecular De Novo Design through Deep Reinforcement Learning https://arxiv.org/abs/1704.07555		
11/13	Paper presentations	1. druGAN: An Advanced Generative Adversarial Autoencoder Model for de Novo Generation of New Molecules with Desired Molecular Properties in Silico http://pubs.acs.org/doi/abs/10.1021/acs.molpharmaceut.7b00346 2. Molecular Graph Convolutions: Moving Beyond Fingerprints https://arxiv.org/abs/1603.00856 , Convolutional Networks on Graphs for Learning Molecular Fingerprints http://papers.nips.cc/paper/5954-convolutional-networks-on-graphs-for-learning-molecular-fingerprints 3. Automatic chemical design using a data-driven continuous representation of molecules https://arxiv.org/abs/1610.02415v2		

Research project

Teams of 3-5 students. Each team will be given Azure GPU credits. We'll have suggested projects, datasets and mentors.


Milestones:

- Select project and **form teams** 10/11.
- **Proposal** presentation in class 10/30.
- Initial paper for **peer review** 11/29.
- **Poster** presentation 12/11
- Final **paper** due 12/15.

Paper presentations + reviews

Each team will also present one recent research paper.

- 20 mins + 5 mins Q&A in class.
- Publish review on bioRxiv.



6	Drug discovery + protein structure	1. Deep learning for computational chemistry http://onlinelibrary.wiley.com/doi/10.1002/jcc.24764/abstract 2. MoleculeNet 3. One shot learning drug discovery		
8	Paper presentations	1. Accurate De Novo Prediction of Protein Contact Map by Ultra-Deep Learning Model http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1005324 2. De novo peptide sequencing by deep learning http://www.pnas.org/content/114/31/8247 3. Molecular De Novo Design through Deep Reinforcement Learning https://arxiv.org/abs/1704.07555		
13	Paper presentations	1. druGAN: An Advanced Generative Adversarial Autoencoder Model for de Novo Generation of New Molecules with Desired Molecular Properties in Silico http://pubs.acs.org/doi/abs/10.1021/acs.molpharmaceut.7b00346 2. Molecular Graph Convolutions: Moving Beyond Fingerprints https://arxiv.org/abs/1603.00856 , Convolutional Networks on Graphs for Learning Molecular Fingerprints http://papers.nips.cc/paper/5954-convolutional-networks-on-graphs-for-learning-molecular-fingerprints 3. Automatic chemical design using a data-driven continuous representation of molecules https://arxiv.org/abs/1610.02415v2		

Each team will also review two peer papers.

Logistics

Section: Fridays 10:30 – 11:20am (380-380D).

Office hours:

James: Mondays 4:30 – 6pm (Packard 253)

TA: Thursdays 5:30 – 6:30pm (Gates B30)

Start thinking about **projects/teams** in the next two weeks.

Ask questions on **Piazza!**

Outline

1

Supervised Machine Learning

- Loss Functions

2

Artificial neuron

- Linear regression
- Logistic regression

3

Training models

- Gradient descent

4

Performance evaluation

5

Overfitting, generalization, CV

Types of machine Learning

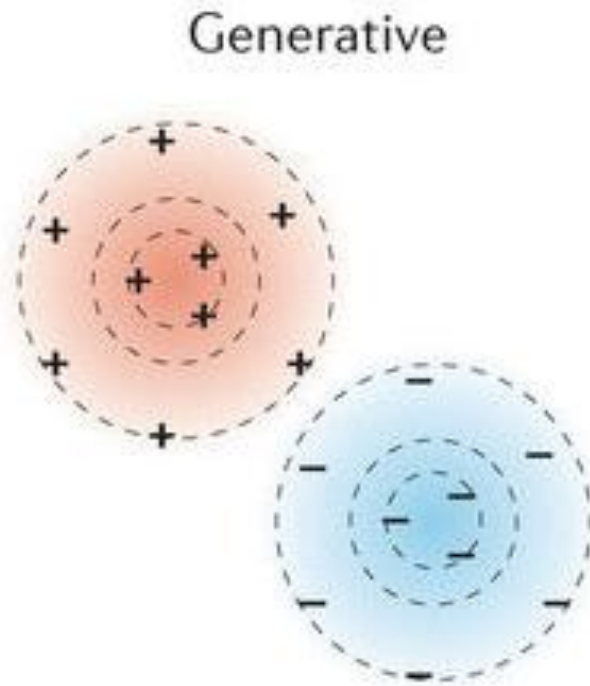
- Supervised learning
- Unsupervised learning
- Semi-supervised learning
- Reinforcement learning

Supervised learning

- Outcome measurement Y (also called dependent variable, response, target).
- Vector of p predictor measurements X (also called inputs, regressors, covariates, features, independent variables).
- In the *regression problem*, Y is quantitative (e.g price, blood pressure).
- In the *classification problem*, Y takes values in a finite, unordered set (survived/died, digit 0-9, cancer class of tissue sample).
- We have training data $(x_1, y_1), \dots, (x_N, y_N)$. These are observations (examples, instances) of these measurements.
 - Accurately predict unseen test cases.
 - Understand which inputs affect the outcome, and how.
 - Assess the quality of our predictions and inferences.

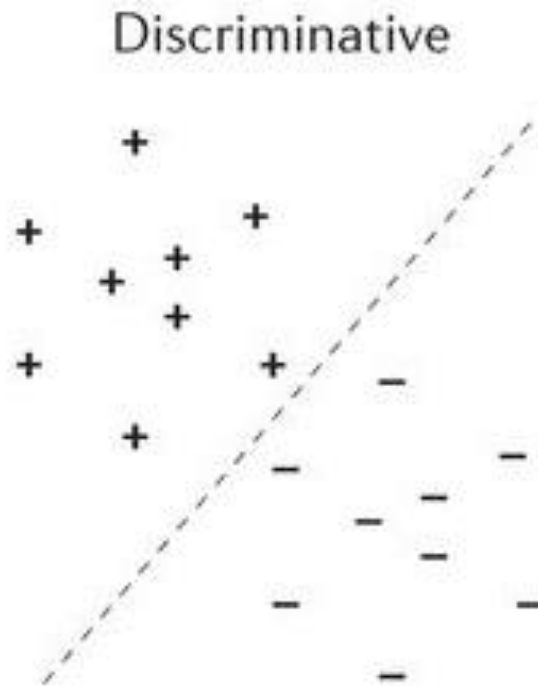
$$p(y_i | \mathbf{x}_i, \boldsymbol{\theta})$$

Generative vs. discriminative models



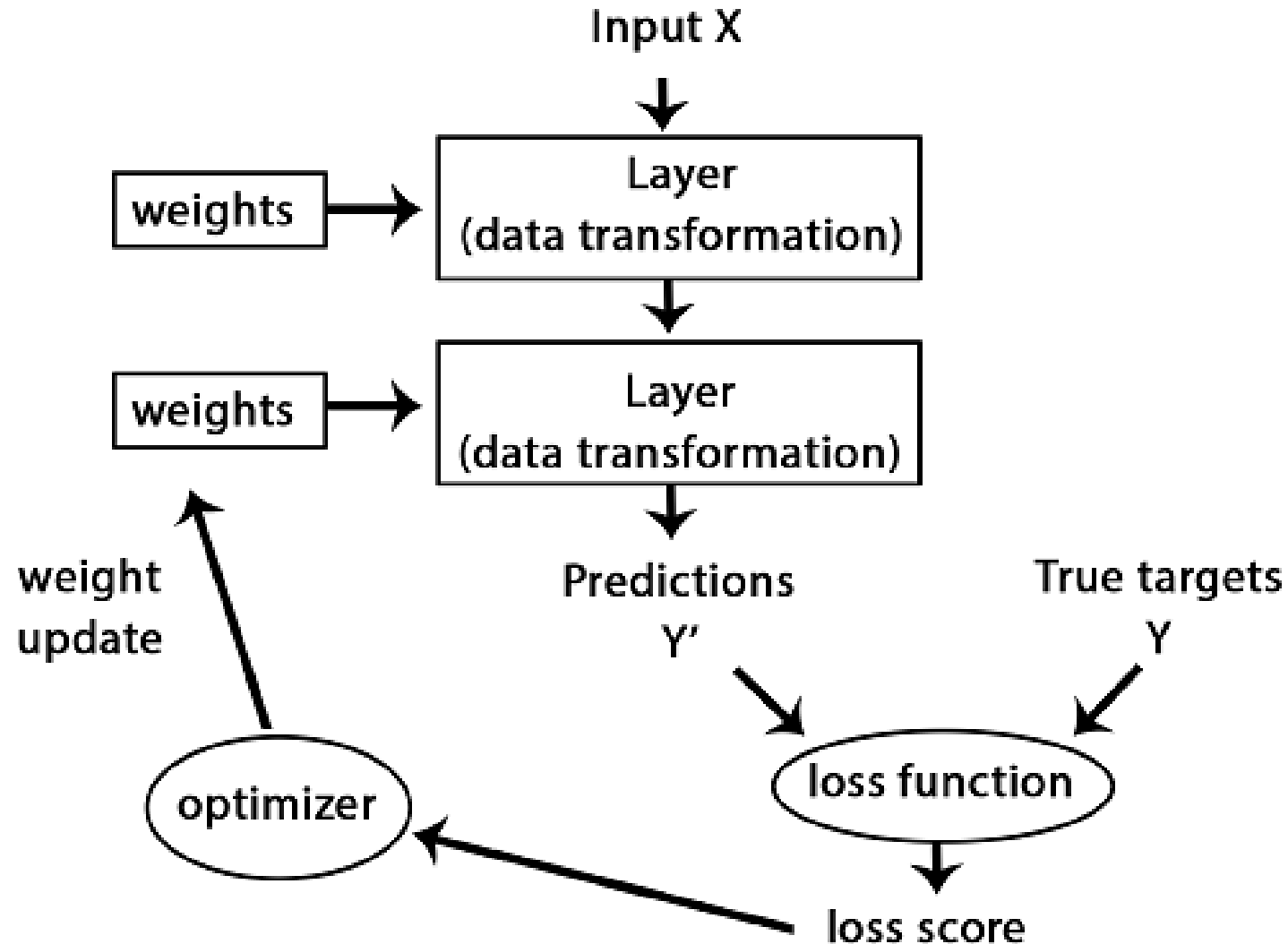
$$p(\mathbf{x}|y = c, \boldsymbol{\theta})$$

$$p(y = c|\mathbf{x}, \boldsymbol{\theta}) \propto p(\mathbf{x}|y = c, \boldsymbol{\theta})p(y = c|\boldsymbol{\theta})$$



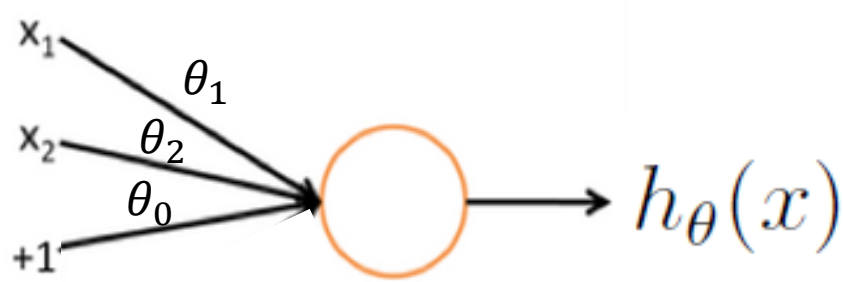
$$p(y = c|\mathbf{x}, \boldsymbol{\theta})$$

Supervised machine learning



Linear Regression

A linear artificial neuron



parameters

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 = \sum_{i=0}^n \theta_i x_i = \theta^T x,$$

Training the neuron means learning the parameters to minimize some loss

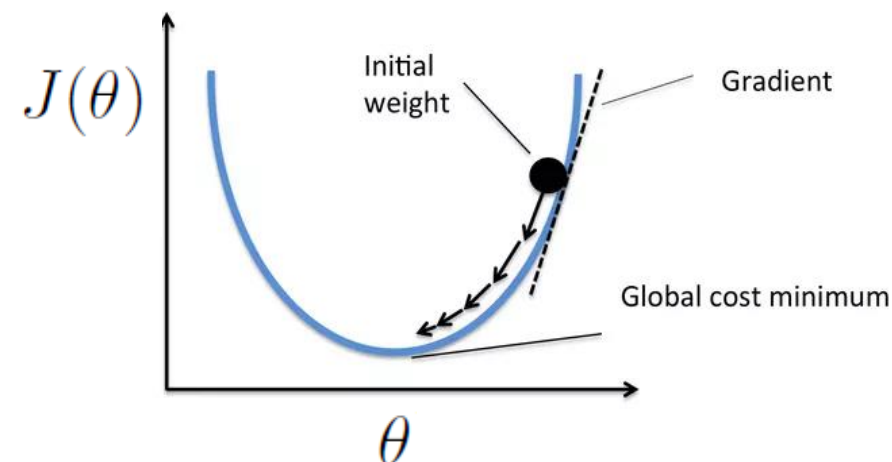
Loss function

Square loss

$$J(\theta) = \frac{1}{2} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2.$$

Gradient

$$\begin{aligned} \frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_{\theta}(x) - y)^2 \\ &= 2 \cdot \frac{1}{2} (h_{\theta}(x) - y) \cdot \frac{\partial}{\partial \theta_j} (h_{\theta}(x) - y) \\ &= (h_{\theta}(x) - y) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^n \theta_i x_i - y \right) \\ &= (h_{\theta}(x) - y) x_j \end{aligned}$$

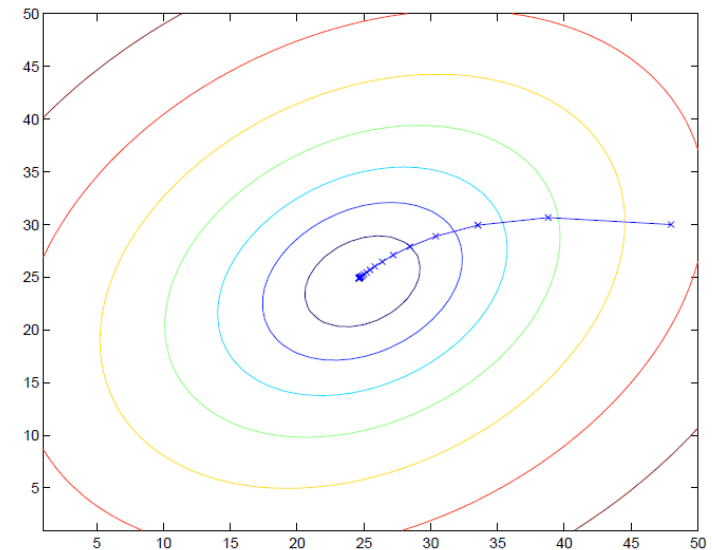
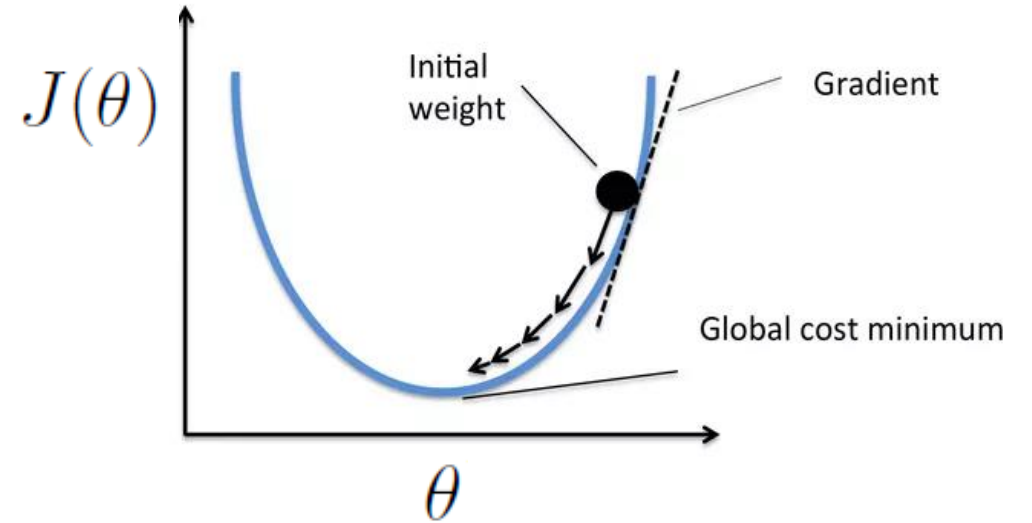


Minimize loss

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta).$$

α is called the **learning rate**.

$$\theta_j := \theta_j + \alpha (y^{(i)} - h_{\theta}(x^{(i)})) x_j^{(i)}.$$



Stochastic vs. batch gradient descent

Loop {

for i=1 to m, {

$$\theta_j := \theta_j + \alpha (y^{(i)} - h_{\theta}(x^{(i)})) x_j^{(i)} \quad (\text{for every } j).$$

}

}

Repeat until convergence {

$$\theta_j := \theta_j + \alpha \sum_{i=1}^m (y^{(i)} - h_{\theta}(x^{(i)})) x_j^{(i)} \quad (\text{for every } j).$$

}

Minimize loss in closed form

For a function $f : \mathbb{R}^{m \times n} \mapsto \mathbb{R}$ mapping from m -by- n matrices to the real numbers, we define the derivative of f with respect to A to be:

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \cdots & \frac{\partial f}{\partial A_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{m1}} & \cdots & \frac{\partial f}{\partial A_{mn}} \end{bmatrix}$$

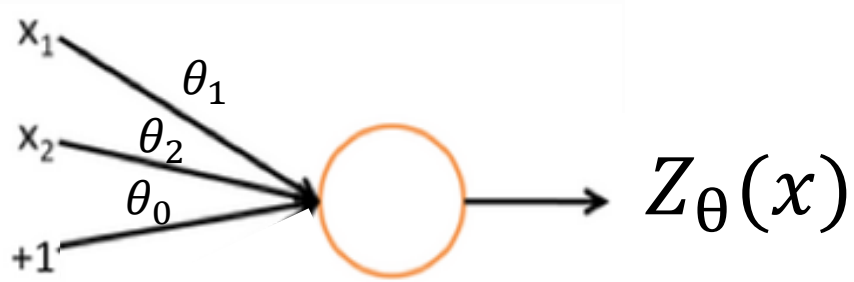
To minimize J , we set its derivatives to zero.

$$\begin{aligned} \nabla_{\theta} J(\theta) &= \nabla_{\theta} \frac{1}{2} (X\theta - \vec{y})^T (X\theta - \vec{y}) \\ &= \frac{1}{2} \nabla_{\theta} (\theta^T X^T X \theta - \theta^T X^T \vec{y} - \vec{y}^T X \theta + \vec{y}^T \vec{y}) \\ &= \frac{1}{2} \nabla_{\theta} \text{tr} (\theta^T X^T X \theta - \theta^T X^T \vec{y} - \vec{y}^T X \theta + \vec{y}^T \vec{y}) \\ &= \frac{1}{2} \nabla_{\theta} (\text{tr} \theta^T X^T X \theta - 2 \text{tr} \vec{y}^T X \theta) \\ &= \frac{1}{2} (X^T X \theta + X^T X \theta - 2 X^T \vec{y}) \\ &= X^T X \theta - X^T \vec{y} \end{aligned}$$

$$\theta = (X^T X)^{-1} X^T \vec{y}.$$

Logistic regression

A non-linear activation (logistic)



Logistic / Sigmoid

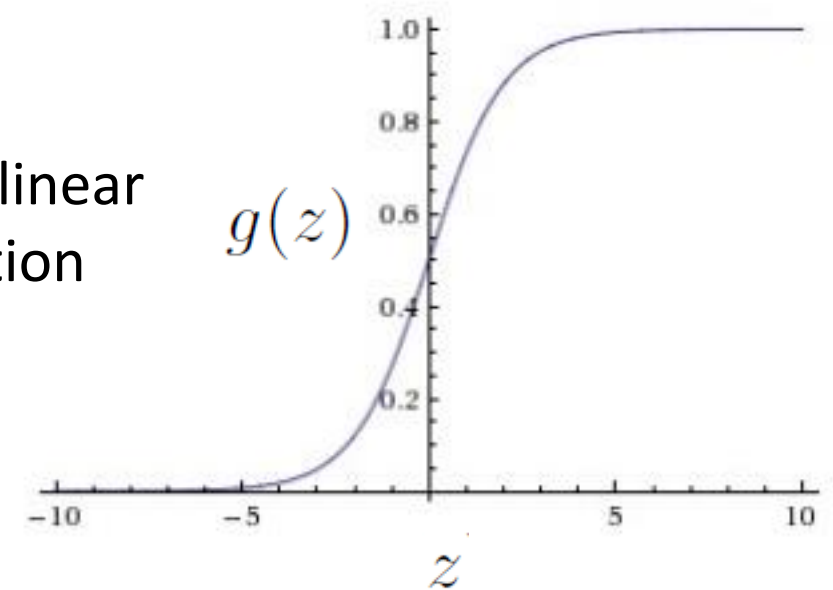
Useful for predicting probabilities

$$Z_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

$$h_{\theta}(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}},$$

Non-linear
function

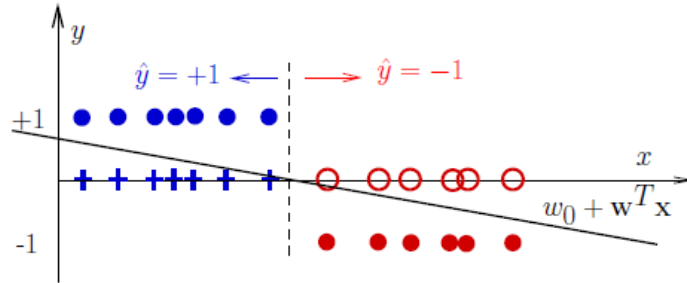


Training the neuron means learning the parameters to minimize some loss

Derivative of logistic function

$$\begin{aligned} g'(z) &= \frac{d}{dz} \frac{1}{1 + e^{-z}} \\ &= \frac{1}{(1 + e^{-z})^2} (e^{-z}) \\ &= \frac{1}{(1 + e^{-z})} \cdot \left(1 - \frac{1}{(1 + e^{-z})} \right) \\ &= g(z)(1 - g(z)). \end{aligned}$$

Logistic regression (binary classification)



- What are the drawbacks of fitting y with least squares?
 - No guarantee that y has valid values; sensitive to outliers.
 - Most importantly: no probabilistic interpretation or basis for the decision boundary.
- Optimal decision boundary is given by log-odds ratio:

$$\log \frac{p(y = 1 | \mathbf{x})}{p(y = 0 | \mathbf{x})} = 0.$$

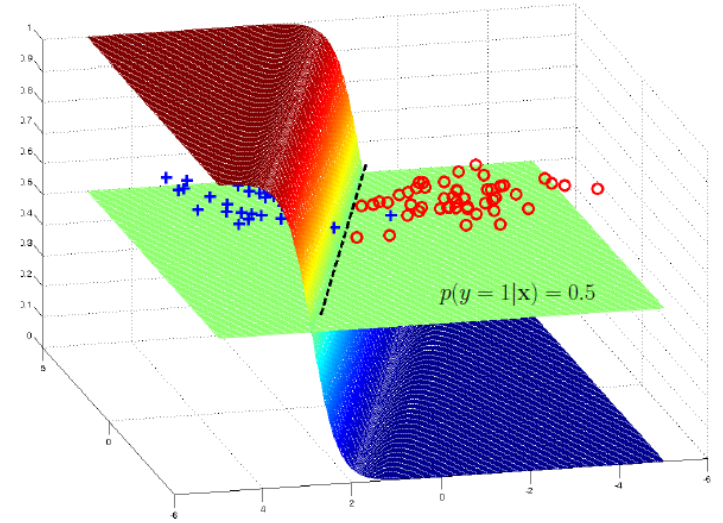
Logistic regression (binary classification)

- We can model the (unknown) decision boundary directly:

$$\log \frac{p(y = 1 | \mathbf{x})}{p(y = 0 | \mathbf{x})} = \theta^T x$$

- Since $p(y = 1 | \mathbf{x}) = 1 - p(y = 0 | \mathbf{x})$, we have (after exponentiating):

$$\begin{aligned} \frac{p(y = 1 | \mathbf{x})}{1 - p(y = 1 | \mathbf{x})} &= \exp(\theta^T x) \\ \Rightarrow \frac{1}{p(y = 1 | \mathbf{x})} &= 1 + \exp(-\theta^T x) \\ \Rightarrow p(y = 1 | \mathbf{x}) &= \frac{1}{1 + e^{-\theta^T x}} \end{aligned}$$



Logistic regression (binary classification)

$$h_{\theta}(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}},$$

$$P(y = 1 \mid x; \theta) = h_{\theta}(x)$$

$$P(y = 0 \mid x; \theta) = 1 - h_{\theta}(x)$$

$$p(y \mid x; \theta) = (h_{\theta}(x))^y (1 - h_{\theta}(x))^{1-y}$$

Likelihood for logistic regression model

$$h_{\theta}(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}},$$

$$\begin{aligned} L(\theta) &= p(\vec{y} \mid X; \theta) \\ &= \prod_{i=1}^m p(y^{(i)} \mid x^{(i)}; \theta) \\ &= \prod_{i=1}^m (h_{\theta}(x^{(i)}))^{y^{(i)}} (1 - h_{\theta}(x^{(i)}))^{1-y^{(i)}} \end{aligned}$$

Maximize log-likelihood = minimize logistic loss

$$h_{\theta}(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}},$$

$$\begin{aligned}\ell(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^m y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial \theta_j} \ell(\theta) &= \left(y \frac{1}{g(\theta^T x)} - (1 - y) \frac{1}{1 - g(\theta^T x)} \right) \frac{\partial}{\partial \theta_j} g(\theta^T x) \\ &= \left(y \frac{1}{g(\theta^T x)} - (1 - y) \frac{1}{1 - g(\theta^T x)} \right) g(\theta^T x)(1 - g(\theta^T x)) \frac{\partial}{\partial \theta_j} \theta^T x \\ &= (y(1 - g(\theta^T x)) - (1 - y)g(\theta^T x)) x_j \\ &= (y - h_{\theta}(x)) x_j\end{aligned}$$

Stochastic gradient ascent

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_{\theta}(x^{(i)}) \right) x_j^{(i)}$$

Equivalent to minimizing negative log-likelihood or the logistic loss

Multi-class classification (softmax loss)

$$\begin{aligned} p(y = i|x; \theta) &= \frac{e^{\eta_i}}{\sum_{j=1}^k e^{\eta_j}} \\ &= \frac{e^{\theta_i^T x}}{\sum_{j=1}^k e^{\theta_j^T x}} \end{aligned}$$

$$\begin{aligned} \ell(\theta) &= \sum_{i=1}^m \log p(y^{(i)}|x^{(i)}; \theta) \\ &= \sum_{i=1}^m \log \prod_{l=1}^k \left(\frac{e^{\theta_l^T x^{(i)}}}{\sum_{j=1}^k e^{\theta_j^T x^{(i)}}} \right)^{1_{\{y^{(i)}=l\}}} \end{aligned}$$

Performance measures

Measures of regression performance

- We compute the *Residual Standard Error*

$$\text{RSE} = \sqrt{\frac{1}{n-2} \text{RSS}} = \sqrt{\frac{1}{n-2} \sum_{i=1}^n (y_i - \hat{y}_i)^2},$$

where the *residual sum-of-squares* is $\text{RSS} = \sum_{i=1}^n (y_i - \hat{y}_i)^2$.

- *R-squared* or fraction of variance explained is

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}}$$

where $\text{TSS} = \sum_{i=1}^n (y_i - \bar{y})^2$ is the *total sum of squares*.

- It can be shown that in this simple linear regression setting that $R^2 = r^2$, where r is the correlation between X and Y :

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}.$$

Measures of classification performance

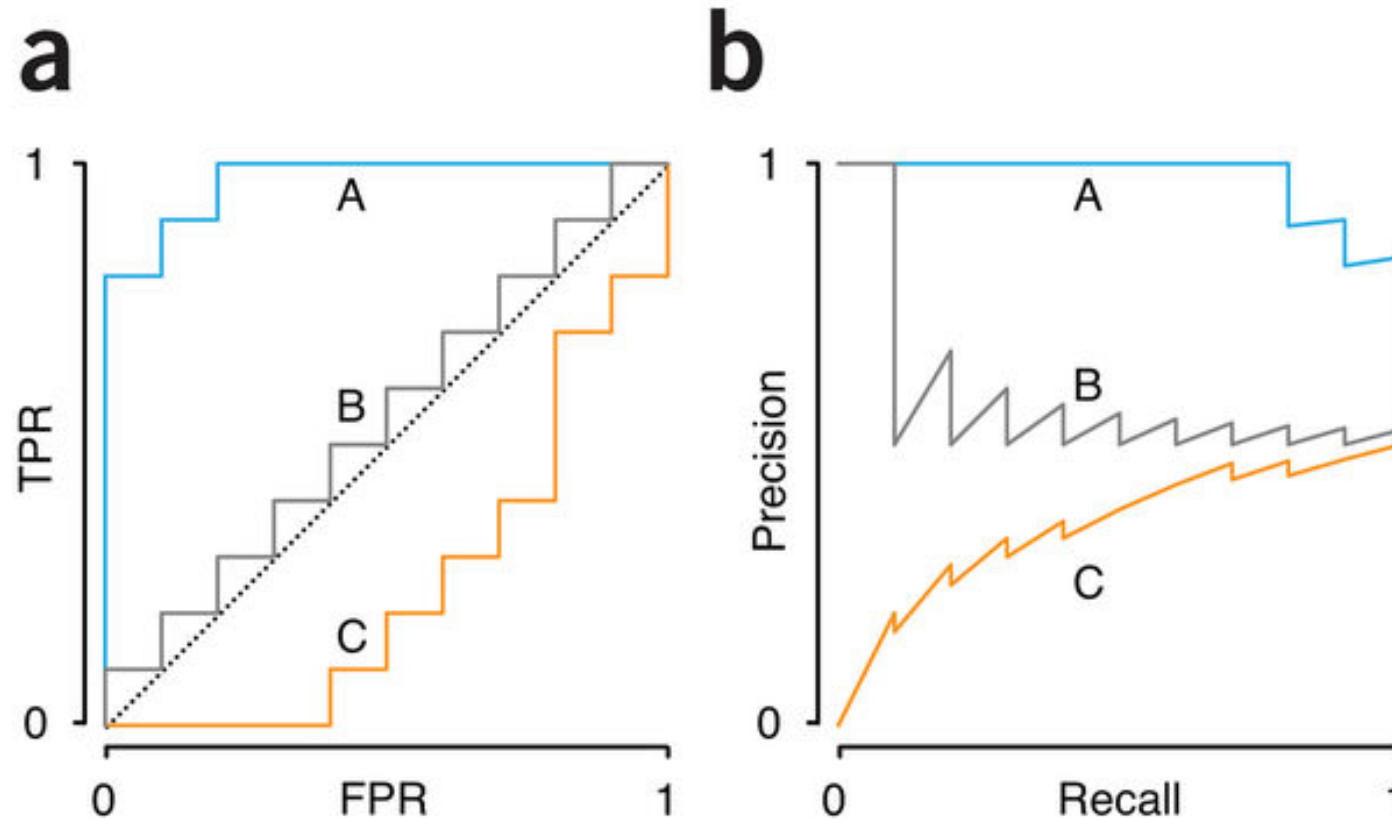
		Predicted			
		+	-		
Actual	+	TP Type II error	FN Type I error	Sensitivity (recall) TP/●	False negative rate FN/●
	-	FP Type I error	TN	False positive rate FP/●	Specificity TN/●
		Precision TP/■	False omission rate FN/■	Accuracy (TP + TN)/(● + ●)	
		FDR FP/■	Negative predictive value TN/■	F_1 score $2TP/(2TP + FP + FN)$	

Measures of classification performance

Several commonly used performance measures

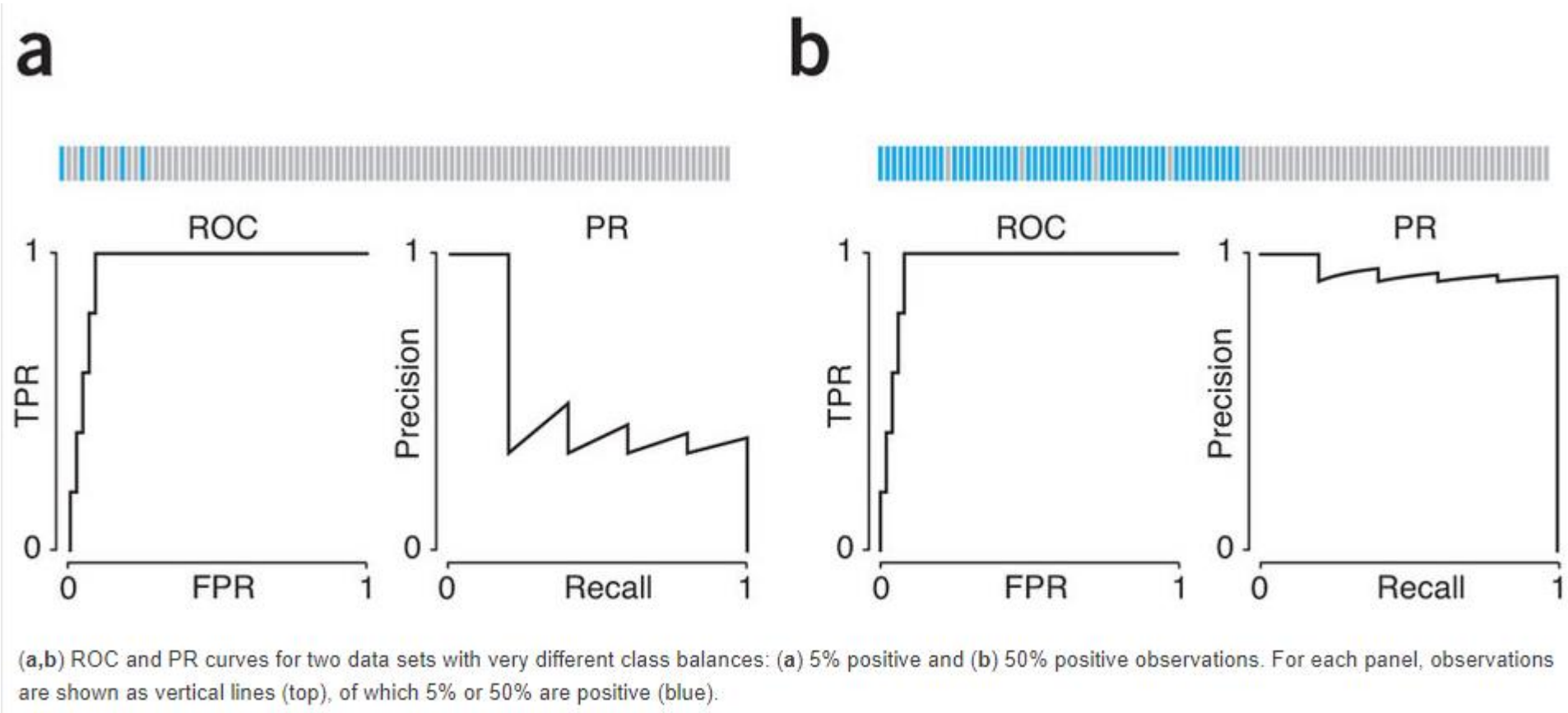
Name	Computation
Accuracy	$ACC = \frac{TP+TN}{N}$
Error rate (1-accuracy)	$ERR = \frac{FP+FN}{N}$
Balanced error rate	$BER = \frac{1}{2} \left(\frac{FN}{FN+TP} + \frac{FP}{FP+TN} \right)$
Weighted relative accuracy	$WRACC = \frac{TP}{TP+FN} - \frac{FP}{FP+TN}$
F1 score	$F1 = \frac{2*TP}{2*TP+FP+FN}$
Cross-correlation coefficient	$CC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$
Sensitivity/recall	$TPR = TP/N^+ = \frac{TP}{TP+FN}$
Specificity	$TNR = TN/N^- = \frac{TN}{TN+FP}$
1-sensitivity	$FNR = FN/N^+ = \frac{FN}{FN+TP}$
1-specificity	$FPR = FP/N^- = \frac{FP}{FP+TN}$
P.p.v. / precision	$PPV = TP/O^+ = \frac{TP}{TP+FP}$
False discovery rate	$FDR = FP/O^+ = \frac{FP}{FP+TP}$

Measures of classification performance



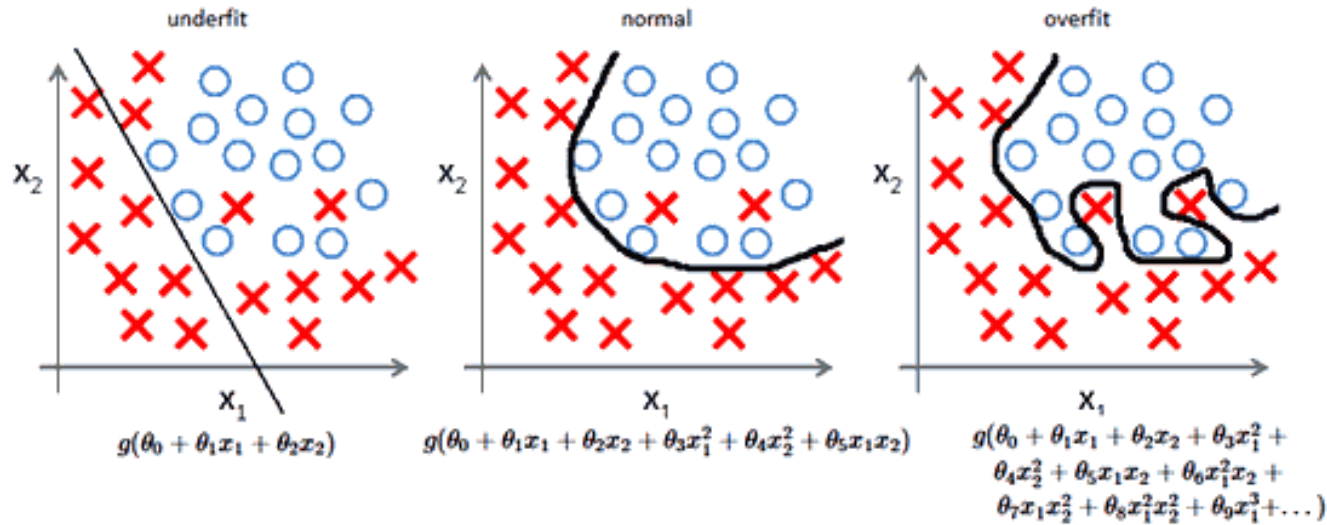
(a,b) Findings obtained with the (a) ROC, which plots the true positive rate (TPR) versus the false positive rate (FPR), and (b) PR curves. In both panels, curves depict classifiers that are (A) good, (B) similar to random classification and (C) worse than random. The expected performance of a random classifier is shown by the dotted line in a. The equivalent for the PR curve depends on the class balance and is not shown.

auROC can be misleading for imbalanced classes



Overfitting and generalization

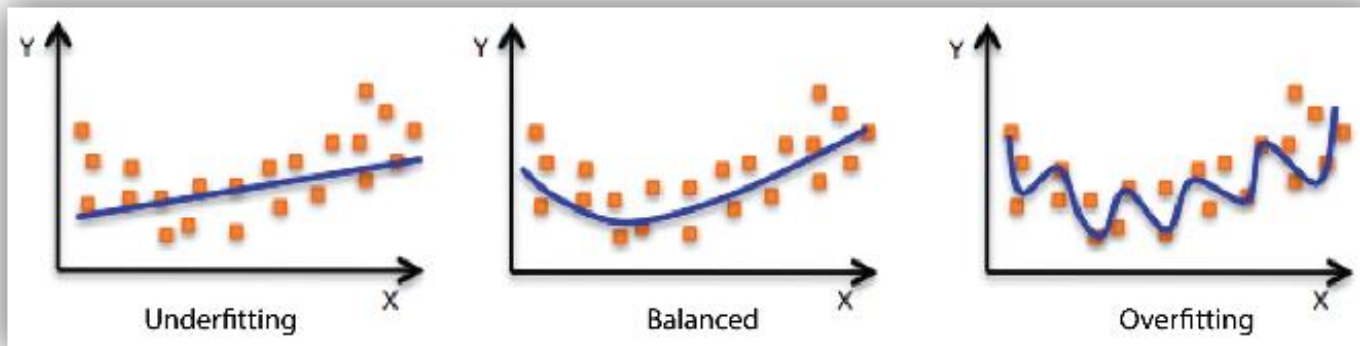
Underfitting and overfitting



More complex models (with more parameters) can always fit the 'training data' better but may do poorly on unseen data i.e. not generalize

Why?

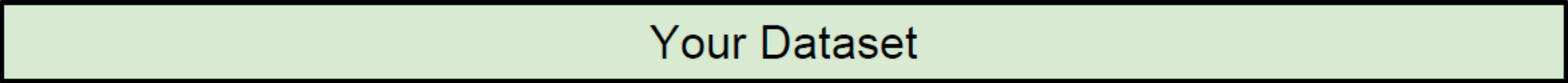
Because data (inputs and outputs) is noisy. Complex models will start fitting noise.



Evaluate performance of model on held-out test data

Idea #1: Choose hyperparameters that work best on the data


BAD: $K = 1$ always works perfectly on training data



Your Dataset

Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data




train

test

Idea #3: Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

Better!



train

validation

test

Cross-validation

Your Dataset

Idea #4: Cross-Validation: Split data into **folds**,
try each fold as validation and average the results

fold 1

fold 2

fold 3

fold 4

fold 5

test

fold 1

fold 2

fold 3

fold 4

fold 5

test

fold 1

fold 2

fold 3

fold 4

fold 5

test