Advanced Introduction to Machine Learning

— Spring Quarter, Week 2 —

https://canvas.uw.edu/courses/1372141

Prof. Jeff Bilmes

University of Washington, Seattle
Departments of: Electrical & Computer Engineering, Computer Science & Engineering
http://melodi.ee.washington.edu/~bilmes

April 6th/8th, 2020

Announcements

- HW1 (wo) ~~is~~ posted ~~this evening~~, due in 1.5 weeks.
- Virtual office hours this week, Thursday night at 10:00pm via zoom (same link as class).

Class Road Map

- W1(3/30,4/1): What is ML, Probability, Coins, Gaussians and linear regression, Associative Memories, Supervised Learning
- W2(4/6,4/8): More supervised, logistic regression, complexity and bias/variance tradeoff
- W3(4/13,4/15): Bias/Variance, Regularization, Ridge, CrossVal, Multiclass
- W4(4/20,4/22): Multiclass classification, ERM, Gen/Disc, Naïve Bayes
- W5(4/27,4/29): Lasso, Regularizers, Curse of Dimensionality
- W6(5/4,5/6): Curse of Dimensionality, Dimensionality Reduction, k-NN
- W7(5/11,5/13): k-NN, LSH, DTs, Bootstrap/Bagging, Boosting & Random Forests, GBDTs
- W8(5/18,5/20): Graphs; Graphical Models (Factorization, Inference, MRFs, BNs);
- W9(5/27,6/1): Learning Paradigms; Clustering; EM Algorithm;
- W10(6/3,6/8): Spectral Clustering, Graph SSL, Deep models, (SVMs, RL); The Future.

Last lecture is 6/8 since 5/25 is holiday (or we could just have lecture on 5/25).

Acknowledgments/References

Some of the below material was drawn from:

- Bishop, 1996.
- https://courses.cs.washington.edu/courses/cse546/18au/, https://courses.cs.washington.edu/courses/cse546/16au/, https://courses.cs.washington.edu/courses/cse546/14au/, http://cs229.stanford.edu/syllabus.html, https://ml.berkeley.edu/blog/2017/07/13/tutorial-4/

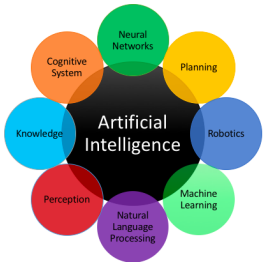
Review

- This is where each day we will be reviewing previous lecture material.

Some readings

- Matrix cookbook `https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf`
- Linear algebra notes `http://cs229.stanford.edu/section/cs229-linalg.pdf`
- Writeup on Overfitting and Underfitting on our web page (`https://canvas.uw.edu/courses/1372141`). (see in particular `https://canvas.uw.edu/courses/1372141/discussion_topics/5384617`)

Class (and Machine Learning) overview

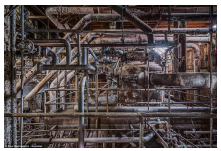


1. Introduction
- What is ML
- What is AI
- Why are we so interested in these topics right now?

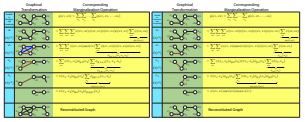
2. ML Paradigms/Concepts
- Overfitting/Underfitting, model complexity, bias/variance
- size of data, big data, sample complexity
- ERM, loss + regularization, loss functions, regularizers
- Supervised, unsupervised, and semi-supervised learning;
- reinforcement learning, RL, multi-agent, planning/control
- transfer and multi-task learning
- federated and distributed learning
- active learning, machine teaching
- self-supervised, zero/one-shot, open-set learning

3. Dealing with Features
- dimensionality reduction, PCA, LDA, MDS, T-SNE, UMAP
- Locality sensitive hashing (LSH)
- feature selection
- feature engineering
- matrix factorization & feature engineering
- representation learning

4. Evaluation
- accuracy/error, precision/recall, ROC, likelihood/posterior, cost/utility, margin
- train/eval/test data splits
- n-fold cross validation
- method of the bootstrap



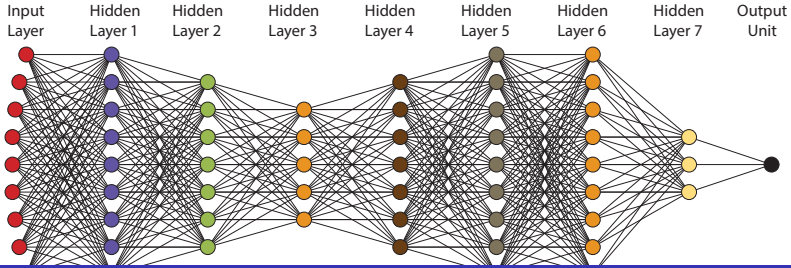
5. Optimization Methods
- Unconstrained Continuous Optimization: (stochastic) gradient descent (SGD), adaptive learning rates, conjugate gradient, 2nd order Newton
- Constrained Continuous Optimization : Frank-Wolf (conditional gradient descent), projected gradient, linear, quadratic, and convex programming
- Discrete optimization - greedy, beam search, branch-and-bound, submodular optimization.



6. Inference Methods
- probabilistic inference
- MLE, MAP
- belief propagation
- forward/backpropagation
- Monte Carlo methods

7. Models & Representation
- linear least squares, linear regression, logistic regression, sparsity, ridge, lasso
- generative vs. discriminative models
- Naive Bayes
- k-nearest neighbors
- clustering, k-means, k-mediods, EM & GMMs, single linkage
- decision trees and random forests
- support vector machines, kernel methods, max margin
- perceptron, neural networks, DNNs
- Gaussian processes
- Bayesian nonparametric methods
- ensemble methods
- the bootstrap, bagging, and boosting
- graphical models
- time-series, HMMs, DBNs, RNNs, LSTMs, Attention, Transformers
- structured prediction
- grammars (as in NLP)

12. Other Techniques
- compressed sensing
- submodularity, diversity/homogeneity modeling

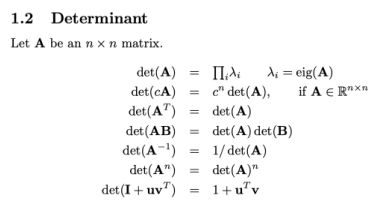


8. Philosophy, Humanity, Spirituality
- artificial intelligence (AI)
- artifical general intelligence (AGI)
- artificial intelligence vs. science fiction

9. Applications
- computational biology
- social networks
- computer vision
- speech recognition
- natural language processing
- information retrieval
- collaborative filtering/matrix factorization

10. Programming
- python
- libraries (e.g., NumPy, SciPy, matplotlib, scikit-learn (sklearn), pytorch, CNTK, Theano, tensorflow, keras, H2O, etc.
- HPC: C/C++, CUDA, vector processing

11. Background
- linear algebra
- multivariate calculus
- probability theory and statistics
- information theory
- mathematical (e.g., convex) optimization



1.2 Determinant

Let $\mathbf{A}$ be an $n \times n$ matrix.

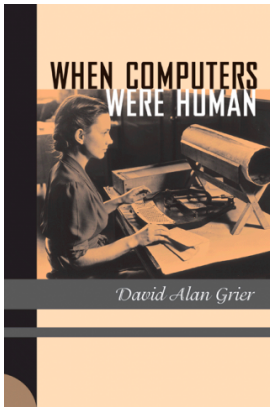
$$\begin{aligned}
\det(\mathbf{A}) &= \textstyle\prod_i \lambda_i \qquad \lambda_i = \mathrm{eig}(\mathbf{A}) \\
\det(c\mathbf{A}) &= c^n \det(\mathbf{A}), \qquad \text{if } \mathbf{A} \in \mathbb{R}^{n\times n} \\
\det(\mathbf{A}^T) &= \det(\mathbf{A}) \\
\det(\mathbf{AB}) &= \det(\mathbf{A})\det(\mathbf{B}) \\
\det(\mathbf{A}^{-1}) &= 1/\det(\mathbf{A}) \\
\det(\mathbf{A}^n) &= \det(\mathbf{A})^n \\
\det(\mathbf{I} + \mathbf{uv}^T) &= 1 + \mathbf{u}^T\mathbf{v}
\end{aligned}$$

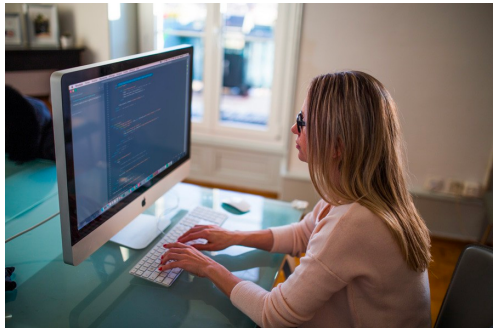
Strategy

Strategy for the next period of time.

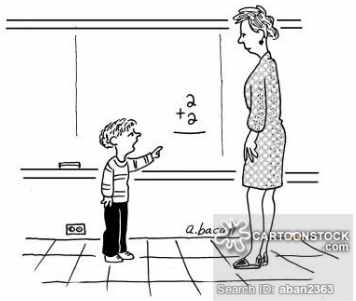
- For some topic in (2)
  - for subtopic in subset of (6) relevant to topic.

Traditional Computer Programming vs. ML



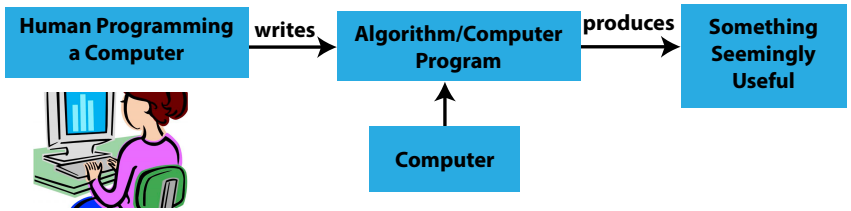


"It's a discipline note from my teacher. I solved the math problems in my head and I was supposed to use a calculator."



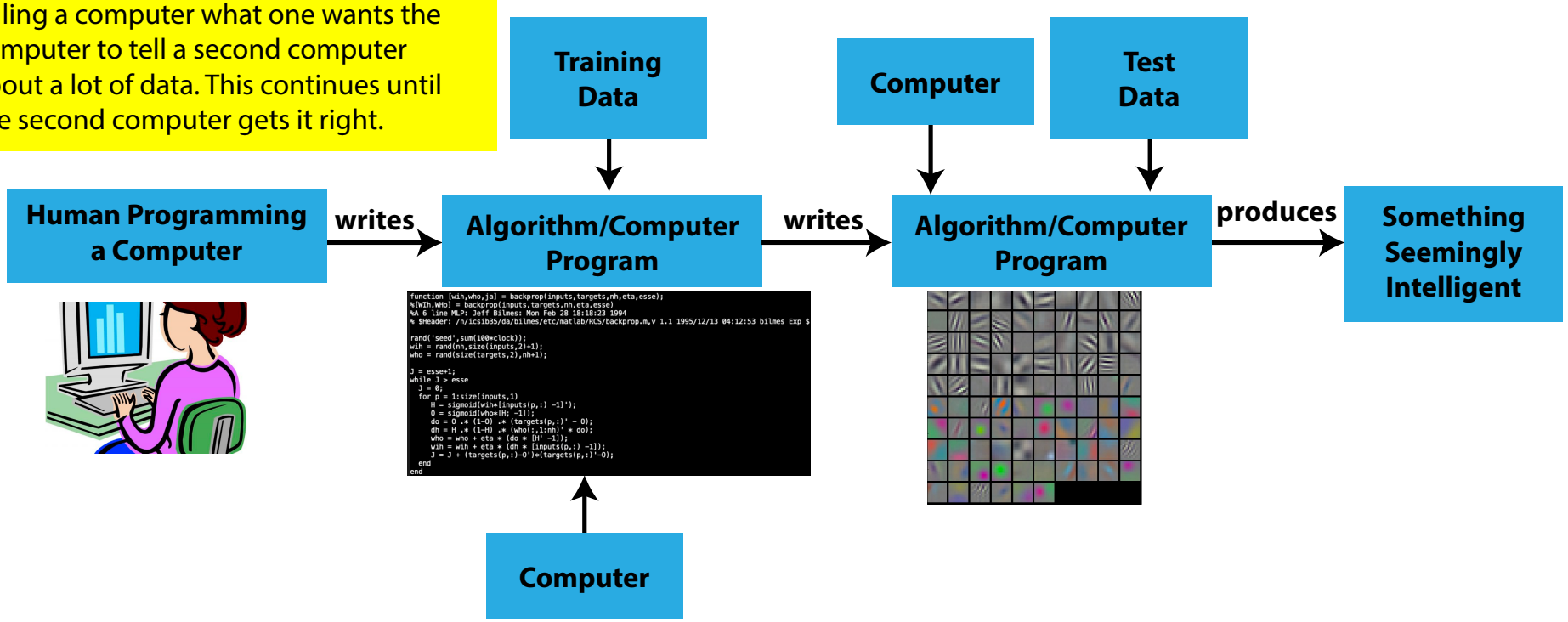
"Rather than learning how to solve that, shouldn't we be learning how to operate software that can solve that problem?"

Let us change our traditional attitude to the construction of programs. Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do. -- **Donald Knuth**

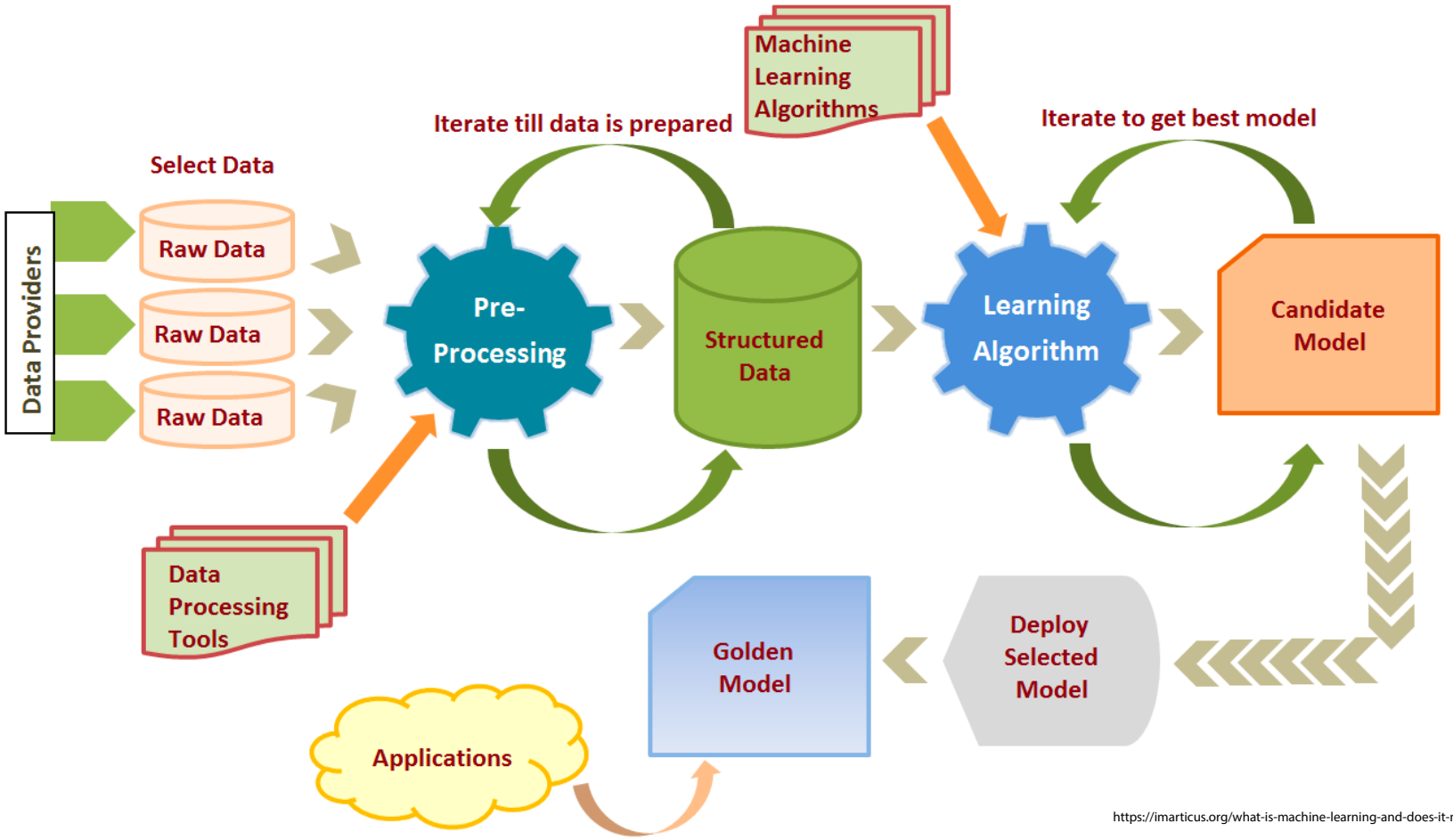


Traditional Computer Programming vs. ML

Machine Learning is the art of repeatedly telling a computer what one wants the computer to tell a second computer about a lot of data. This continues until the second computer gets it right.



Traditional Computer Programming vs. ML



other defs of ML: `https://www.kdnuggets.com/2018/12/essence-machine-learning.html`

Probability and Uncertainty

- Key point: the world is a complicated place, we cannot know everything, and even what we think we know we can (nor should) not always be certain. Uncertainty abounds!
- Need a representation of uncertainty.
- Probability has a precise mathematical definition (Kolmogorov axioms), but we use it in deference to the inevitable uncertainty surrounding all decisions.
- Simple and subjective working definition:

$$\text{probability} = \frac{\text{number of cases something happened}}{\text{number of total cases}}. \tag{1.2}$$

- Good for repeatable measurable events (e.g., coins flips, dice, etc.).
- Harder for future events (probability it will rain tomorrow, probability Manchester City wins Liverpool, etc.).
- Despite shortcomings, used as representation of uncertainty/certainty (i.e., probability that image x contains face of person y).
- Machine learning often strives for the "best" probabilities in data using learning algorithms.

Coin Flipping and ML

- $\mathcal{D} = \{b_1, b_2, \ldots, b_n\}$ is series of n independent and identical coin flips, $b_i \in \{\mathsf{H}, \mathsf{T}\}$.
- $k = |\{i : b_i = \mathsf{H}\}|$ is the count of the number of heads in $\mathcal{D}$
- How true, or likely, is it that θ is probability of heads?

$$\Pr(\mathcal{D}|\theta) = \theta^k(1-\theta)^{n-k} = \underline{\text{Likelihood}} \text{ of } \mathcal{D} \text{ given } \theta \tag{1.2}$$

- How to find the most likely explanation of $\mathcal{D}$? Maximum likelihood

$$\hat{\theta}_{\mathsf{MLE}} = \operatorname*{argmax}_{\theta \in [0,1]} \Pr(\mathcal{D}|\theta) = \operatorname*{argmax}_{\theta \in [0,1]} \log \Pr(\mathcal{D}|\theta) \tag{1.3}$$

- How to find $\hat{\theta}_{\mathsf{MLE}}$, calculus, $\frac{\partial}{\partial \theta} \log \Pr(\mathcal{D}|\theta) = 0$ leads to

$$\hat{\theta}_{\mathsf{MLE}} = k/n \tag{1.4}$$

- Thus, computing k and dividing by n is a simple way to learn!

Learning Gaussians

- Given the data sample $\mathcal{D}$ without knowing μ, C, how likely is the sample under some hypothesized parameters $\tilde{\mu}, \tilde{C}$.

$$\log \Pr(\mathcal{D}|\tilde{\mu}, \tilde{C}) = \sum_{i=1}^{n} \log \Pr(x_i|\tilde{\mu}, \tilde{C}) \tag{1.3}$$

$$\triangleq \log \underline{\text{Likelihood}} \text{ of } \mathcal{D} \text{ given } \tilde{\mu}, \tilde{C} \tag{1.4}$$

- How to find the most likely explanation of $\mathcal{D}$? Maximum likelihood

$$[\hat{\mu}_{\text{MLE}}, \hat{C}_{\text{MLE}}] = \underset{\mu \in \mathbb{R}^m, C \in \mathbb{P}(m)}{\text{argmax}} \log \Pr(\mathcal{D}|\mu, C) \tag{1.5}$$

- How to find MLE quantities, again calculus, $\frac{\partial}{\partial \mu} \log \Pr(\mathcal{D}|\mu, C) = 0$ and $\frac{\partial}{\partial C} \log \Pr(\mathcal{D}|\mu, C) = 0$ leads to

$$\mu_{\text{MLE}} = \frac{1}{n} \sum_{i=1}^{n} x_i \text{ and } C_{\text{MLE}} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu_{\text{MLE}})(x_i - \mu_{\text{MLE}})^{\mathsf{T}} \tag{1.6}$$

Associations and Associative Memories

- Associative memory, auto-associative memory, or hetero-associative memory. In general, associate $x \in \mathcal{X}$ to $y \in \mathcal{Y}$ via $h : \mathcal{X} \to \mathcal{Y}$.
- Examples: memory subsystem (separate address for each $x \in \mathcal{X}$), data structures like hash tables, or red-black trees, etc.
- Often $\mathcal{X}$, $\mathcal{Y}$ is very large, and we have only a sample associations $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ where $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$ where $n \ll |\mathcal{X}|$.
- We want to build an associative memory that works even outside of $\mathcal{D}$. That is, even for $x \notin \{x : x = x_i \text{ for some } i \in [n], (x_i, y_i) \in \mathcal{D}\}$.
- Why? $\mathcal{D}$ might not be complete, variation, noise, or possible data corruption not fully captured in $\mathcal{D}$. Also, $\mathcal{X}$ might be infinitely large.

Associations and Associative Memories

- Machine learning: Write an algorithm that, given large enough $\mathcal{D}$, produces a program h that generalizes (works) well on unseen samples. Respond reasonably to variation, noise, data corruption (be robust). Do this computationally as efficiently as possible, and (ideally) understand it mathematically.
- Boils down to finding a good $h : \mathcal{X} \to \mathcal{Y}$ that can do the mapping (association). Sometimes we choose some $h \in \mathcal{H}$ where $\mathcal{H}$ is large collection of possible associators. More frequently, h is parameterized via some parameters θ and we find a good θ leading to h_θ.
- Many ways to do this, depends on nature of $\mathcal{X}, \mathcal{Y}$, how big the data is (number of samples n), and available resources (compute, core machine memory/RAM, storage/disk, communication (latency/bandwidth), time, money, energy usage).
- Often, $x \in \mathbb{R}^m$ is an m-dimensional vector of features. In general, x is known as a feature vector.

Statistical parameter estimation

- Training data $\mathcal{D} = \left\{(x^{(i)}, y^{(i)})\right\}_{i=1}^{n}$ where $(x^{(i)}, y^{(i)}) \sim p(x, y)$ are drawn from some distribution, $x^{(i)} \in \mathbb{R}^m$ and $y^{(i)} \in \mathbb{R}$.
- $x^{(i)}$ is m-dimensional column vector of features, $y^{(i)}$ is scalar.
- Goal: find $h_\theta : \mathcal{X} \to \mathcal{Y}$ with minimum error, where

$$\text{Error}_i = e_i = h_\theta(x^{(i)}) - y^{(i)} \tag{1.3}$$

$$E[e^2] = E_{p(x,y)}[(h_\theta(x) - y)^2] = \int p(x, y)(h_\theta(x) - y)^2 dxdy \tag{1.4}$$

$$= \int p(x) \int (h_\theta(x) - y)^2 p(y|x) dydx \tag{1.5}$$

  and $\theta \in \mathbb{R}^m$ is a parameter vector, $\theta = (\theta_1, \theta_2, \ldots, \theta_m)$, $\theta_i \in \mathbb{R}$.
- Taking derivatives and setting to zero, we get best solution:

$$h_\theta(x) = \int y p(y|x) dy = E[Y|x] = \text{ best association.} \tag{1.6}$$

- This assumes we have the distribution p and also the resources to compute $E[Y|x]$.

Linear estimator: Objective Optimization

-
- Recall, $h_\theta(x) \triangleq \theta^\intercal x$ is parameterized by parameters θ so

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})^2 \tag{1.3}$$

- Taking derivative of error objective $J(\theta)$ w.r.t. θ and set to zero gets:

$$\frac{\partial J}{\partial \theta} = \frac{2}{n}\sum_{n=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})\frac{\partial h_\theta(x^{(i)})}{\partial \theta} = 0 \tag{1.4}$$

- Linear $h_\theta(x) = x^\intercal\theta$ assumption, yields $\frac{\partial h_\theta(x^{(i)})}{\partial \theta} = x^{(i)}$.

Linear Least Squares

- This gives objective to be minimized (smallest, or least of the sum of squares of the errors).

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{2}{n}\sum_{n=1}^{n}(x^{(i)\mathsf{T}}\theta - y^{(i)})x^{(i)} = 0 \tag{1.3}$$

- We simplify this a bit by defining matrices associated with these quantities. First define a $n \times m$ design matrix X and length-n column vector $\vec{y}$

$$X = \begin{pmatrix} — & x^{(1)\mathsf{T}} & — \\ — & x^{(2)\mathsf{T}} & — \\ & \vdots & \\ — & x^{(n)\mathsf{T}} & — \end{pmatrix}, \text{ and } \vec{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \tag{1.4}$$

- Objective Equation (**??**), equivalent matrix-vector form:

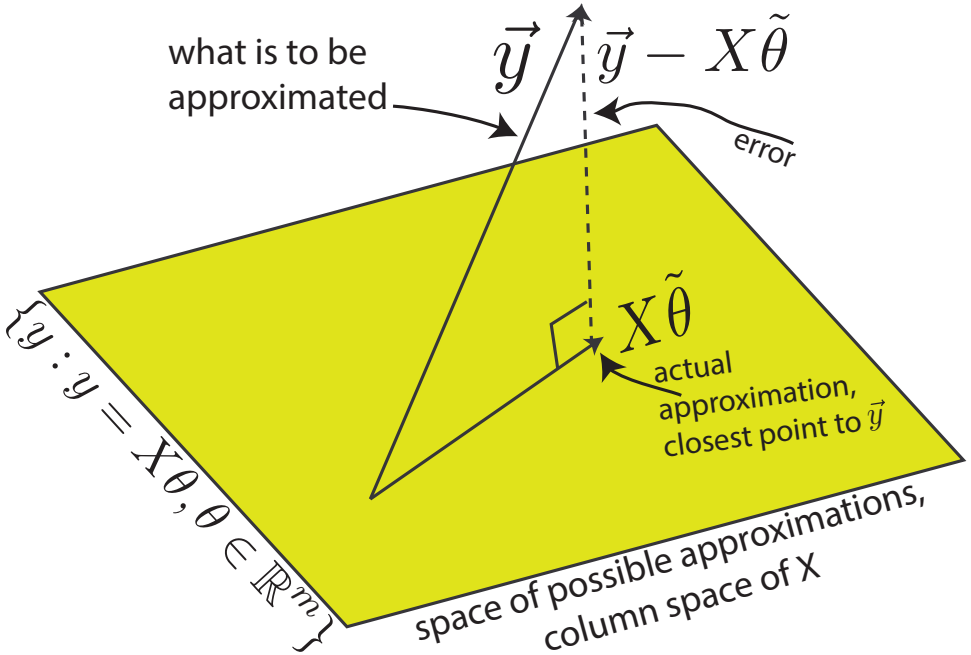
$$J(\theta) = \frac{1}{2}(X\theta - \vec{y})^{\mathsf{T}}(X\theta - \vec{y}) \tag{1.5}$$

Normal Equations

- With this, we get the "normal equations"

$$\nabla_\theta J(\theta) = X^\intercal(X\theta - \vec{y}) = \vec{0} \tag{1.3}$$

  i.e., modeling $\vec{y}$ to be in column space of matrix X (linear combinations of columns of X), when $\vec{y}$ is being approximated by $X\theta$.
- Called normal equations because column space of X is orthogonal to the residual error $E = (\vec{y} - X\theta)$, giving solution $\theta = \tilde{\theta}$ as shown.



If $X^\intercal X$ invertible (typical if $n \gg m$), solution has form:

$$\tilde{\theta} = (X^\intercal X)^{-1} X^\intercal \vec{y}$$

where $(X^\intercal X)^{-1} X^\intercal$ is known as the Moore-Penrose pseudo-inverse of matrix X.

Gradient Descent, Batch Gradient Descent

- Gradient updates for all elements of θ at the same time and for sample pair $(x^{(i)}, y^{(i)})$

$$\theta \leftarrow \theta + \alpha(y^{(i)} - h_\theta(x^{(i)}))x^{(i)} = \theta + \alpha(y^{(i)} - \theta^\intercal x^{(i)})x^{(i)} \tag{1.9}$$

  move θ in the direction of $x^{(i)}$ weighted by $\alpha(y^{(i)} - h_\theta(x^{(i)})) \in \mathbb{R}$, α times the error.
- Called LMS (least mean squares) update rule, also called Widrow-Hoff (early NN folks) learning rule.
- Batch Gradient Descent uses $J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})^2$, and since the gradient is a linear operator, this yields the following:

Algorithm 2: Batch Gradient descent learning

Input : Training data $\mathcal{D}$, learning rate α, initial parameter estimate θ

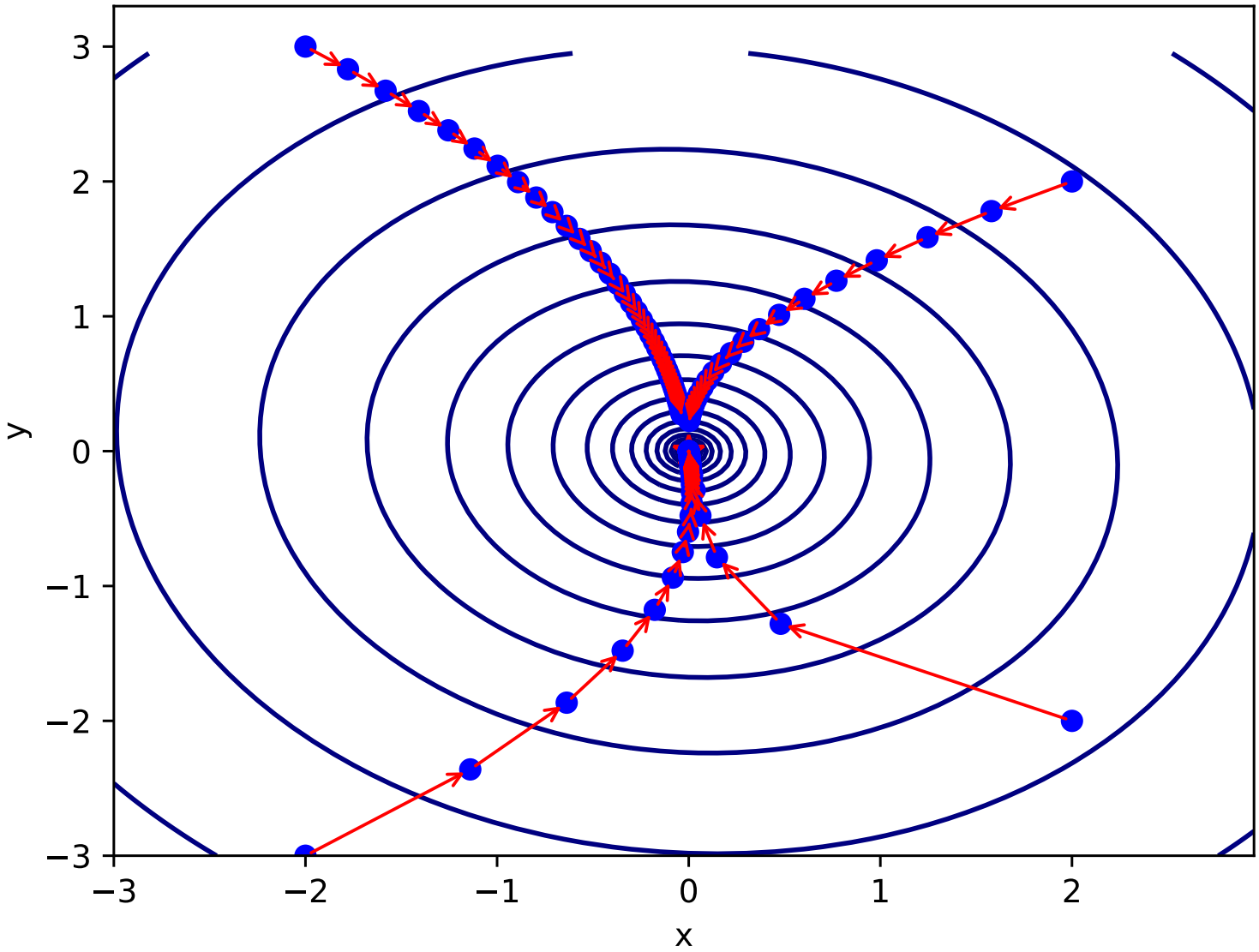
Output: Learnt model parameters θ

for $t = 1, \cdots, T$ **do**

  $\theta \leftarrow \theta + \alpha\sum_{i=1}^{n}(y^{(i)} - h_\theta(x^{(i)}))x^{(i)}$

Return : the final parameters θ

More visualization: Batch Gradient Descent



Incremental and Stochastic Gradient Descent

Algorithm 3: Incremental Gradient Descent (IGD) learning

Input : Training data $\mathcal{D}$, learning rate α, initial parameter estimate θ

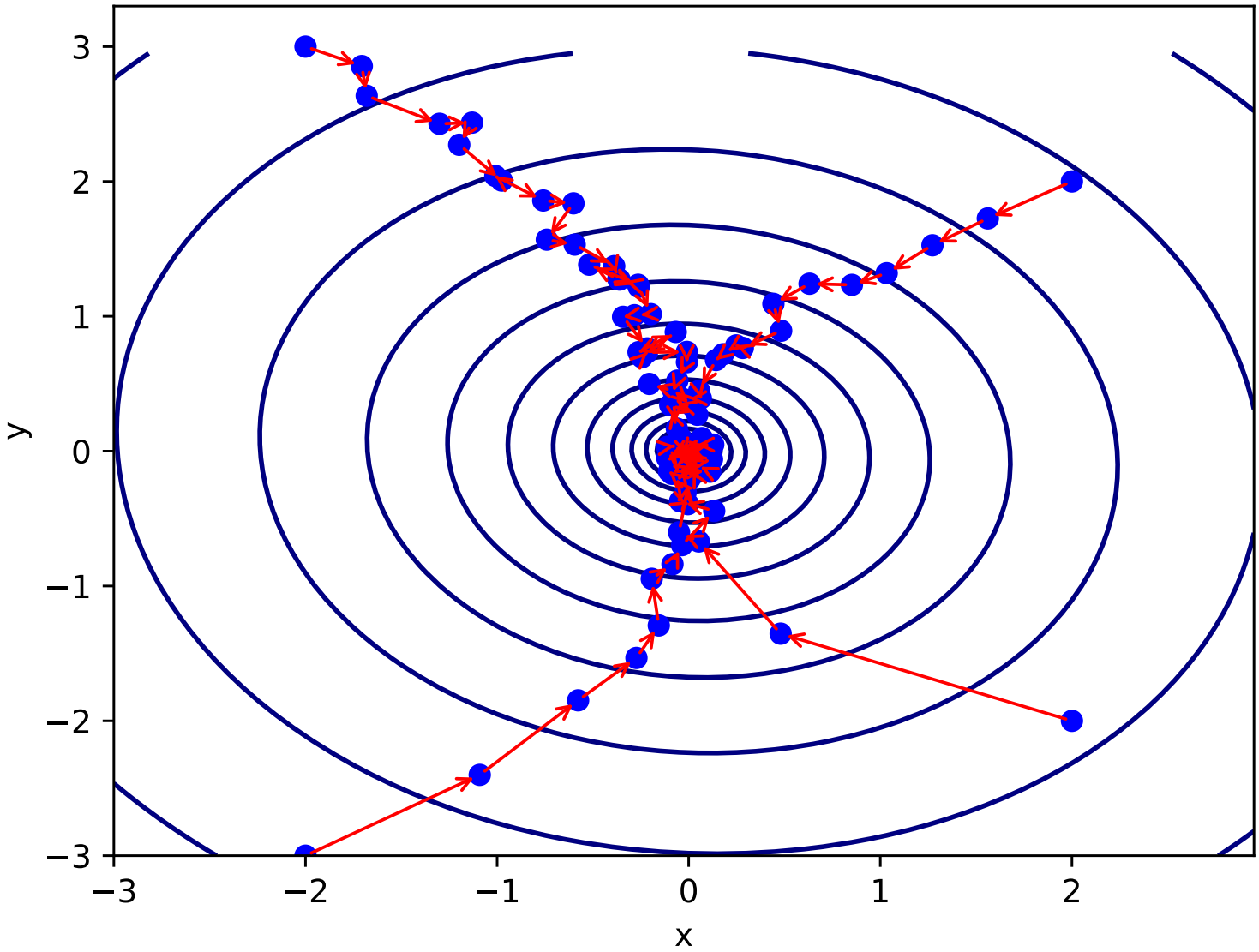
Output: Learnt model parameters θ

for $t = 1, \cdots, T$ **do**
  for $i = 1, \cdots, n$ **do**
    $\theta \leftarrow \theta + \alpha(y^{(i)} - h_\theta(x^{(i)}))x^{(i)}$

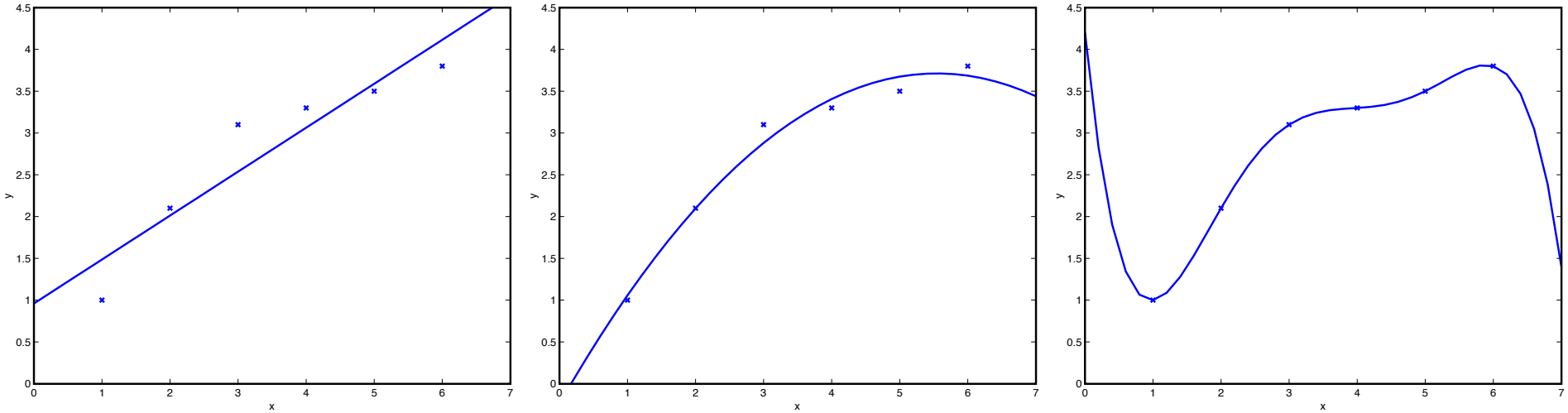
Return : the final parameters θ

Optimization folks (e.g., Bertsekas) call this incremental gradient methods. It is called Stochastic Gradient Descent (SGD) if we randomize (with or without replacement) the order of the data items.

More visualization: Stochastic Gradient Descent

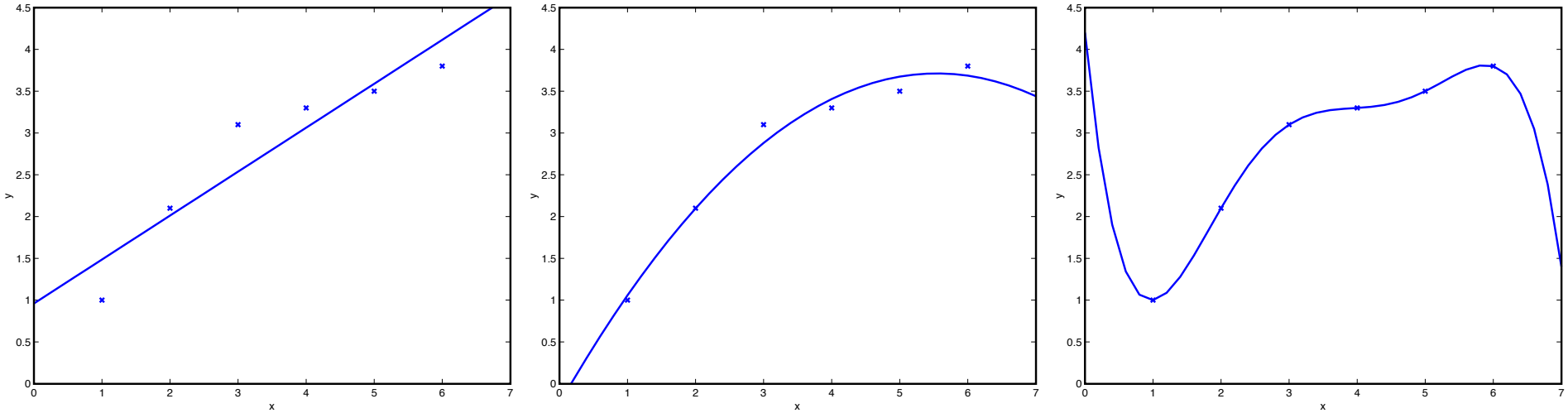


Underfitting vs. Overfitting



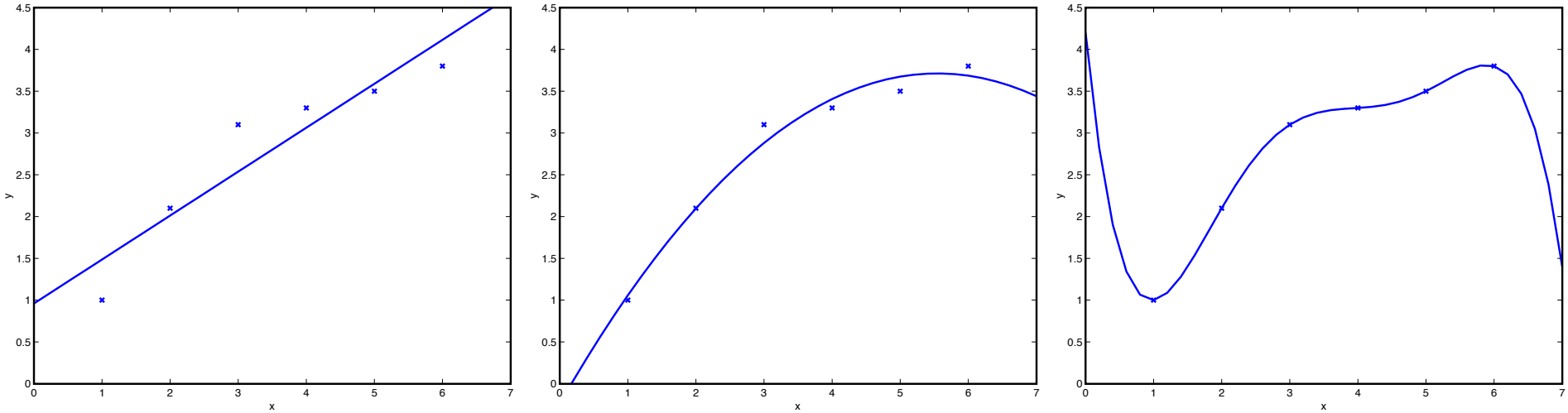
- Fit a model with various input features, values of powers of x, goal is to predict y based on xy-pair samples $\mathcal{D} = \{(x^{(i)}, y^{(i)}\}_i$.

Underfitting vs. Overfitting



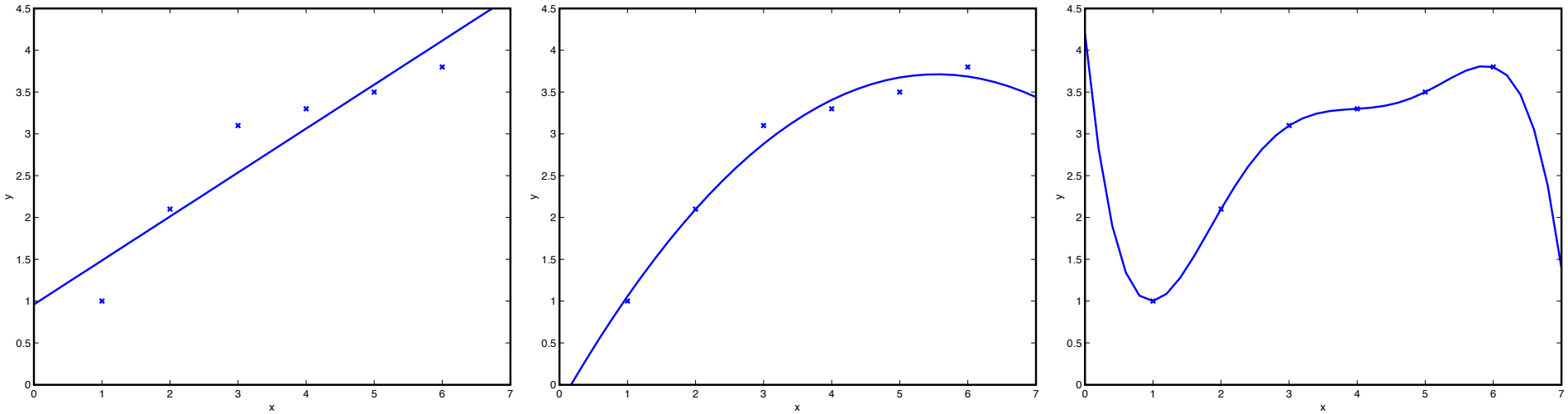
- Fit a model with various input features, values of powers of x, goal is to predict y based on xy-pair samples $\mathcal{D} = \{(x^{(i)}, y^{(i)}\}_i$.
- Fit models: left $y = \theta_0 + \theta_1 x$; middle $y = \theta_0 + \theta_1 x + \theta_2 x^2$; right $y = \sum_{j=0}^{5} \theta_j x^j$.

Underfitting vs. Overfitting



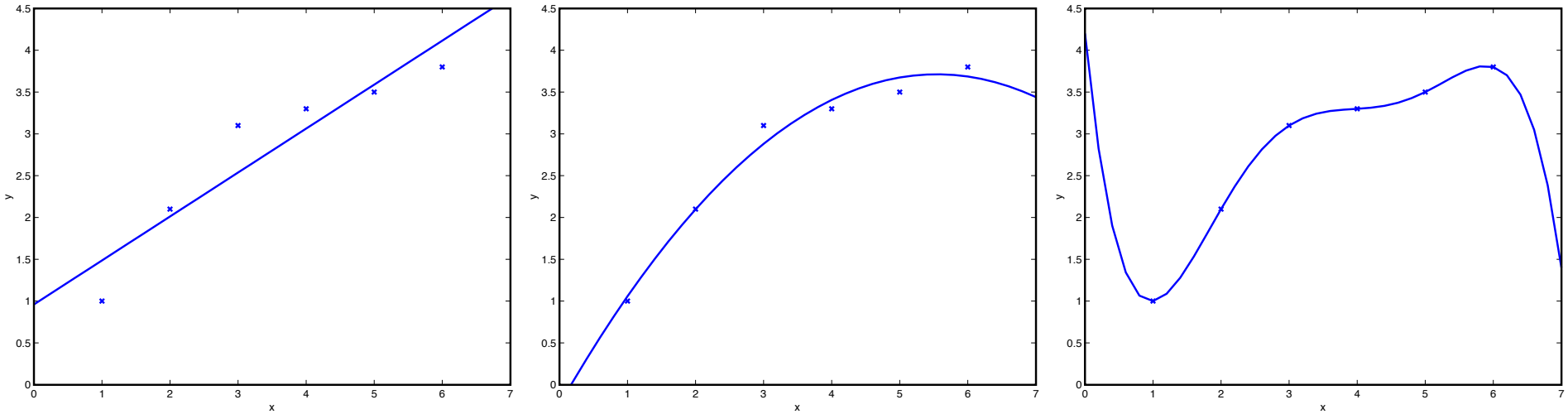
- Fit a model with various input features, values of powers of x, goal is to predict y based on xy-pair samples $\mathcal{D} = \{(x^{(i)}, y^{(i)}\}_i$.
- Fit models: left $y = \theta_0 + \theta_1 x$; middle $y = \theta_0 + \theta_1 x + \theta_2 x^2$; right $y = \sum_{j=0}^{5} \theta_j x^j$.
- Both left and right plots poorly fit the data, but they are poor for different reasons.

Underfitting vs. Overfitting



- Fit a model with various input features, values of powers of x, goal is to predict y based on xy-pair samples $\mathcal{D} = \{(x^{(i)}, y^{(i)}\}_i$.
- Fit models: left $y = \theta_0 + \theta_1 x$; middle $y = \theta_0 + \theta_1 x + \theta_2 x^2$; right $y = \sum_{j=0}^{5} \theta_j x^j$.
- Both left and right plots poorly fit the data, but they are poor for different reasons. The left could be underfitting, and the right could be overfitting.

Underfitting vs. Overfitting



- Fit a model with various input features, values of powers of x, goal is to predict y based on xy-pair samples $\mathcal{D} = \{(x^{(i)}, y^{(i)}\}_i$.
- Fit models: left $y = \theta_0 + \theta_1 x$; middle $y = \theta_0 + \theta_1 x + \theta_2 x^2$; right $y = \sum_{j=0}^{5} \theta_j x^j$.
- Both left and right plots poorly fit the data, but they are poor for different reasons. The left could be underfitting, and the right could be overfitting. The center plot looks better.

Overfitting definition (T. Mitchell)

- We say that a hypothesis overfits the training examples if some other hypothesis that fits the training examples less well actually performs better over the entire distribution of instances (i.e., including instances beyond the training set).

Overfitting definition (T. Mitchell)

- We say that a hypothesis overfits the training examples if some other hypothesis that fits the training examples less well actually performs better over the entire distribution of instances (i.e., including instances beyond the training set).

Definition 2.3.1 (overfitting)

Given a hypothesis space $\mathcal{H}$, a hypothesis $h \in \mathcal{H}$ is said to overfit the training data if there exists some alternative hypothesis $h' \in \mathcal{H}$, such that h has smaller error than h' over the training examples, but h' has a smaller overall error than h over the entire distribution (or data set) of instances.

Overfitting definition (T. Mitchell)

- We say that a hypothesis overfits the training examples if some other hypothesis that fits the training examples less well actually performs better over the entire distribution of instances (i.e., including instances beyond the training set).

Definition 2.3.1 (overfitting)

Given a hypothesis space $\mathcal{H}$, a hypothesis $h \in \mathcal{H}$ is said to overfit the training data if there exists some alternative hypothesis $h' \in \mathcal{H}$, such that h has smaller error than h' over the training examples, but h' has a smaller overall error than h over the entire distribution (or data set) of instances.

- We'll visit this topic again when we discuss bias/variance, but first lets discuss a few more models.

Linear Regression

- Linear regression involved fitting a model of the form $y = h_\theta(x) = \sum_i \theta_i x_i$ where x_i is the i^{th} input feature and θ_i is the i^{th} parameter.

Linear Regression

- Linear regression involved fitting a model of the form $y = h_\theta(x) = \sum_i \theta_i x_i$ where x_i is the i^{th} input feature and θ_i is the i^{th} parameter.
- model is linear in the parameters, $h_{\alpha\theta+\alpha'\theta'}(x) = \alpha h_\theta(x) + \alpha' h_{\theta'}(x)$ that we "regress" to.

Linear Regression

- Linear regression involved fitting a model of the form $y = h_\theta(x) = \sum_i \theta_i x_i$ where x_i is the i^{th} input feature and θ_i is the i^{th} parameter.
- model is linear in the parameters, $h_{\alpha\theta+\alpha'\theta'}(x) = \alpha h_\theta(x) + \alpha' h_{\theta'}(x)$ that we "regress" to.
- Reasonable starting model for when $y \in \mathbb{R}$.

Logistic Regression

- What if y is category? E.g., $y \in \{0, 1, 2, \ldots, \ell - 1\}$ for ℓ categories.

Logistic Regression

- What if y is category? E.g., $y \in \{0, 1, 2, \ldots, \ell - 1\}$ for ℓ categories.
- Simplest case is $y \in \{0, 1\}$, a binary label (more general, y could be an integer, or sometimes $\in \{-1, +1\}$, or a vector of integers).

Logistic Regression

- What if y is category? E.g., $y \in \{0, 1, 2, \ldots, \ell - 1\}$ for ℓ categories.
- Simplest case is $y \in \{0, 1\}$, a binary label (more general, y could be an integer, or sometimes $\in \{-1, +1\}$, or a vector of integers).
- Categorical prediction, or binary classification, predict which class (zero or one) data object x is in (e.g., buy/sell, good/defective, fresh/stale, fraud/genuine, spam/not-spam, fake news/fake-fake news, etc.).

Logistic Regression

- What if y is category? E.g., $y \in \{0, 1, 2, \ldots, \ell - 1\}$ for ℓ categories.
- Simplest case is $y \in \{0, 1\}$, a binary label (more general, y could be an integer, or sometimes $\in \{-1, +1\}$, or a vector of integers).
- Categorical prediction, or binary classification, predict which class (zero or one) data object x is in (e.g., buy/sell, good/defective, fresh/stale, fraud/genuine, spam/not-spam, fake news/fake-fake news, etc.).
- Linear model can still be used but might not be ideal, we do not need values outside the range $[0, 1]$.

Logistic Regression

- What if y is category? E.g., $y \in \{0, 1, 2, \ldots, \ell - 1\}$ for ℓ categories.
- Simplest case is $y \in \{0, 1\}$, a binary label (more general, y could be an integer, or sometimes $\in \{-1, +1\}$, or a vector of integers).
- Categorical prediction, or binary classification, predict which class (zero or one) data object x is in (e.g., buy/sell, good/defective, fresh/stale, fraud/genuine, spam/not-spam, fake news/fake-fake news, etc.).
- Linear model can still be used but might not be ideal, we do not need values outside the range $[0, 1]$.
- With probabilities, binary prediction can still be presented with a representation of uncertainty — rather than crisp 0/1 decision, can look at $p(y = 0|x)$ vs. $p(y = 1|x)$.

Logistic Regression

- What if y is category? E.g., $y \in \{0, 1, 2, \ldots, \ell - 1\}$ for ℓ categories.
- Simplest case is $y \in \{0, 1\}$, a binary label (more general, y could be an integer, or sometimes $\in \{-1, +1\}$, or a vector of integers).
- Categorical prediction, or binary classification, predict which class (zero or one) data object x is in (e.g., buy/sell, good/defective, fresh/stale, fraud/genuine, spam/not-spam, fake news/fake-fake news, etc.).
- Linear model can still be used but might not be ideal, we do not need values outside the range $[0, 1]$.
- With probabilities, binary prediction can still be presented with a representation of uncertainty — rather than crisp 0/1 decision, can look at $p(y = 0|x)$ vs. $p(y = 1|x)$.
- Most uncertain about x if $p(y = 0|x) = p(y = 1|x) = 0.5$. Most certain bout x if $p(y = 1|x) = 0$ or $p(y = 1|x) = 1$.

Logistic Regression

- What if y is category? E.g., $y \in \{0, 1, 2, \ldots, \ell - 1\}$ for ℓ categories.
- Simplest case is $y \in \{0, 1\}$, a binary label (more general, y could be an integer, or sometimes $\in \{-1, +1\}$, or a vector of integers).
- Categorical prediction, or binary classification, predict which class (zero or one) data object x is in (e.g., buy/sell, good/defective, fresh/stale, fraud/genuine, spam/not-spam, fake news/fake-fake news, etc.).
- Linear model can still be used but might not be ideal, we do not need values outside the range $[0, 1]$.
- With probabilities, binary prediction can still be presented with a representation of uncertainty — rather than crisp 0/1 decision, can look at $p(y = 0|x)$ vs. $p(y = 1|x)$.
- Most uncertain about x if $p(y = 0|x) = p(y = 1|x) = 0.5$. Most certain bout x if $p(y = 1|x) = 0$ or $p(y = 1|x) = 1$.
- Correctness is not the same as certainty!!!

Logistic Regression

- What if y is category? E.g., $y \in \{0, 1, 2, \ldots, \ell - 1\}$ for ℓ categories.
- Simplest case is $y \in \{0, 1\}$, a binary label (more general, y could be an integer, or sometimes $\in \{-1, +1\}$, or a vector of integers).
- Categorical prediction, or binary classification, predict which class (zero or one) data object x is in (e.g., buy/sell, good/defective, fresh/stale, fraud/genuine, spam/not-spam, fake news/fake-fake news, etc.).
- Linear model can still be used but might not be ideal, we do not need values outside the range $[0, 1]$.
- With probabilities, binary prediction can still be presented with a representation of uncertainty — rather than crisp 0/1 decision, can look at $p(y = 0|x)$ vs. $p(y = 1|x)$.
- Most uncertain about x if $p(y = 0|x) = p(y = 1|x) = 0.5$. Most certain bout x if $p(y = 1|x) = 0$ or $p(y = 1|x) = 1$.
- Correctness is not the same as certainty!!!
- It is harder for a linear models to give such an interpretation since linear model output has no bound, in general.

Making a decision

- Given a probability model $p(y|x)$, how do we make a final decision?

Making a decision

- Given a probability model $p(y|x)$, how do we make a final decision?
- Let $y \in \{0, 1\}$ be the true label and $\tilde{y} \in \{0, 1\}$ be a prediction.

Making a decision

- Given a probability model $p(y|x)$, how do we make a final decision?
- Let $y \in \{0, 1\}$ be the true label and $\tilde{y} \in \{0, 1\}$ be a prediction.
- Decide true if $p(y = 1|x) \geq \tau$ where $\tau \in [0, 1]$ is a decision threshold, i.e., $\tilde{y} = \mathbf{1}_{\{p(y=1|x) \geq \tau\}}$. Natural value of $\tau = 0.5$ but other values are also not unreasonable.

Making a decision

- Given a probability model $p(y|x)$, how do we make a final decision?
- Let $y \in \{0, 1\}$ be the true label and $\tilde{y} \in \{0, 1\}$ be a prediction.
- Decide true if $p(y = 1|x) \geq \tau$ where $\tau \in [0, 1]$ is a decision threshold, i.e., $\tilde{y} = \mathbf{1}_{\{p(y=1|x) \geq \tau\}}$. Natural value of $\tau = 0.5$ but other values are also not unreasonable.
- Given a validation data set $\mathcal{D}_{\text{va}} = \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{n}$ where on which a classifier produces predictions $\left\{\tilde{y}^{(i)}\right\}_{i=1}^{n}$, we can compute the following quantities:

Making a decision

- Given a probability model $p(y|x)$, how do we make a final decision?
- Let $y \in \{0, 1\}$ be the true label and $\tilde{y} \in \{0, 1\}$ be a prediction.
- Decide true if $p(y = 1|x) \geq \tau$ where $\tau \in [0, 1]$ is a decision threshold, i.e., $\tilde{y} = \mathbf{1}_{\{p(y=1|x) \geq \tau\}}$. Natural value of $\tau = 0.5$ but other values are also not unreasonable.
- Given a validation data set $\mathcal{D}_{\text{va}} = \left\{(x^{(i)}, y^{(i)})\right\}_{i=1}^{n}$ where on which a classifier produces predictions $\left\{\tilde{y}^{(i)}\right\}_{i=1}^{n}$, we can compute the following quantities:
  - True positives $\text{TP} = \sum_{i=1}^{n} \mathbf{1}_{\left\{\tilde{y}^{(i)}=1 \wedge y^{(i)}=1\right\}}$

Making a decision

- Given a probability model $p(y|x)$, how do we make a final decision?
- Let $y \in \{0, 1\}$ be the true label and $\tilde{y} \in \{0, 1\}$ be a prediction.
- Decide true if $p(y = 1|x) \geq \tau$ where $\tau \in [0, 1]$ is a decision threshold, i.e., $\tilde{y} = \mathbf{1}_{\{p(y=1|x) \geq \tau\}}$. Natural value of $\tau = 0.5$ but other values are also not unreasonable.
- Given a validation data set $\mathcal{D}_{\text{va}} = \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{n}$ where on which a classifier produces predictions $\left\{\tilde{y}^{(i)}\right\}_{i=1}^{n}$, we can compute the following quantities:
  - True positives $\text{TP} = \sum_{i=1}^{n} \mathbf{1}_{\left\{\tilde{y}^{(i)}=1 \wedge y^{(i)}=1\right\}}$
  - True negatives $\text{TN} = \sum_{i=1}^{n} \mathbf{1}_{\left\{\tilde{y}^{(i)}=0 \wedge y^{(i)}=0\right\}}$

Making a decision

- Given a probability model $p(y|x)$, how do we make a final decision?
- Let $y \in \{0, 1\}$ be the true label and $\tilde{y} \in \{0, 1\}$ be a prediction.
- Decide true if $p(y = 1|x) \geq \tau$ where $\tau \in [0, 1]$ is a decision threshold, i.e., $\tilde{y} = \mathbf{1}_{\{p(y=1|x) \geq \tau\}}$. Natural value of $\tau = 0.5$ but other values are also not unreasonable.
- Given a validation data set $\mathcal{D}_{\text{va}} = \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{n}$ where on which a classifier produces predictions $\left\{\tilde{y}^{(i)}\right\}_{i=1}^{n}$, we can compute the following quantities:
  - True positives $\text{TP} = \sum_{i=1}^{n} \mathbf{1}_{\left\{\tilde{y}^{(i)}=1 \wedge y^{(i)}=1\right\}}$
  - True negatives $\text{TN} = \sum_{i=1}^{n} \mathbf{1}_{\left\{\tilde{y}^{(i)}=0 \wedge y^{(i)}=0\right\}}$
  - False positives $\text{FP} = \sum_{i=1}^{n} \mathbf{1}_{\left\{\tilde{y}^{(i)}=1 \wedge y^{(i)}=0\right\}}$

Making a decision

- Given a probability model $p(y|x)$, how do we make a final decision?
- Let $y \in \{0, 1\}$ be the true label and $\tilde{y} \in \{0, 1\}$ be a prediction.
- Decide true if $p(y = 1|x) \geq \tau$ where $\tau \in [0, 1]$ is a decision threshold, i.e., $\tilde{y} = \mathbf{1}_{\{p(y=1|x) \geq \tau\}}$. Natural value of $\tau = 0.5$ but other values are also not unreasonable.
- Given a validation data set $\mathcal{D}_{\text{va}} = \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{n}$ where on which a classifier produces predictions $\left\{\tilde{y}^{(i)}\right\}_{i=1}^{n}$, we can compute the following quantities:
  - True positives $\text{TP} = \sum_{i=1}^{n} \mathbf{1}_{\left\{\tilde{y}^{(i)}=1 \wedge y^{(i)}=1\right\}}$
  - True negatives $\text{TN} = \sum_{i=1}^{n} \mathbf{1}_{\left\{\tilde{y}^{(i)}=0 \wedge y^{(i)}=0\right\}}$
  - False positives $\text{FP} = \sum_{i=1}^{n} \mathbf{1}_{\left\{\tilde{y}^{(i)}=1 \wedge y^{(i)}=0\right\}}$
  - False negatives $\text{FN} = \sum_{i=1}^{n} \mathbf{1}_{\left\{\tilde{y}^{(i)}=0 \wedge y^{(i)}=1\right\}}$

Making a decision

- Given a probability model $p(y|x)$, how do we make a final decision?
- Let $y \in \{0, 1\}$ be the true label and $\tilde{y} \in \{0, 1\}$ be a prediction.
- Decide true if $p(y = 1|x) \geq \tau$ where $\tau \in [0, 1]$ is a decision threshold, i.e., $\tilde{y} = \mathbf{1}_{\{p(y=1|x) \geq \tau\}}$. Natural value of $\tau = 0.5$ but other values are also not unreasonable.
- Given a validation data set $\mathcal{D}_{\text{va}} = \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{n}$ where on which a classifier produces predictions $\left\{\tilde{y}^{(i)}\right\}_{i=1}^{n}$, we can compute the following quantities:
  - True positives $\text{TP} = \sum_{i=1}^{n} \mathbf{1}_{\left\{\tilde{y}^{(i)}=1 \wedge y^{(i)}=1\right\}}$
  - True negatives $\text{TN} = \sum_{i=1}^{n} \mathbf{1}_{\left\{\tilde{y}^{(i)}=0 \wedge y^{(i)}=0\right\}}$
  - False positives $\text{FP} = \sum_{i=1}^{n} \mathbf{1}_{\left\{\tilde{y}^{(i)}=1 \wedge y^{(i)}=0\right\}}$
  - False negatives $\text{FN} = \sum_{i=1}^{n} \mathbf{1}_{\left\{\tilde{y}^{(i)}=0 \wedge y^{(i)}=1\right\}}$
- Note that number of samples $n = \text{TP} + \text{TN} + \text{FP} + \text{FN}$

Accuracy and Error

- Overall accuracy

$$\text{Accuracy} = \frac{\text{Total correct predictions}}{\text{Total predictions}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.1}$$

Accuracy and Error

- Overall accuracy

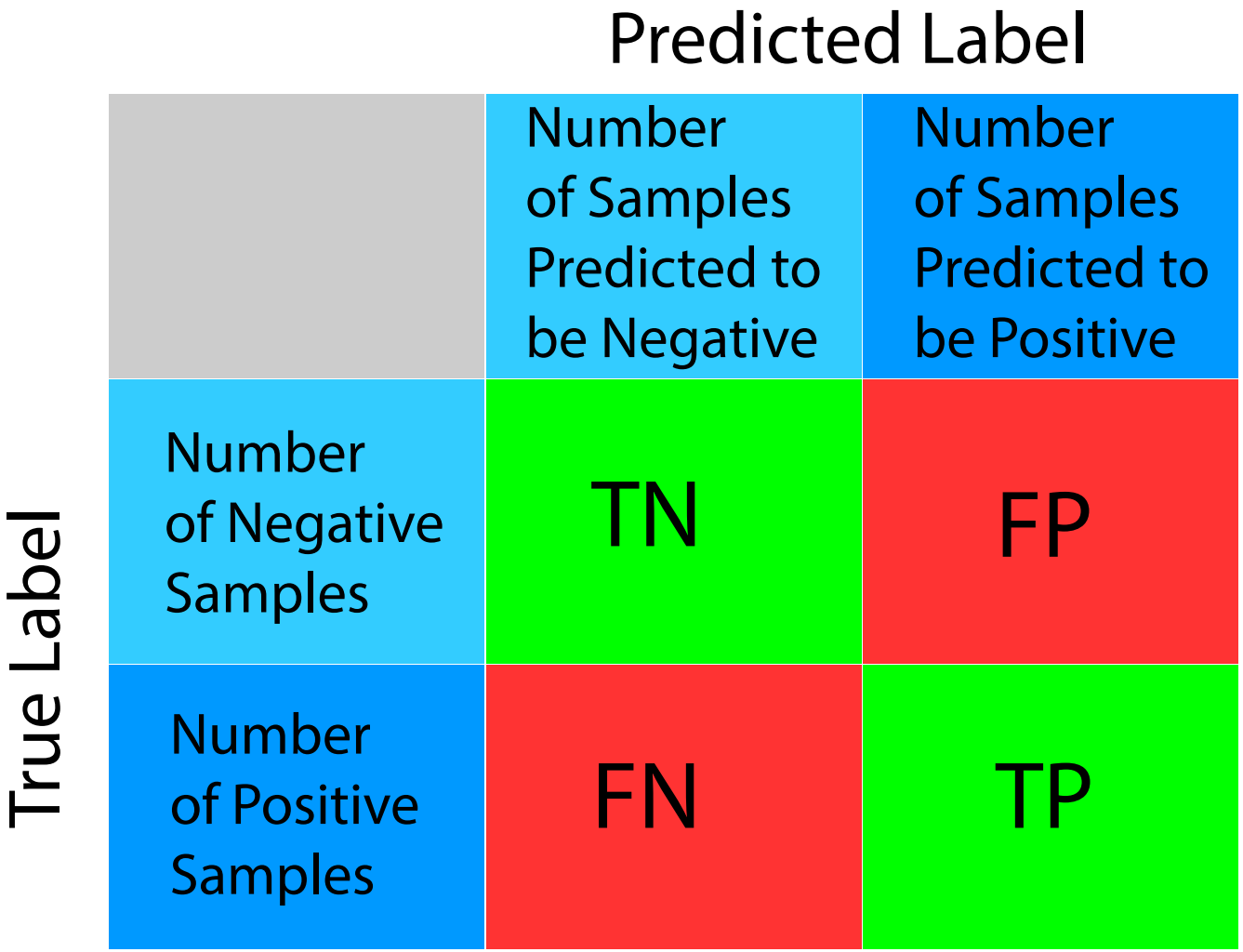
$$\text{Accuracy} = \frac{\text{Total correct predictions}}{\text{Total predictions}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.1}$$

- Error

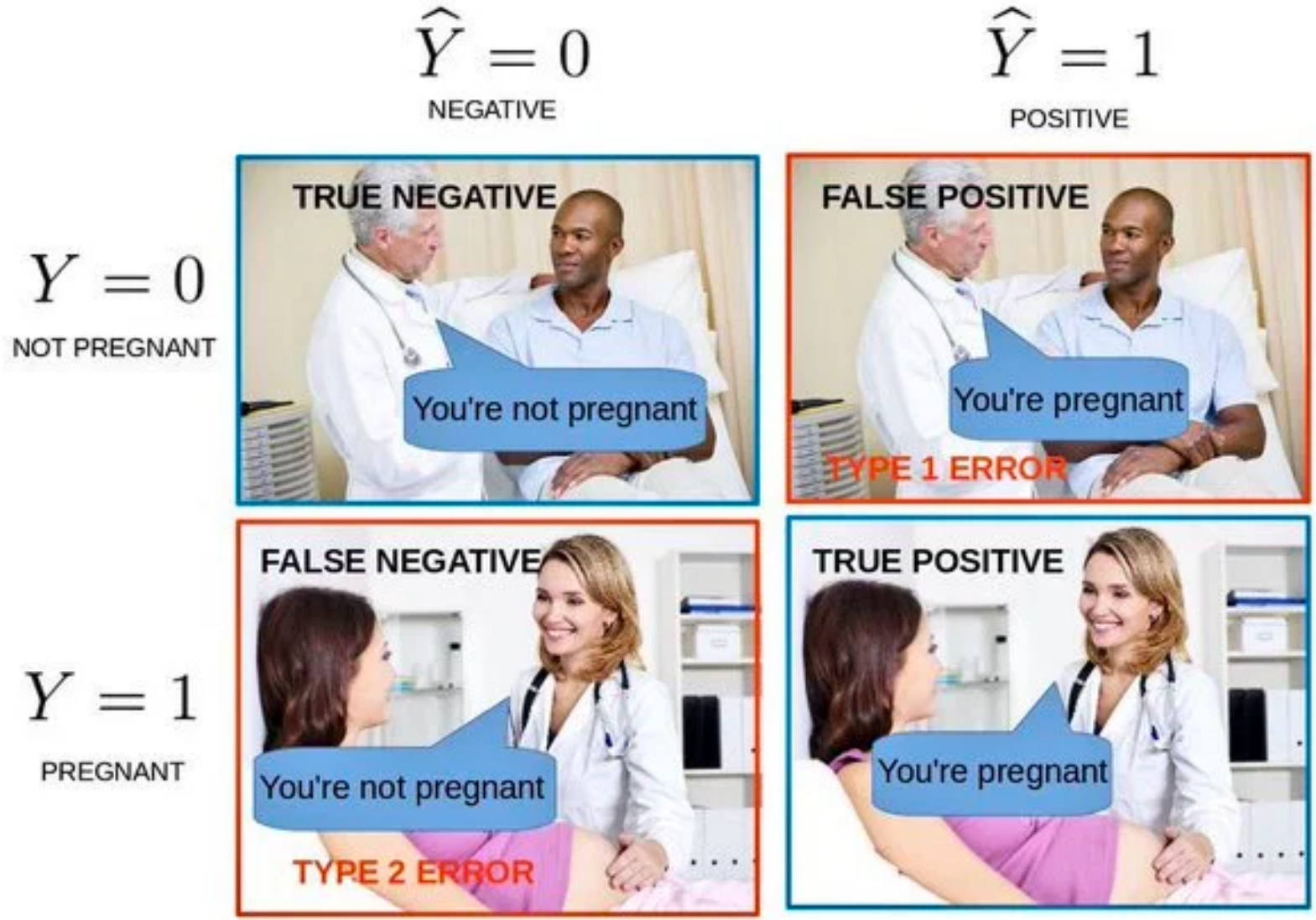
$$\text{Error} = 1.0 - \text{Accuracy} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.2}$$

Binary Confusion Matrix

- Given n samples in a validation data set, we can plot the relationship between TP, TN, FP, FN.



False Positive and False Negative



https://www.kdnuggets.com/2020/01/guide-precision-recall-confusion-matrix.html

Precision, Recall, and F-Feasure

- Precision:

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} = \frac{\mathrm{TP}}{\text{number of predicted positives}} \tag{2.3}$$

Precision, Recall, and F-Feasure

- Precision:

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} = \frac{\mathrm{TP}}{\text{number of predicted positives}} \tag{2.3}$$

- Recall

$$\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} = \frac{\mathrm{TP}}{\text{number of positives}} \tag{2.4}$$

Precision, Recall, and F-Feasure

- Precision:

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} = \frac{\mathrm{TP}}{\text{number of predicted positives}} \tag{2.3}$$

- Recall

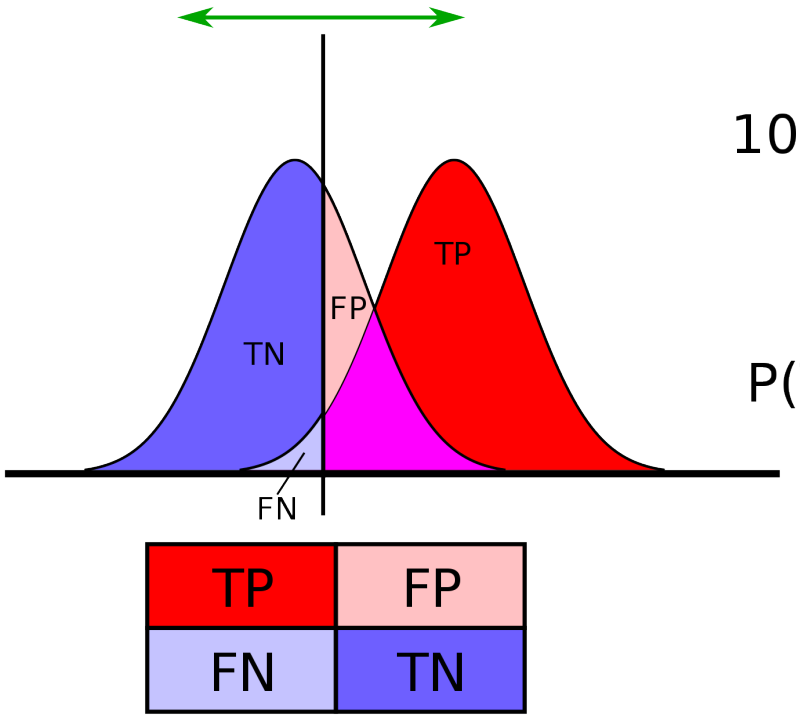
$$\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} = \frac{\mathrm{TP}}{\text{number of positives}} \tag{2.4}$$

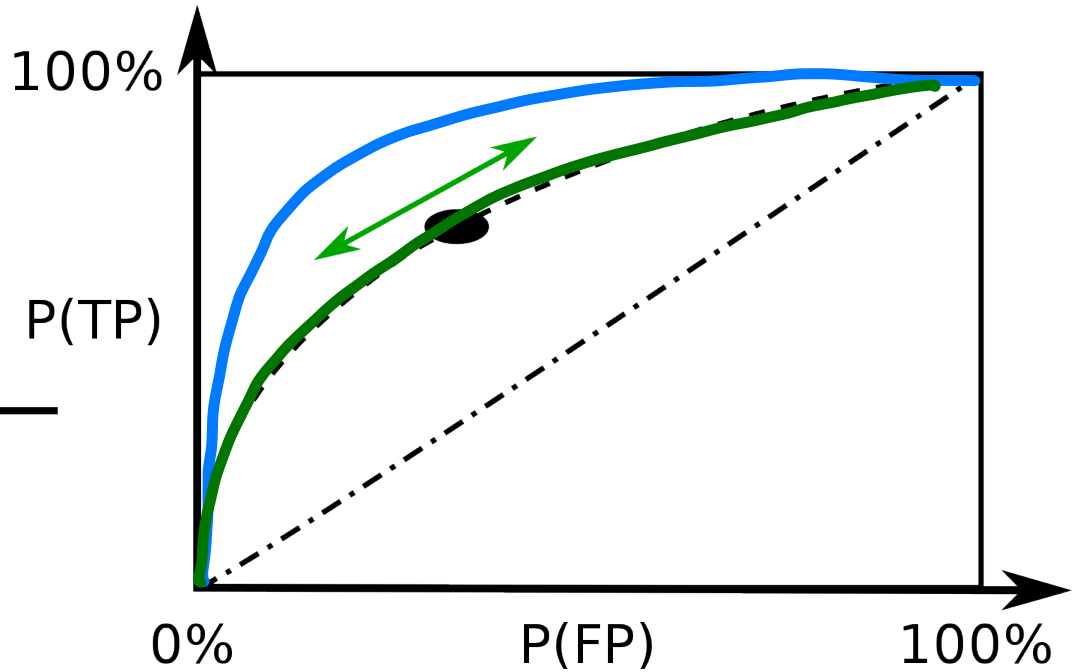
- F-measure (or F1-score), harmonic mean of precision and recall,

$$\text{F-measure} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} \neq 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.5}$$

ROC and AUC

- Recall, we predict as $\tilde{y} = \mathbf{1}_{\{p(y=1|x) \geq \tau\}}$, but how does TP and FP change as we vary the decision threshold τ?

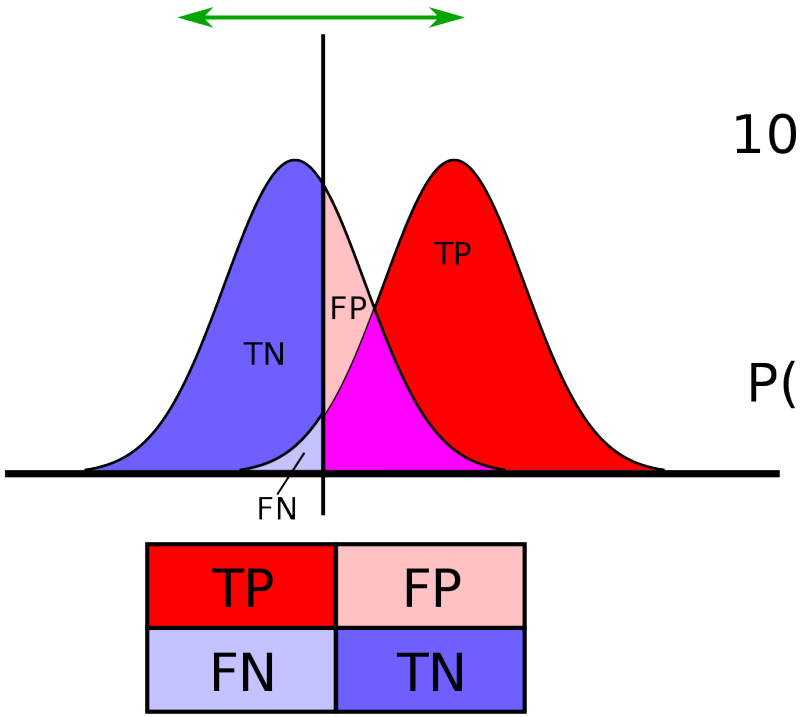


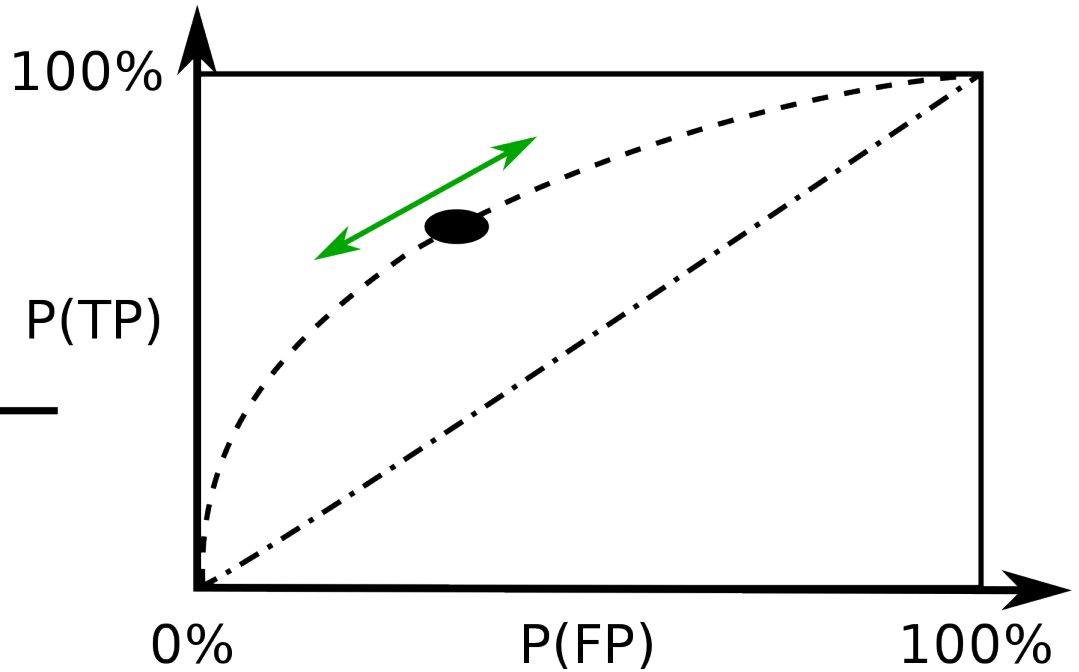


https://en.wikipedia.org/wiki/Receiver_operating_characteristic

ROC and AUC

- Recall, we predict as $\tilde{y} = \mathbf{1}_{\{p(y=1|x) \geq \tau\}}$, but how does TP and FP change as we vary the decision threshold τ?
- Receiver Operating Characteristic (ROC) curve is determined by τ.

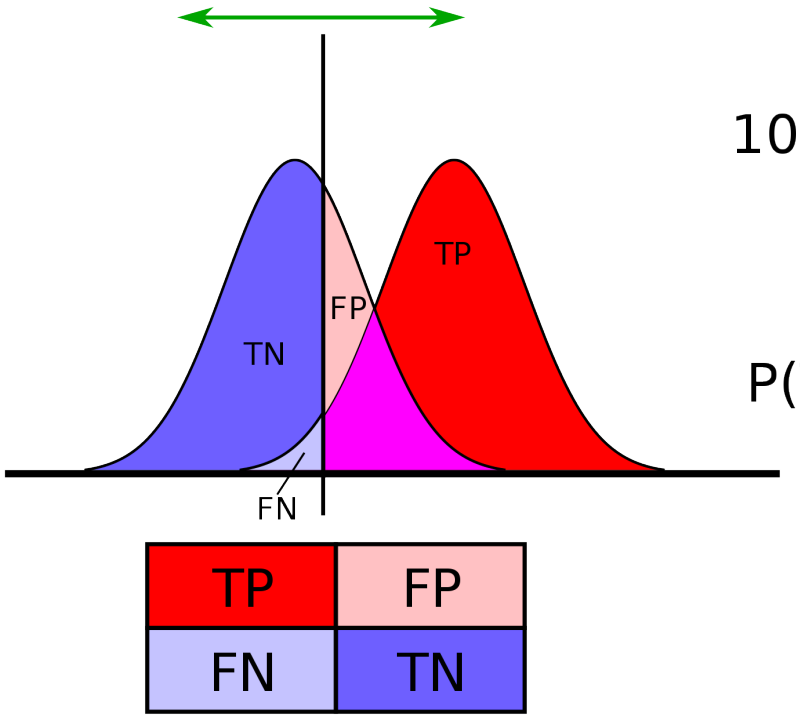


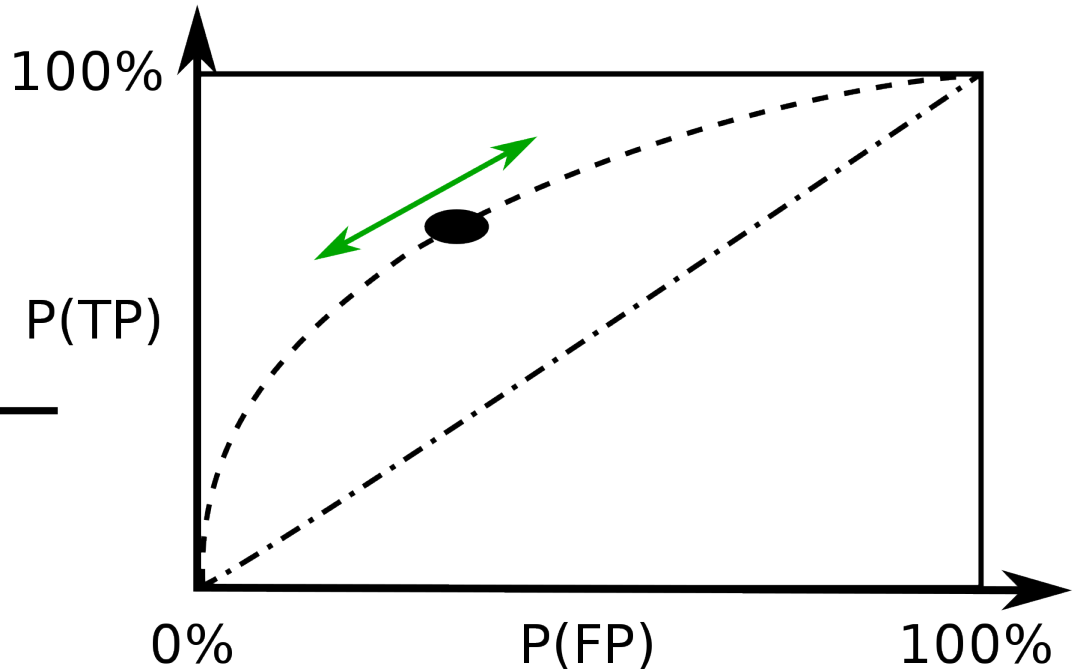


https://en.wikipedia.org/wiki/Receiver_operating_characteristic

ROC and AUC

- Recall, we predict as $\tilde{y} = \mathbf{1}_{\{p(y=1|x) \geq \tau\}}$, but how does TP and FP change as we vary the decision threshold τ?
- Receiver Operating Characteristic (ROC) curve is determined by τ.





- Area under the curve (AUC) gives an overall measure of how well model is doing for all τ. Higher AUC is better.

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

Logistic Regression

- Solution for binary classification: use a logistic function as in

$$\Pr(y = 1|x) = h_\theta(x) = g(\theta^\intercal x) = \frac{1}{1 + \exp(-\theta^\intercal x)} \tag{2.6}$$

where $\theta^\intercal x = \theta_0 + \sum_{i=1}^m \theta_i x_i$, $x_0 \equiv 1$ so θ_0 is the bias/shift.

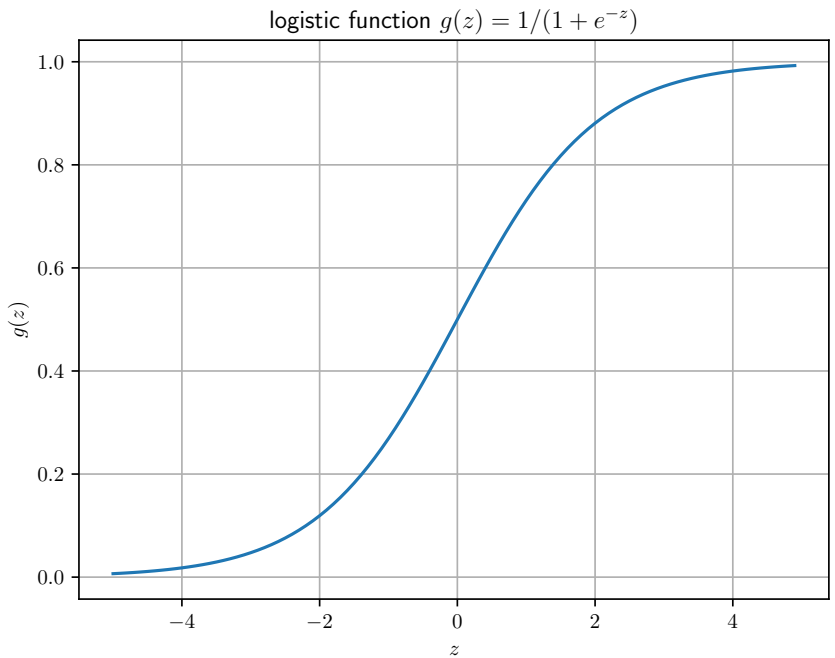
Logistic Regression

- Solution for binary classification: use a logistic function as in

$$\Pr(y = 1|x) = h_\theta(x) = g(\theta^\intercal x) = \frac{1}{1 + \exp(-\theta^\intercal x)} \tag{2.6}$$

where $\theta^\intercal x = \theta_0 + \sum_{i=1}^{m} \theta_i x_i$, $x_0 \equiv 1$ so θ_0 is the bias/shift.

- $g(z) = 1/(1 + e^{-x})$ is known as a logistic function.



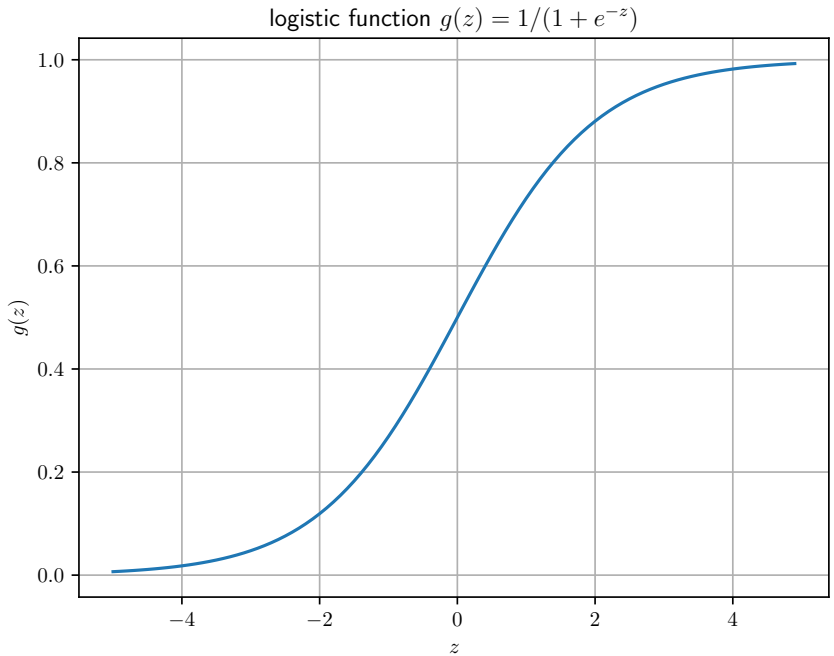
Logistic Regression

- Solution for binary classification: use a logistic function as in

$$\Pr(y = 1|x) = h_\theta(x) = g(\theta^\intercal x) = \frac{1}{1 + \exp(-\theta^\intercal x)} \tag{2.6}$$

where $\theta^\intercal x = \theta_0 + \sum_{i=1}^m \theta_i x_i$, $x_0 \equiv 1$ so θ_0 is the bias/shift.

- $g(z) = 1/(1 + e^{-x})$ is known as a logistic function.



- A logistic function is one type of sigmoid function, others being hyperbolic tangent, arctan, error function, etc. (see https://en.wikipedia.org/wiki/Sigmoid_function).

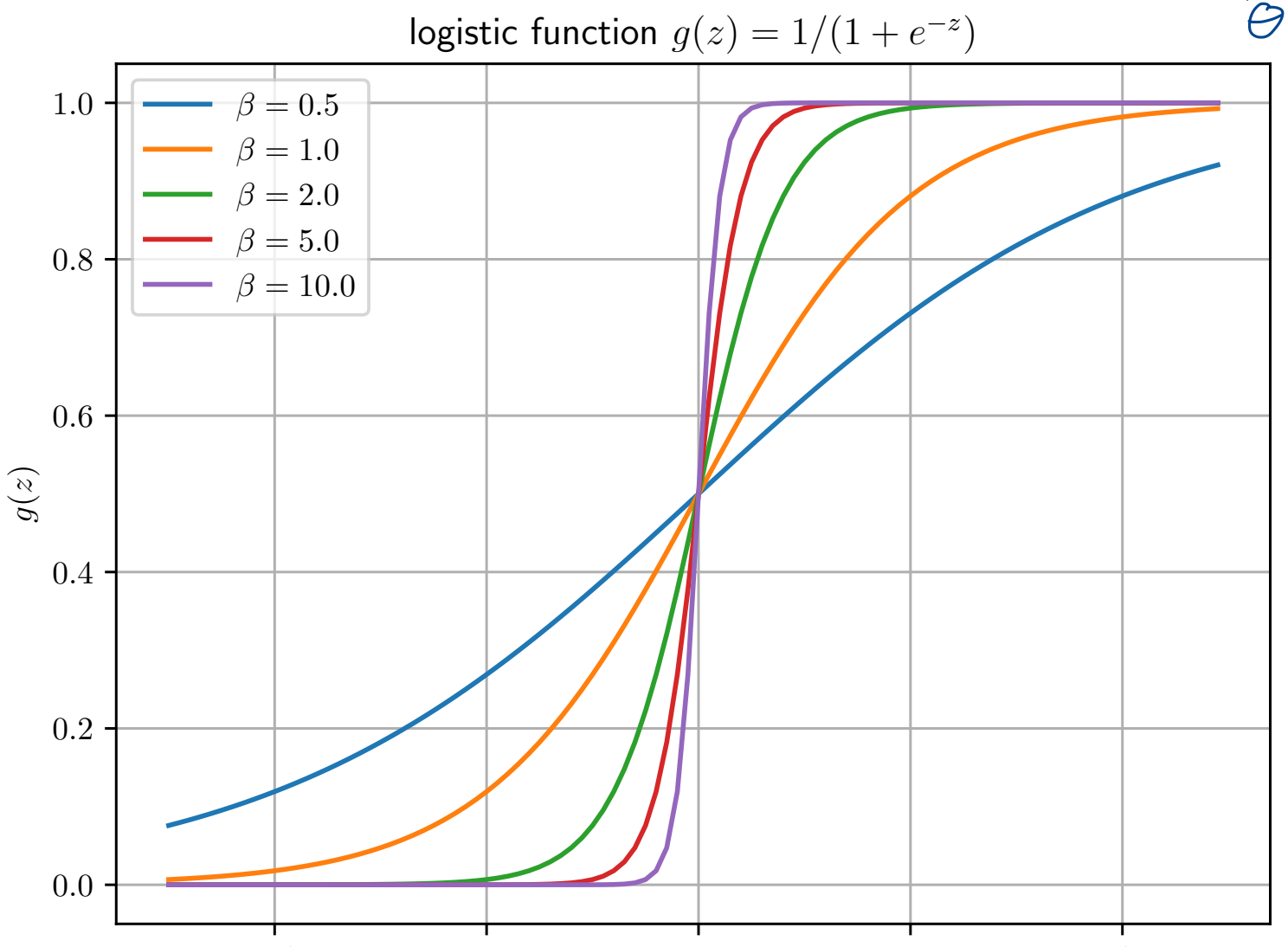
Logistic with scale parameter β

- Approximate a step function with scale parameter $\beta \in \mathbb{R}_+$, giving $g_\beta(z) = 1/(1+e^{-\beta x})$

$1/(1+\exp(-\langle\theta, x\rangle))$

$\theta \leftarrow \theta \cdot 100$

$x \in \mathbb{R}^m$



Gradients of Logistic

- A logistic function's gradient is easy to compute. This follows since:

$$g'(z) = \frac{d}{dz}\frac{1}{1+e^{-z}} \tag{2.7}$$

$$= -\frac{1}{(1+e^{-z})^2}(-e^{-z}) \tag{2.8}$$

$$= \frac{1}{1+e^{-z}} \cdot \frac{e^{-z}}{1+e^{-z}} \tag{2.9}$$

$$= \frac{1}{1+e^{-z}}\left(1 - \frac{1}{1+e^{-z}}\right) \tag{2.10}$$

$$= g(z)(1-g(z)) \tag{2.11}$$

Gradients of Logistic

- A logistic function's gradient is easy to compute. This follows since:

$$g'(z) = \frac{d}{dz}\frac{1}{1+e^{-z}} \tag{2.7}$$

$$= -\frac{1}{(1+e^{-z})^2}(-e^{-z}) \tag{2.8}$$

$$= \frac{1}{1+e^{-z}} \cdot \frac{e^{-z}}{1+e^{-z}} \tag{2.9}$$

$$= \frac{1}{1+e^{-z}}\left(1 - \frac{1}{1+e^{-z}}\right) \tag{2.10}$$

$$= g(z)(1-g(z)) \tag{2.11}$$

- Given this, we can derive a gradient descent learning rule, similar to LMS, but for logistic regression.

Fitting Logistic Regression Using Gradient Descent

- Training data $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i \in [n]}$, now $y^{(i)} \in \{0, 1\}$ is a binary label.

Fitting Logistic Regression Using Gradient Descent

- Training data $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i \in [n]}$, now $y^{(i)} \in \{0, 1\}$ is a binary label.
- Goal: formulate likelihood (to maximize) in terms of parameters θ.

Fitting Logistic Regression Using Gradient Descent

- Training data $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i \in [n]}$, now $y^{(i)} \in \{0, 1\}$ is a binary label.
- Goal: formulate likelihood (to maximize) in terms of parameters θ.
- Probability model. $\Pr(y = 1|x; \theta) = h_\theta(x) = g(\theta^\intercal x)$ and $\Pr(y = 0|x; \theta) = 1 - h_\theta(x)$, thus for $y \in \{0, 1\}$,

$$\Pr(y|x; \theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y} \tag{2.12}$$

Fitting Logistic Regression Using Gradient Descent

- Training data $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i \in [n]}$, now $y^{(i)} \in \{0, 1\}$ is a binary label.
- Goal: formulate likelihood (to maximize) in terms of parameters θ.
- Probability model. $\Pr(y = 1|x; \theta) = h_\theta(x) = g(\theta^\intercal x)$ and $\Pr(y = 0|x; \theta) = 1 - h_\theta(x)$, thus for $y \in \{0, 1\}$,

$$\Pr(y|x; \theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y} \tag{2.12}$$

- Likelihood $L(\theta) = \prod_i \Pr(y^{(i)}|x^{(i)}; \theta)$ and log likelihood

$$\ell(\theta) = \sum_{i=1}^{n} \left[y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right] \tag{2.13}$$

Fitting Logistic Regression Using Gradient Descent

- Training data $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i \in [n]}$, now $y^{(i)} \in \{0, 1\}$ is a binary label.
- Goal: formulate likelihood (to maximize) in terms of parameters θ.
- Probability model. $\Pr(y = 1|x; \theta) = h_\theta(x) = g(\theta^\intercal x)$ and $\Pr(y = 0|x; \theta) = 1 - h_\theta(x)$, thus for $y \in \{0, 1\}$,

$$\Pr(y|x; \theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y} \tag{2.12}$$

- Likelihood $L(\theta) = \prod_i \Pr(y^{(i)}|x^{(i)}; \theta)$ and log likelihood

$$\ell(\theta) = \sum_{i=1}^{n} \left[y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right] \tag{2.13}$$

- Negative Log likelihood (or cost$(\theta) = -\ell(\theta)$) is convex in θ.

Fitting Logistic Regression Using Gradient Descent

- Gradient of log likelihood on one training pair (x, y)

$$\frac{\partial}{\partial \theta}\ell(\theta) = (y - h_\theta(x))x. \tag{2.14}$$

Derivation of this derivative, via chain rule, uses the logistic derivative property $g'(z) = g(z)(1 - g(z))$.

Fitting Logistic Regression Using Gradient Descent

- Gradient of log likelihood on one training pair (x, y)

$$\frac{\partial}{\partial \theta}\ell(\theta) = (y - h_\theta(x))x. \tag{2.14}$$

Derivation of this derivative, via chain rule, uses the logistic derivative property $g'(z) = g(z)(1 - g(z))$.

- Gradient descent steps

$$\theta \leftarrow \theta + \alpha(y^{(i)} - h_\theta(x^{(i)}))x^{(i)} \tag{2.15}$$

again, direction is given by $x^{(i)}$, by an amount equal to $\alpha \cdot$ error $= \alpha(y^{(i)} - h_\theta(x^{(i)}))$

Fitting Logistic Regression Using Gradient Descent

- Gradient of log likelihood on one training pair (x, y)

$$\frac{\partial}{\partial \theta}\ell(\theta) = (y - h_\theta(x))x. \tag{2.14}$$

Derivation of this derivative, via chain rule, uses the logistic derivative property $g'(z) = g(z)(1 - g(z))$.

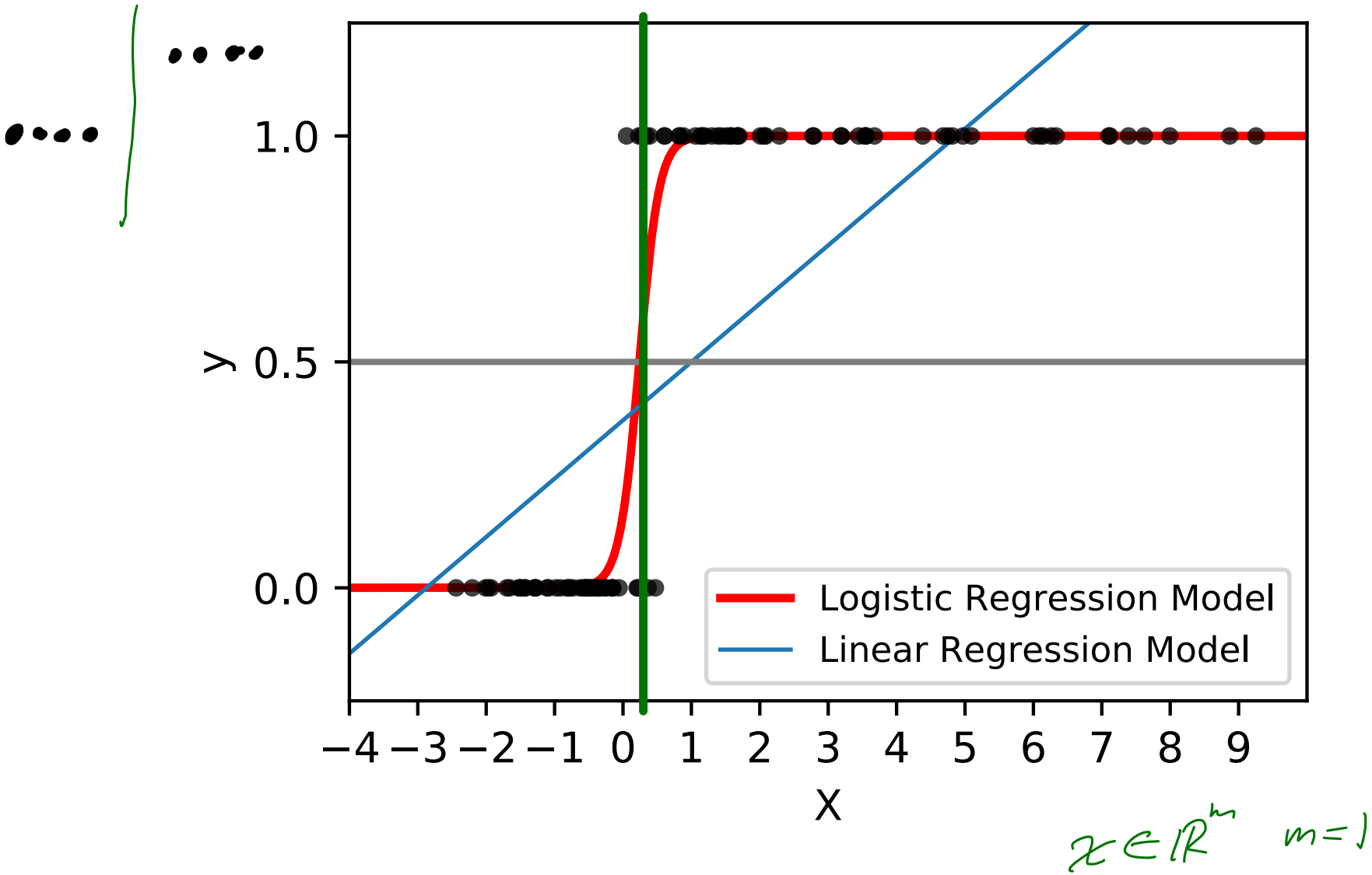
- Gradient descent steps

$$\theta \leftarrow \theta + \alpha(y^{(i)} - h_\theta(x^{(i)}))\frac{x^{(i)}}{\|x^{(i)}\|} \tag{2.15}$$

again, direction is given by $x^{(i)}$, by an amount equal to $\alpha \cdot$ error $= \alpha(y^{(i)} - h_\theta(x^{(i)}))$

- Error has same form as linear case (the answer, $y^{(i)}$ minus prediction $h_\theta(x^{(i)})$) but prediction is quite different from before.

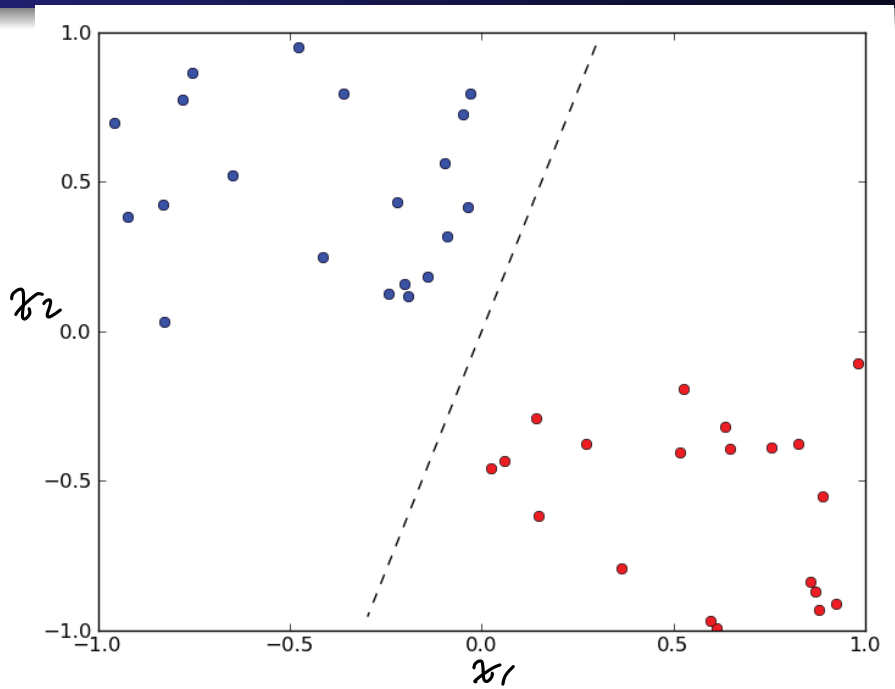
Fit Linear vs. Logistic Comparison



from https://scikit-learn.org/stable/auto_examples/linear_model/plot_logistic.html

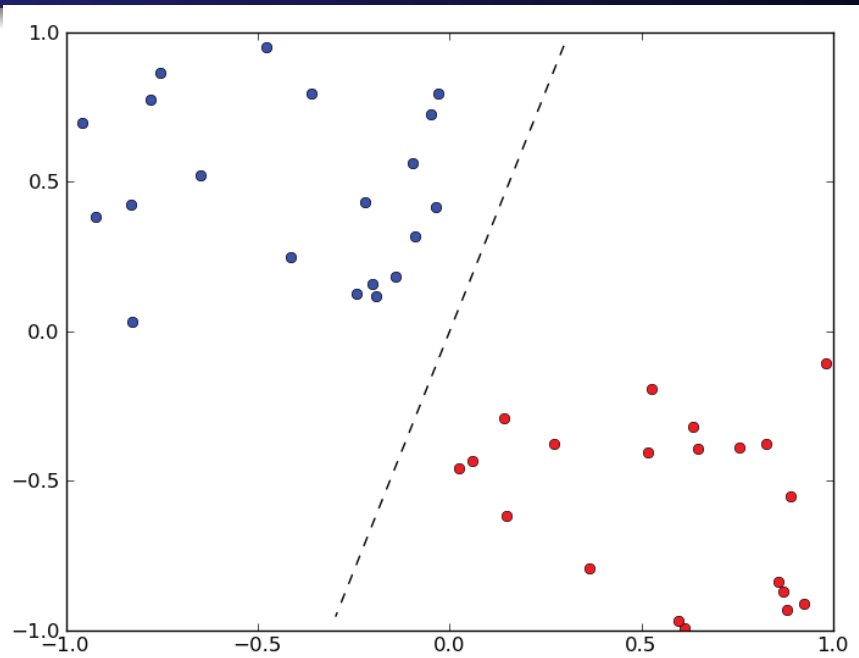
Logistic Regression, Linear Separability, and Overfitting

- logistic regression and linearly separable data. How smooth is the transition in logistic regression?



Logistic Regression, Linear Separability, and Overfitting

- logistic regression and linearly separable data. How smooth is the transition in logistic regression?



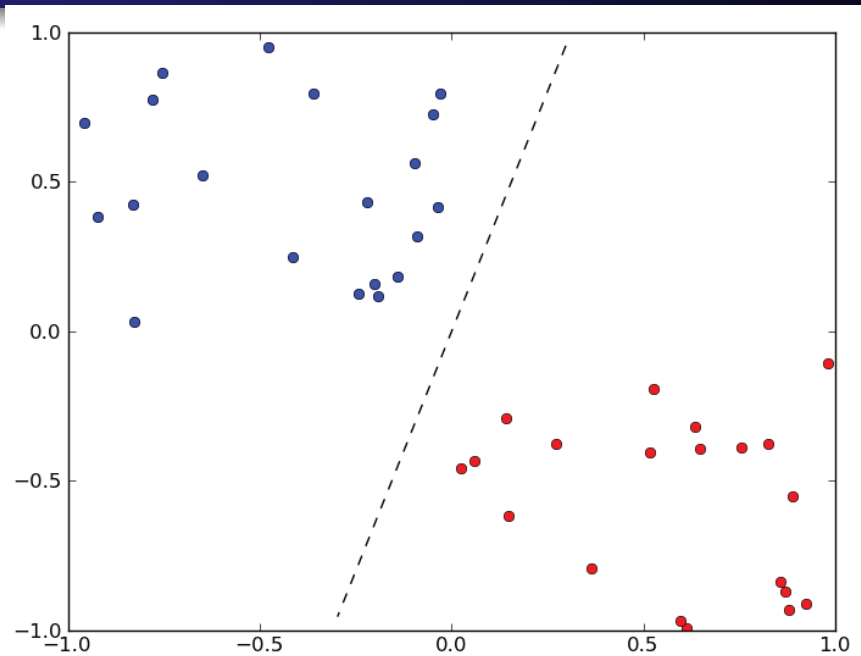
- Logistic regression negative log likelihood (i.e., cost) $J(\theta)$

$$J(\theta) = -\sum_{i=1}^{n}\left[y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))\right] \qquad (2.16)$$

to be minimized, where $h_\theta(x) = \frac{1}{1+\exp(-\theta^\intercal x)}$.

Logistic Regression, Linear Separability, and Overfitting

- logistic regression and linearly separable data. How smooth is the transition in logistic regression?



- Logistic regression negative log likelihood (i.e., cost) $J(\theta)$

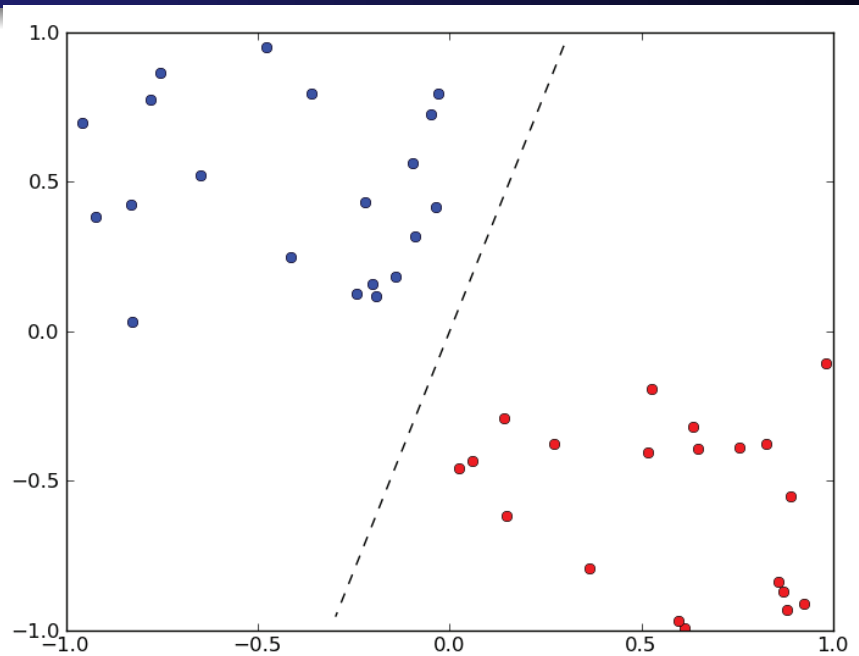
$$J(\theta) = -\sum_{i=1}^{m}\left[y^{(i)}\log h_\theta(x^{(i)}) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right] \qquad (2.16)$$

  to be minimized, where $h_\theta(x) = \frac{1}{1+\exp(-\theta^\intercal x)}$.

- What happens as cost decreases (likelihood improves)?

Logistic Regression, Linear Separability, and Overfitting

- logistic regression and linearly separable data. How smooth is the transition in logistic regression?



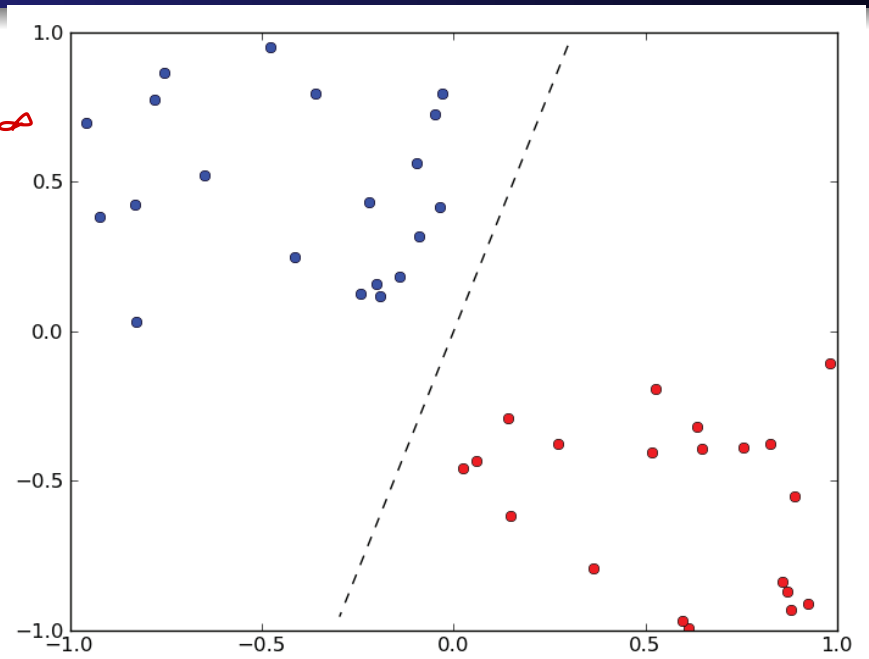
- Logistic regression negative log likelihood (i.e., cost) $J(\theta)$

$$J(\theta) = -\sum_{i=1}^{m}\left[y^{(i)}\log h_\theta(x^{(i)}) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right] \qquad (2.16)$$

  to be minimized, where $h_\theta(x) = \frac{1}{1+\exp(-\theta^\intercal x)}$.

- What happens as cost decreases (likelihood improves)? If $y^{(i)} = 1$, $h_\theta(x^{(i)}) \to 1$; if $y^{(i)} = 0$, $h_\theta(x^{(i)}) \to 0$. Hence, $J(\theta) \to 0$.

Logistic Regression, Linear Separability, and Overfitting



- logistic regression and linearly separable data. How smooth is the transition in logistic regression?

- Logistic regression negative log likelihood (i.e., cost) $J(\theta)$

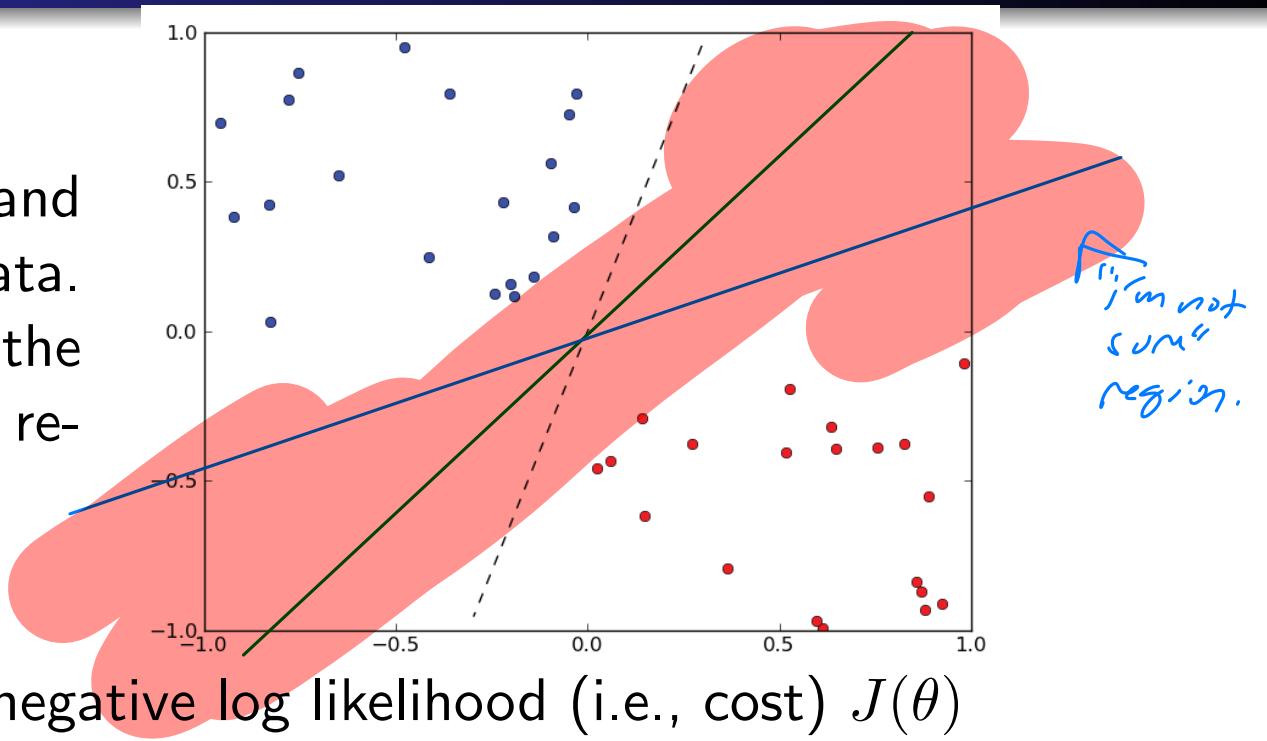
$$J(\theta) = -\sum_{i=1}^{m}\left[y^{(i)}\log h_\theta(x^{(i)}) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right] \qquad (2.16)$$

to be minimized, where $h_\theta(x) = \frac{1}{1+\exp(-\theta^\intercal x)}$.

- What happens as cost decreases (likelihood improves)? If $y^{(i)} = 1$, $h_\theta(x^{(i)}) \to 1$; if $y^{(i)} = 0$, $h_\theta(x^{(i)}) \to 0$. Hence, $J(\theta) \to 0$. Requires $\theta \to \infty$.

Logistic Regression, Linear Separability, and Overfitting

- logistic regression and linearly separable data. How smooth is the transition in logistic regression?



- Logistic regression negative log likelihood (i.e., cost) $J(\theta)$

$$J(\theta) = -\sum_{i=1}^{m}\left[y^{(i)}\log h_\theta(x^{(i)}) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right] \qquad (2.16)$$

to be minimized, where $h_\theta(x) = \frac{1}{1+\exp(-\theta^\intercal x)}$.

- What happens as cost decreases (likelihood improves)? If $y^{(i)} = 1$, $h_\theta(x^{(i)}) \to 1$; if $y^{(i)} = 0$, $h_\theta(x^{(i)}) \to 0$. Hence, $J(\theta) \to 0$. Requires $\theta \to \infty$. Should transition be allowed to be arbitrarily sudden?

Overfitting and large magnitude parameters

- Therefore, it seems that even large magnitude parameters can lead to a form of overfitting. Overfitting in that the i such that $y^{(i)} = 0$ get perfect zero prediction, and i such that $y^{(i)} = 1$ get perfect 1 prediction, where such certainty is probably not warranted.

Overfitting and large magnitude parameters

- Therefore, it seems that even large magnitude parameters can lead to a form of overfitting. Overfitting in that the i such that $y^{(i)} = 0$ get perfect zero prediction, and i such that $y^{(i)} = 1$ get perfect 1 prediction, where such certainty is probably not warranted.
- Better solution: don't overfit, for points close to decision boundary allow gradual prediction transition between 0 and 1 in region of uncertainty.

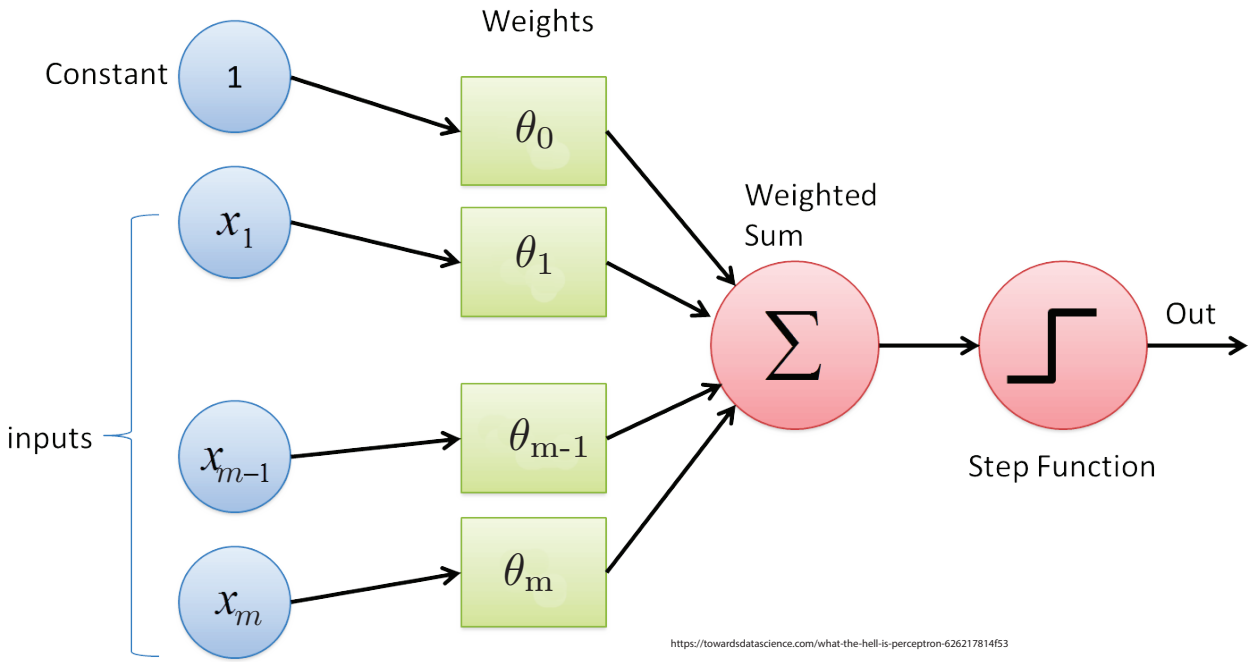
Overfitting and large magnitude parameters

- Therefore, it seems that even large magnitude parameters can lead to a form of overfitting. Overfitting in that the i such that $y^{(i)} = 0$ get perfect zero prediction, and i such that $y^{(i)} = 1$ get perfect 1 prediction, where such certainty is probably not warranted.
- Better solution: don't overfit, for points close to decision boundary allow gradual prediction transition between 0 and 1 in region of uncertainty.
- This requires means putting a restriction on θ (not letting it get too big).

Overfitting and large magnitude parameters

- Therefore, it seems that even large magnitude parameters can lead to a form of overfitting. Overfitting in that the i such that $y^{(i)} = 0$ get perfect zero prediction, and i such that $y^{(i)} = 1$ get perfect 1 prediction, where such certainty is probably not warranted.
- Better solution: don't overfit, for points close to decision boundary allow gradual prediction transition between 0 and 1 in region of uncertainty.
- This requires means putting a restriction on θ (not letting it get too big).
- One possible complexity penality, the 2-norm: $\Omega(\theta) = \|\theta\|_2$, prefers "simple" models which in thise case are those with small coefficients.

Perceptron and Logistic Regression



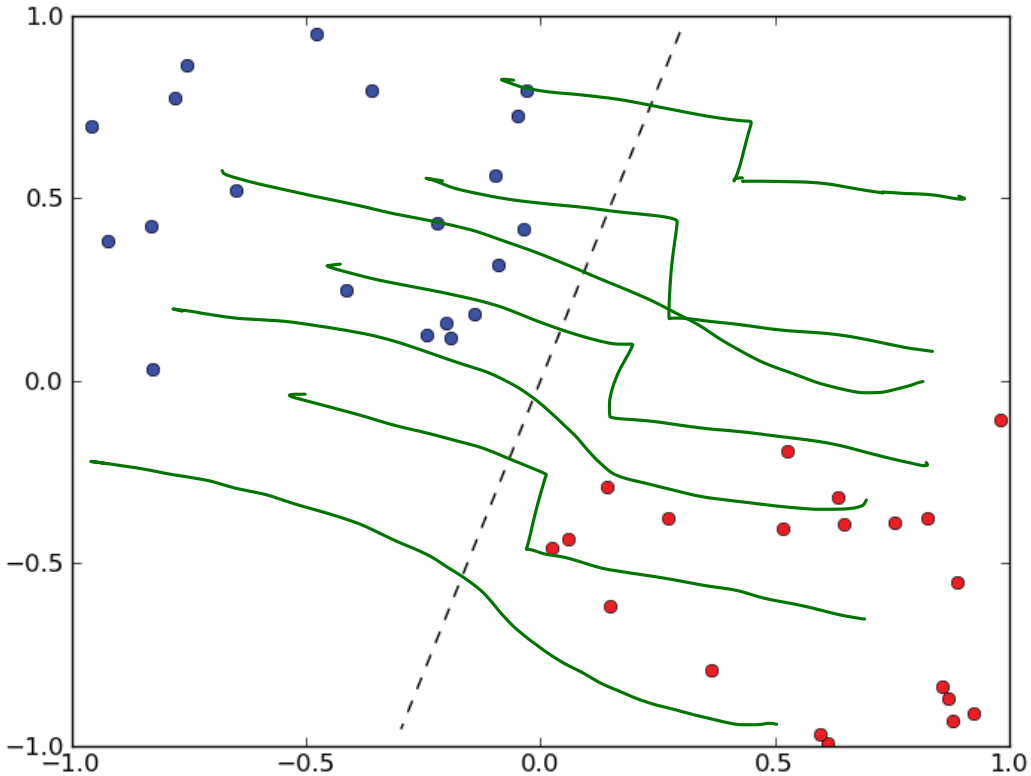
- Model $h_\theta(x) = g(\theta^\mathsf{T} x)$, perceptron uses a hard activation function

$$g(z) = \begin{cases} -1 & \text{if } z < 0 \\ +1 & \text{if } z \geq 0 \end{cases} = \left(\lim_{p \to \infty} g_p(z) \right) \cdot 2 - 1 \qquad (2.17)$$

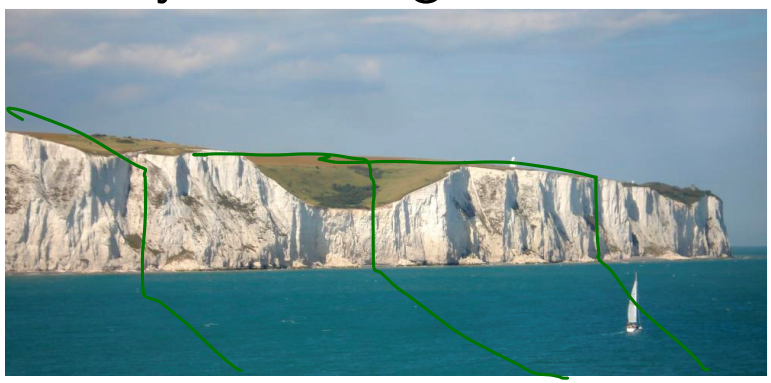
leads to same learning update rule $\theta \leftarrow \theta + \alpha(y^{(i)} - h_\theta(x^{(i)}))x^{(i)}$

Perceptron, Linear Models, and Linearly Separable Data

- logistic regression and perceptron can do perfectly when the data is (nicely) linearly separable.



- Line designates boundary of a "ridge" or "cliff" between the categories.



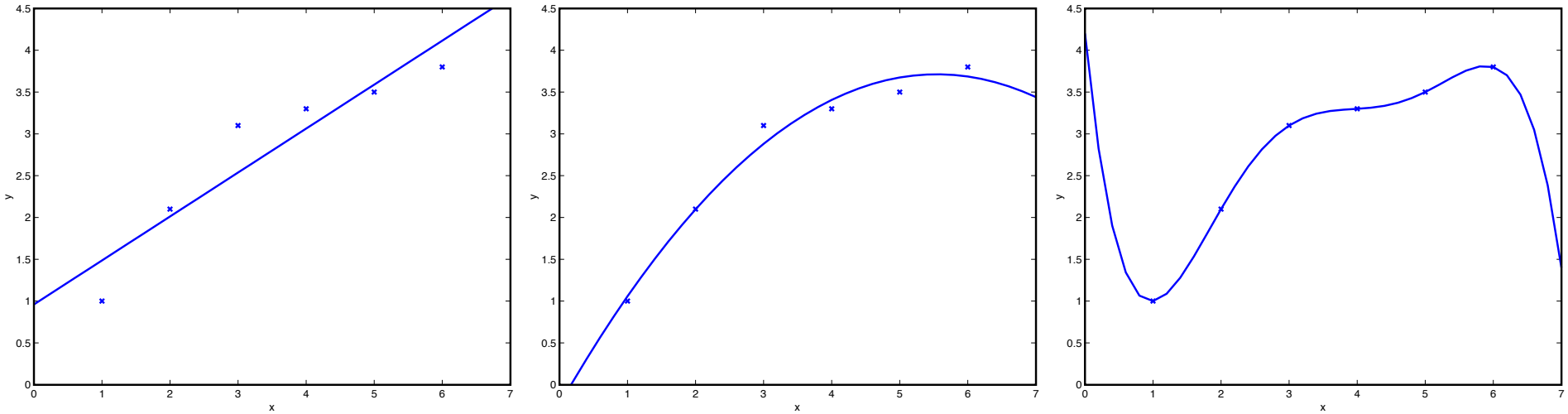
Review

- The next three slides are review from Lecture 1.

Review

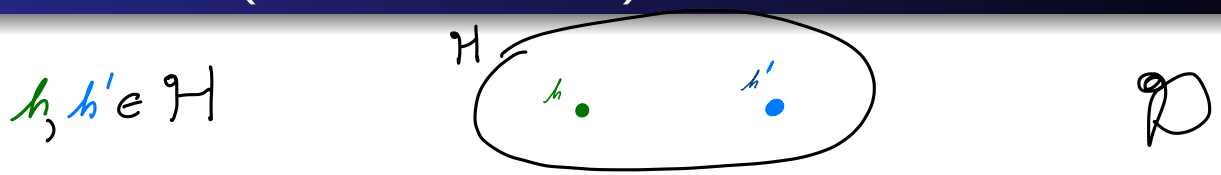
- The next three slides are review from Lecture 1.
- Please read writeup "Underfitting and Overfitting in Machine Learning" to be posted to canvas.

Underfitting vs. Overfitting



- Fit a model with various input features, values of powers of x, goal is to predict y based on xy-pair samples $\mathcal{D} = \{(x^{(i)}, y^{(i)}\}_i$.
- Fit models: left $y = \theta_0 + \theta_1 x$; middle $y = \theta_0 + \theta_1 x + \theta_2 x^2$; right $y = \sum_{j=0}^{5} \theta_j x^j$.
- Both left and right plots poorly fit the data, but they are poor for different reasons. The left could be underfitting, and the right could be overfitting. The center plot looks better.

Overfitting definition (T. Mitchell)



- We say that a hypothesis overfits the training examples if some other hypothesis that fits the training examples less well actually performs better over the entire distribution of instances (i.e., including instances beyond the training set).

Definition 2.5.1 (overfitting)

Given a hypothesis space $\mathcal{H}$, a hypothesis $h \in \mathcal{H}$ is said to overfit the training data if there exists some alternative hypothesis $h' \in \mathcal{H}$, such that h has smaller error than h' over the training examples, but h' has a smaller overall error than h over the entire distribution (or data set) of instances.

- We'll visit this topic again when we discuss bias/variance, but first lets discuss a few more models.

Some Definitions

- Any data set $\mathcal{D} = \left\{(x^{(1)}, y^{(1)}, (x^{(2)}, y^{(2)}), \ldots, (x^{(n)}, y^{(n)})\right\}$ drawn from the a given distribution, meaning that $(x^{(j)}, y^{(j)}) \sim p(x, y)$ for all $1 \leq j \leq n$.

Some Definitions

- Any data set $\mathcal{D} = \left\{(x^{(1)}, y^{(1)}, (x^{(2)}, y^{(2)}), \ldots, (x^{(n)}, y^{(n)})\right\}$ drawn from the a given distribution, meaning that $(x^{(j)}, y^{(j)}) \sim p(x, y)$ for all $1 \leq j \leq n$.
- Training a model by maximizing accuracy on a training set $\mathcal{D}_{\text{tr}}$:

$$\hat{h} \in \operatorname*{argmax}_{h \in \mathcal{H}} \sum_{(x,y) \in \mathcal{D}_{\text{tr}}} \mathcal{A}(y_i, h(x_i)) - \lambda \Omega(h) \tag{2.18}$$

$\mathcal{C}$ accuracy − regularization tradeoff coefficient

λ A hyperparameter.

$\mathcal{A}(y, h(x))$ is big if $h(x)$ is a good predictor of y and is small if $h(x)$ is a poor predictor of y.

Some Definitions

- Any data set $\mathcal{D} = \left\{(x^{(1)}, y^{(1)}, (x^{(2)}, y^{(2)}), \ldots, (x^{(n)}, y^{(n)})\right\}$ drawn from the a given distribution, meaning that $(x^{(j)}, y^{(j)}) \sim p(x, y)$ for all $1 \leq j \leq n$.
- Training a model by maximizing accuracy on a training set $\mathcal{D}_{\text{tr}}$:

$$\hat{h} \in \operatorname*{argmax}_{h \in \mathcal{H}} \sum_{(x,y) \in \mathcal{D}_{\text{tr}}} \mathcal{A}(y_i, h(x_i)) - \lambda \Omega(h) \tag{2.18}$$

- Accuracy according to the sample distribution

$$\text{accuracy}(\hat{h}) = E_{p(x,y)}[\mathcal{A}(y, \hat{h}(x))] = \int p(x, y) \mathcal{A}(y, \hat{h}(x))] dx dy \tag{2.19}$$

Some Definitions

- Any data set $\mathcal{D} = \left\{(x^{(1)}, y^{(1)}, (x^{(2)}, y^{(2)}), \ldots, (x^{(n)}, y^{(n)})\right\}$ drawn from the a given distribution, meaning that $(x^{(j)}, y^{(j)}) \sim p(x, y)$ for all $1 \leq j \leq n$.
- Training a model by maximizing accuracy on a training set $\mathcal{D}_{\text{tr}}$:

$$\hat{h} \in \operatorname*{argmax}_{h \in \mathcal{H}} \sum_{(x,y) \in \mathcal{D}_{\text{tr}}} \mathcal{A}(y_i, h(x_i)) - \lambda \Omega(h) \tag{2.18}$$

- Accuracy according to the sample distribution

$$\text{accuracy}(\hat{h}) = E_{p(x,y)}[\mathcal{A}(y, \hat{h}(x))] = \int p(x, y) \mathcal{A}(y, \hat{h}(x))] dx dy \tag{2.19}$$

- Accuracy of a trained model on a data set $\mathcal{D}$:

$$\text{accuracy}_{\mathcal{D}}(\hat{h}) = \frac{1}{|\mathcal{D}|} \sum_{(x,y) \in \mathcal{D}} \mathcal{A}(y_j, \hat{h}(x_j)) \tag{2.20}$$

Some Definitions

- Any data set $\mathcal{D} = \left\{(x^{(1)}, y^{(1)}, (x^{(2)}, y^{(2)}), \ldots, (x^{(n)}, y^{(n)})\right\}$ drawn from the a given distribution, meaning that $(x^{(j)}, y^{(j)}) \sim p(x, y)$ for all $1 \leq j \leq n$.
- Training a model by maximizing accuracy on a training set $\mathcal{D}_{\text{tr}}$:

$$\hat{h} \in \operatorname*{argmax}_{h \in \mathcal{H}} \sum_{(x,y) \in \mathcal{D}_{\text{tr}}} \mathcal{A}(y_i, h(x_i)) - \lambda \Omega(h) \tag{2.18}$$

- Accuracy according to the sample distribution

$$\text{accuracy}(\hat{h}) = E_{p(x,y)}[\mathcal{A}(y, \hat{h}(x))] = \int p(x, y) \mathcal{A}(y, \hat{h}(x))] dx dy \tag{2.19}$$

- Accuracy of a trained model on a data set $\mathcal{D}$:

$$\text{accuracy}_{\mathcal{D}}(\hat{h}) = \frac{1}{|\mathcal{D}|} \sum_{(x,y) \in \mathcal{D}} \mathcal{A}(y_j, \hat{h}(x_j)) \tag{2.20}$$

- Training data set $\mathcal{D}_{\text{tr}}$ and validation (or development) data set $\mathcal{D}_{\text{va}}$.

Overfitting

Definition 2.5.1 (overfitting)

We say that $\hat{h} \in \mathcal{H}$ **overfits** the training data $\mathcal{D}_{\text{tr}}$ if there exists $h' \in \mathcal{H}$ such that

$$\text{accuracy}_{\mathcal{D}_{\text{tr}}}(\hat{h}) > \text{accuracy}_{\mathcal{D}_{\text{tr}}}(h') \text{ and } \text{accuracy}(\hat{h}) < \text{accuracy}(h'). \tag{2.21}$$

Since we can't compute $\text{accuracy}(\hat{h})$ or $\text{accuracy}(h')$ as mentioned above, a practical definition of overfitting changes this to:

$$\text{accuracy}_{\mathcal{D}_{\text{tr}}}(\hat{h}) > \text{accuracy}_{\mathcal{D}_{\text{tr}}}(h') \text{ and } \text{accuracy}_{\mathcal{D}_{\text{va}}}(\hat{h}) < \text{accuracy}_{\mathcal{D}_{\text{va}}}(h'). \tag{2.22}$$

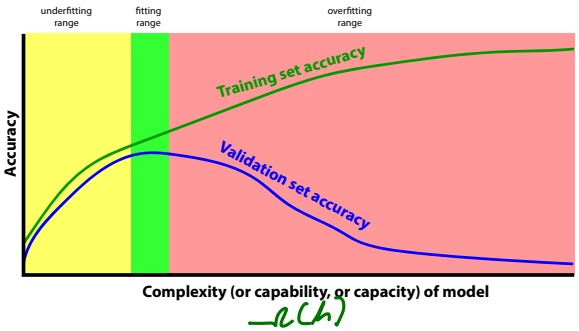
Underfitting

Definition 2.5.2 (underfitting)

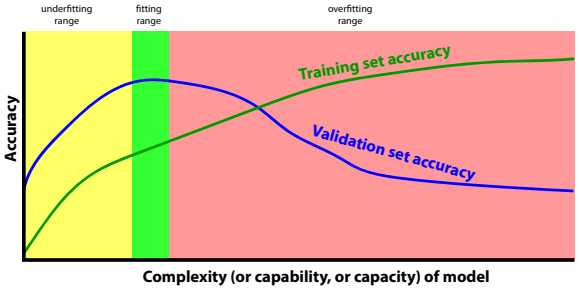
We say that $\hat{h} \in \mathcal{H}$ **underfits** the training data $\mathcal{D}_{\text{tr}}$ if there exists $h'' \in \mathcal{H}$ such that

$$\text{accuracy}_{\mathcal{D}_{\text{tr}}}(\hat{h}) < \text{accuracy}_{\mathcal{D}_{\text{tr}}}(h'') \quad \text{and} \quad \text{accuracy}(\hat{h}) < \text{accuracy}(h''). \tag{2.23}$$

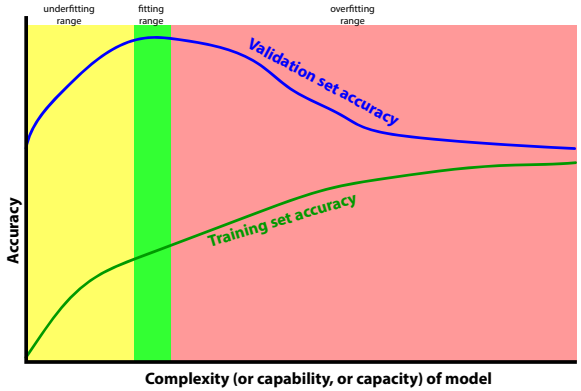
Overfitting and Underfitting : Validation Curves



(a) The highly typical case is when the training set accuracy is higher than the validation set accuracy.



(b) It is possible, but somewhat unlikely, that the validation set accuracy crosses the validation set accuracy. The overfitting/underfitting ranges are still the same.

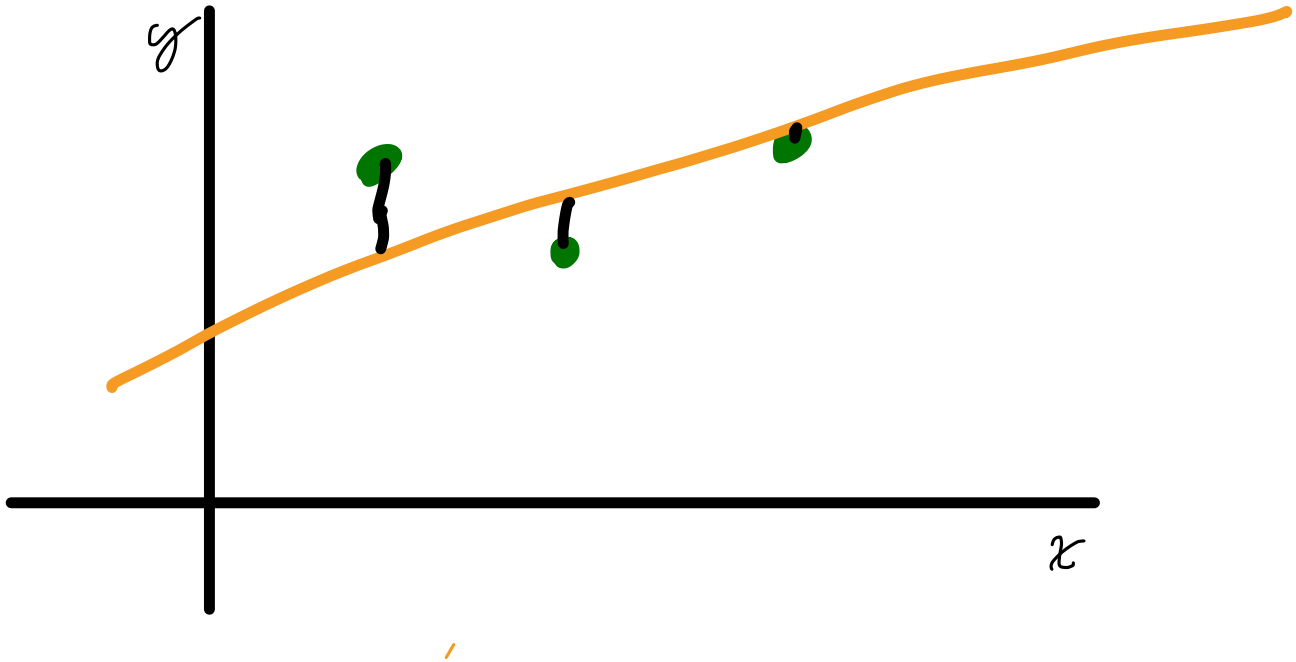


(c) It is possible, but very unlikely, that the validation set accuracy is higher than the training set accuracy, as shown in this plot. But the overfitting/underfitting ranges are still the same.

Figure: Overfitting and underfitting shown as a function of model complexity for a fixed training set size n_{tr}. Any h in the red region overfits the training set. Any h in the yellow region underfits the training set. Any h in the green (middle) region properly fits the training set. The regions are based all on the accuracy $\text{accuracy}_{\mathcal{D}_{\text{va}}}(h)$ computed on an validation data set $\mathcal{D}_{\text{va}}$, but the same principle would be true if it were possible to measure accuracy it on the entire distribution $\text{accuracy}(h)$.

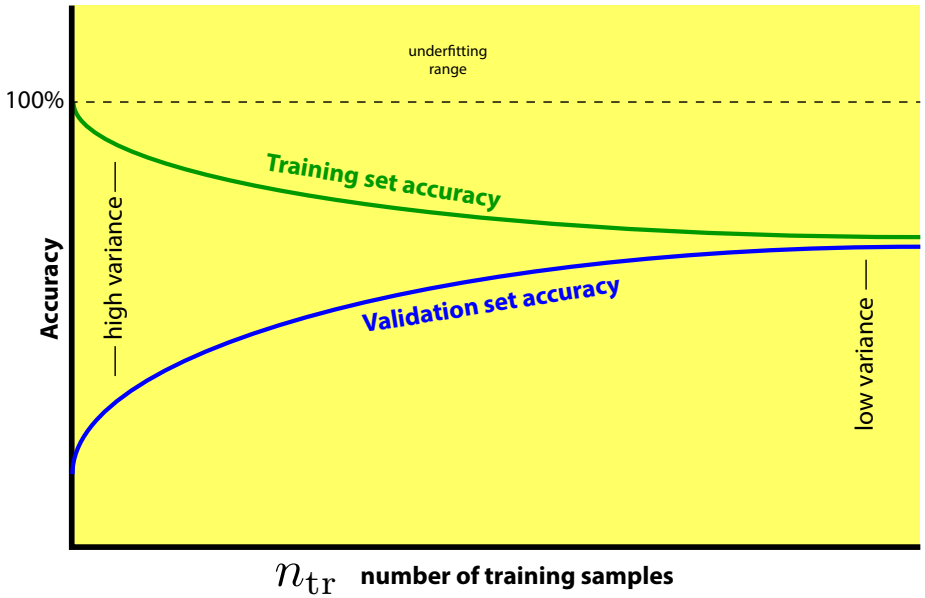
Learning Curves

How does overfitting/underfitting depends on the size of the training data set?

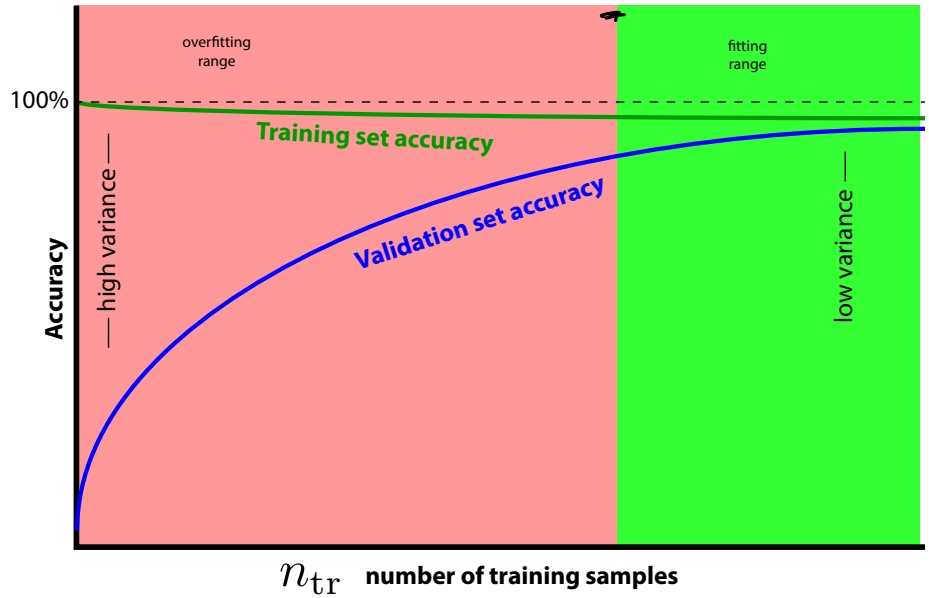


Learning Curves

How does overfitting/underfitting depends on the size of the training data set? A given model (with fixed complexity) will tend to overfit a small training data set, and underfit a large training data set.



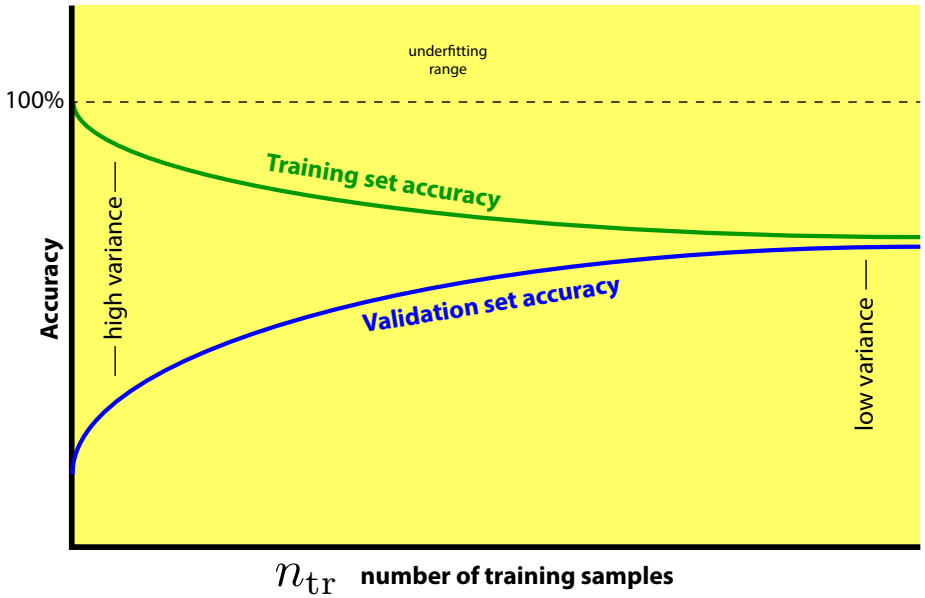
(a) Learning curve with a low complexity (i.e., $\Omega(h)$ small) model.



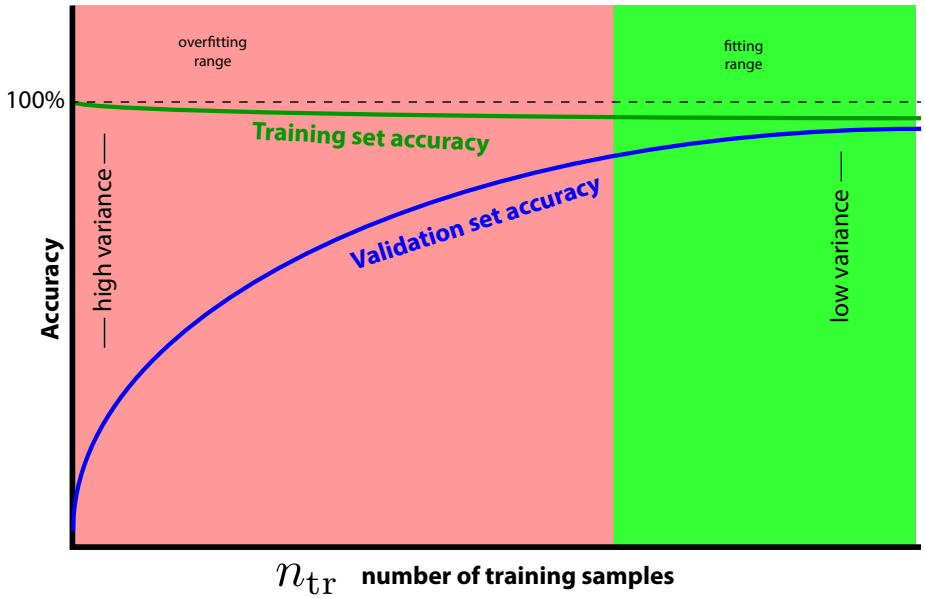
(b) Learning curve with a high complexity (i.e., $\Omega(h)$ large) model.

Learning Curves

How does overfitting/underfitting depends on the size of the training data set? A given model (with fixed complexity) will tend to overfit a small training data set, and underfit a large training data set.



(a) Learning curve with a low complexity (i.e., $\Omega(h)$ small) model.



(b) Learning curve with a high complexity (i.e., $\Omega(h)$ large) model.

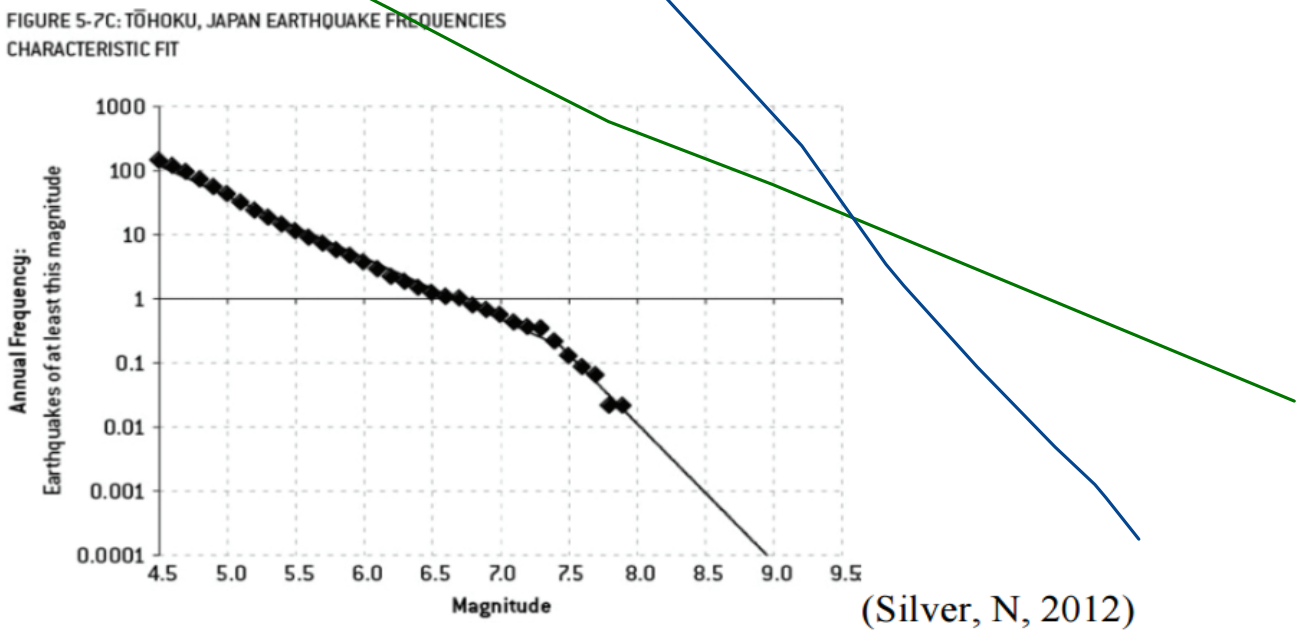
On the left, the model underfits since the model does not have much capability even with much training data. On the right, the model starts out overfitting but eventually fits with enough training data.

Real-world consequences of overfitting

- The Fukushima nuclear power plant disaster in 2011 caused by overfitting a model to data!! see https://ml.berkeley.edu/blog/2017/07/13/tutorial-4/ and https://mpra.ub.uni-muenchen.de/69383/1/MPRA_paper_69383.pdf

Real-world consequences of overfitting

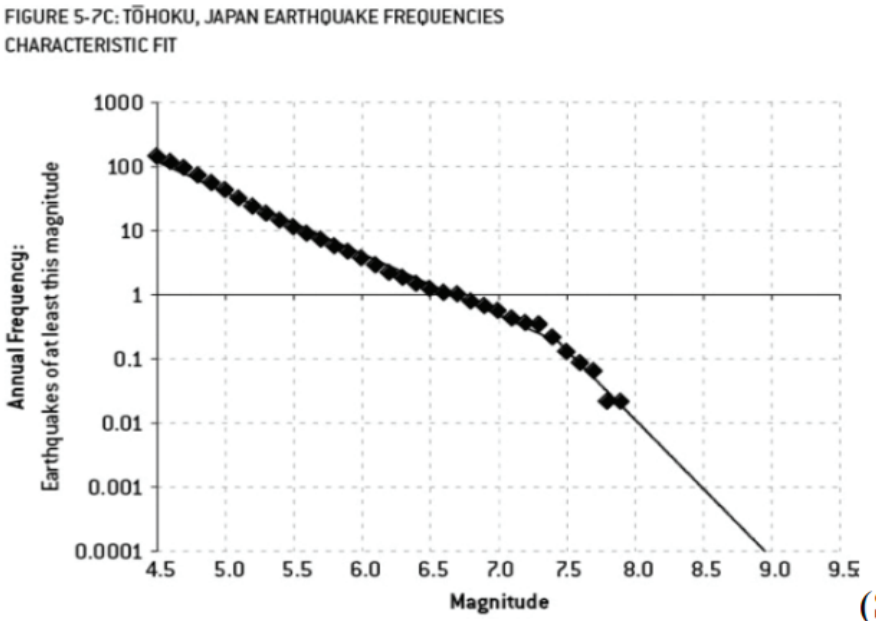
- The Fukushima nuclear power plant disaster in 2011 caused by overfitting a model to data!! see `https://ml.berkeley.edu/blog/2017/07/13/tutorial-4/` and `https://mpra.ub.uni-muenchen.de/69383/1/MPRA_paper_69383.pdf`
- earthquake data going back 400 years and model fit



(Silver, N, 2012)

Real-world consequences of overfitting

- The Fukushima nuclear power plant disaster in 2011 caused by overfitting a model to data!! see `https://ml.berkeley.edu/blog/2017/07/13/tutorial-4/` and `https://mpra.ub.uni-muenchen.de/69383/1/MPRA_paper_69383.pdf`
- earthquake data going back 400 years and model fit

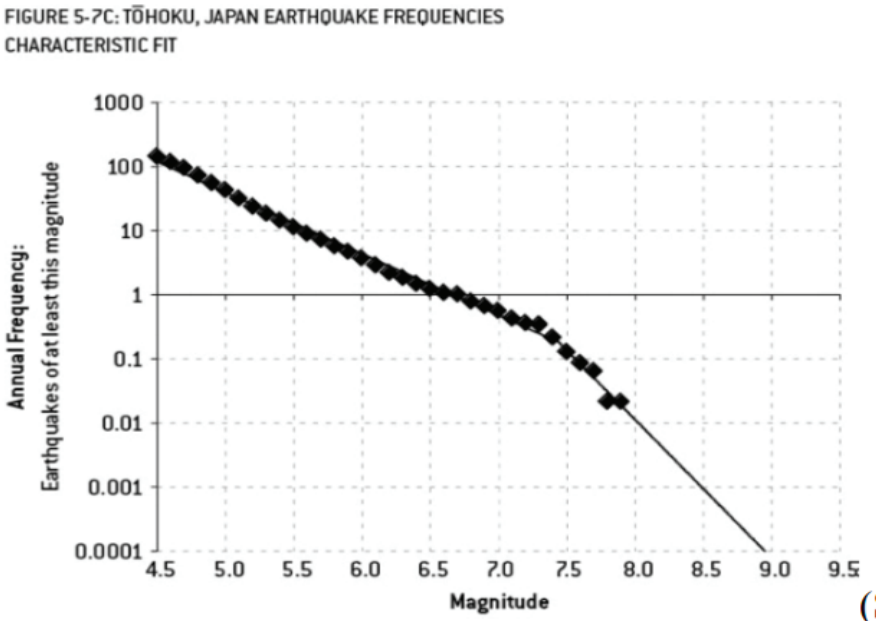


(Silver, N, 2012)

- Non-linear model fit, predicts mag. 9 earthquake every ≈13000 years.

Real-world consequences of overfitting

- The Fukushima nuclear power plant disaster in 2011 caused by overfitting a model to data!! see https://ml.berkeley.edu/blog/2017/07/13/tutorial-4/ and https://mpra.ub.uni-muenchen.de/69383/1/MPRA_paper_69383.pdf
- earthquake data going back 400 years and model fit

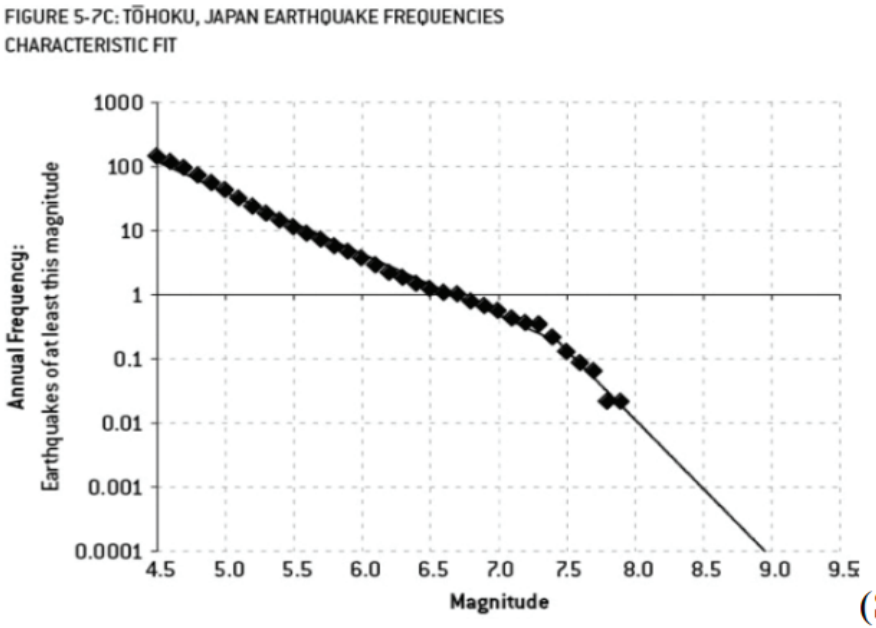


(Silver, N, 2012)

- Non-linear model fit, predicts mag. 9 earthquake every ≈13000 years.
- This was used to design Fukushima Daiichi nuclear power plant, to withstand 8.6 magnitude earthquake, tsunami of 5.7 meters.

Real-world consequences of overfitting

- The Fukushima nuclear power plant disaster in 2011 caused by overfitting a model to data!! see https://ml.berkeley.edu/blog/2017/07/13/tutorial-4/ and https://mpra.ub.uni-muenchen.de/69383/1/MPRA_paper_69383.pdf
- earthquake data going back 400 years and model fit

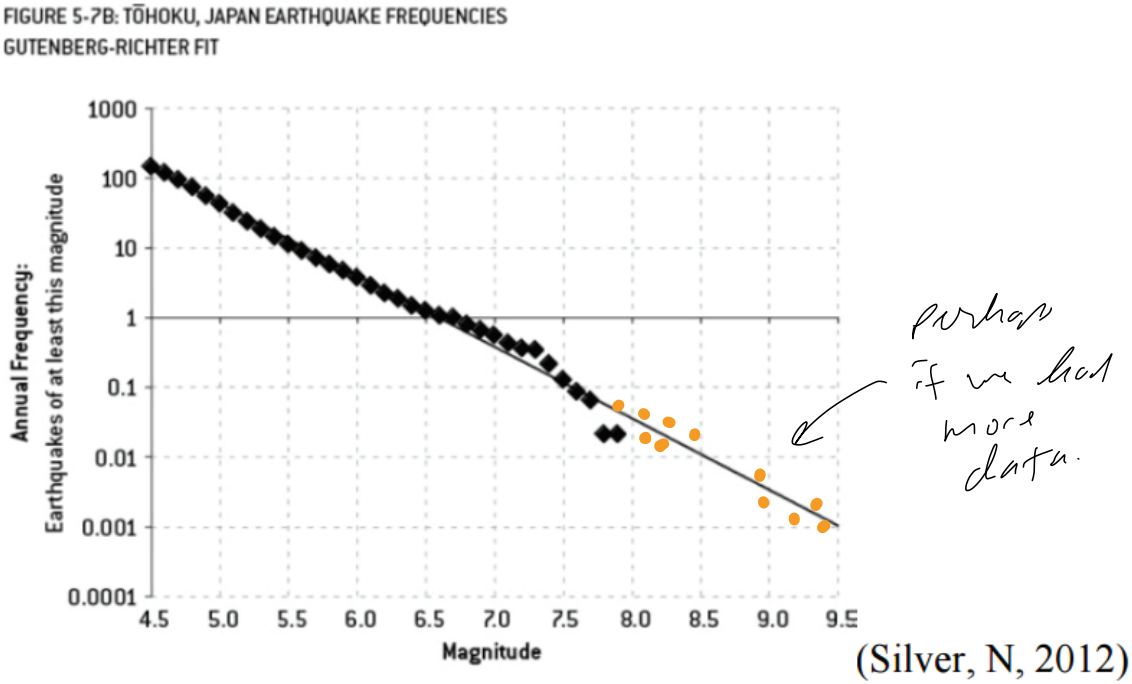


(Silver, N, 2012)

- Non-linear model fit, predicts mag. 9 earthquake every ≈13000 years.
- This was used to design Fukushima Daiichi nuclear power plant, to withstand 8.6 magnitude earthquake, tsunami of 5.7 meters.
- In 2011, 9.0 earthquake and 14 meter tsunami!!

Real-world consequences of overfitting

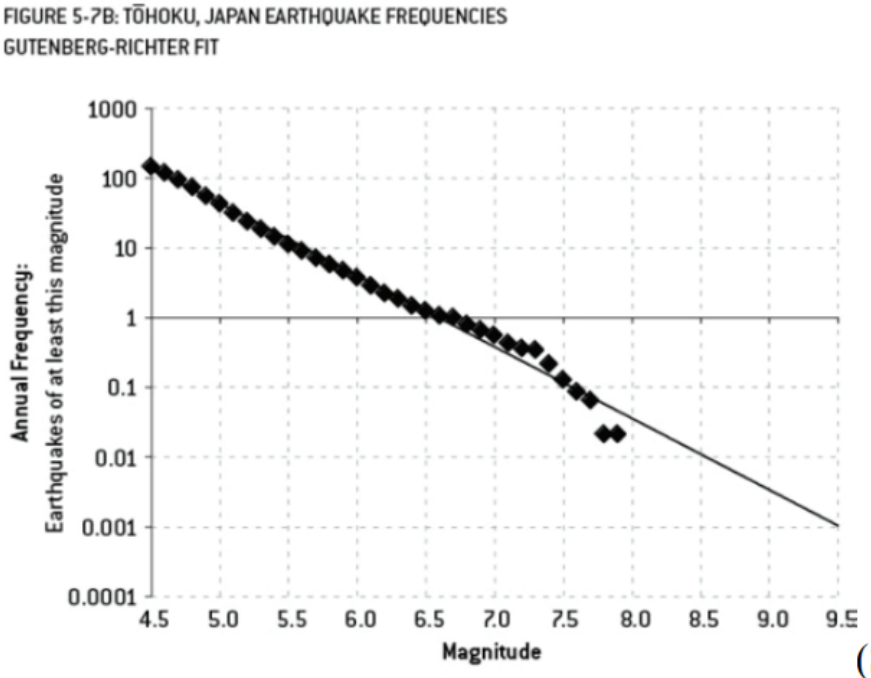
- log-linear model shows different trend:



(Silver, N, 2012)

Real-world consequences of overfitting

- log-linear model shows different trend:

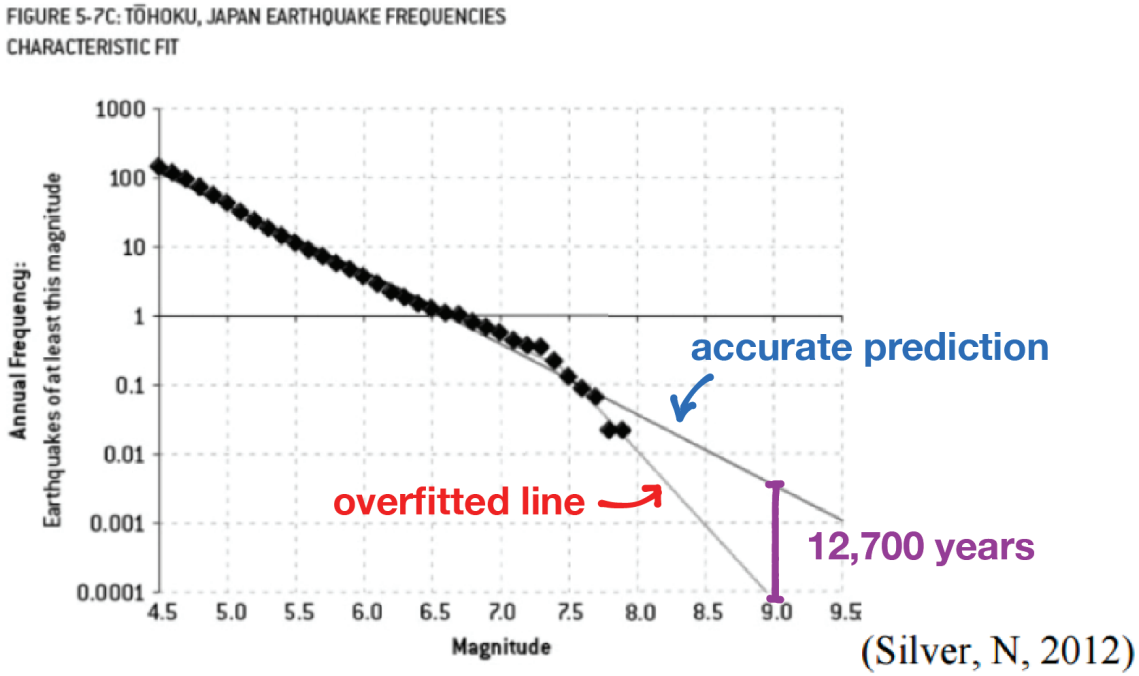


(Silver, N, 2012)

- linear model fit, predicts magnitude 9 earthquake about every 300 years, would have led to quite a different reactor design.

Real-world consequences of overfitting

- log-linear model shows different trend:



(Silver, N, 2012)

- linear model fit, predicts magnitude 9 earthquake about every 300 years, would have led to quite a different reactor design.
- Moral: Overfitting (or underfitting) can have huge consequences!

Overfitting slider on the web

See "Visualization of the bias-variance tradeoff" at the following link entitled "The Bias-Variance Dilemma": `https://medium.com/@ml.at.berkeley/machine-learning-crash-course-part-4-the-bias-variance-dilemma-a94e60ec1d3`

The random process of training

- Recall training data $\mathcal{D} = \left\{(x^{(i)}, y^{(i)})\right\}_{i=1}^{n}$ where $(x^{(i)}, y^{(i)}) \sim p(x, y)$ are drawn from some distribution, $x^{(i)} \in \mathbb{R}^m$ and $y^{(i)} \in \mathbb{R}$.

The random process of training

- Recall training data $\mathcal{D} = \left\{(x^{(i)}, y^{(i)})\right\}_{i=1}^{n}$ where $(x^{(i)}, y^{(i)}) \sim p(x, y)$ are drawn from some distribution, $x^{(i)} \in \mathbb{R}^m$ and $y^{(i)} \in \mathbb{R}$.
- Training data is a random sample, and is itself random.

$$P_r(\mathcal{D}) = \prod_{i=1}^{n} p(x^{(i)}, y^{(i)})$$

$\Rightarrow$ $\mathcal{D}$ is a random variable

The random process of training

- Recall training data $\mathcal{D} = \left\{(x^{(i)}, y^{(i)})\right\}_{i=1}^{n}$ where $(x^{(i)}, y^{(i)}) \sim p(x, y)$ are drawn from some distribution, $x^{(i)} \in \mathbb{R}^m$ and $y^{(i)} \in \mathbb{R}$.
- Training data is a random sample, and is itself random.
- Fitting a model h_θ where θ is derived from the training procedure.

The random process of training

- Recall training data $\mathcal{D} = \left\{(x^{(i)}, y^{(i)})\right\}_{i=1}^{n}$ where $(x^{(i)}, y^{(i)}) \sim p(x, y)$ are drawn from some distribution, $x^{(i)} \in \mathbb{R}^m$ and $y^{(i)} \in \mathbb{R}$.
- Training data is a random sample, and is itself random.
- Fitting a model h_θ where θ is derived from the training procedure.
- We can think of $\theta(\mathcal{D})$ is the parameters of the model via the process of model fitting. $\theta(\mathcal{D})$ is a random variable since deterministic function of random sample $\mathcal{D}$ of size n.

The random process of training

- Recall training data $\mathcal{D} = \left\{(x^{(i)}, y^{(i)})\right\}_{i=1}^{n}$ where $(x^{(i)}, y^{(i)}) \sim p(x, y)$ are drawn from some distribution, $x^{(i)} \in \mathbb{R}^m$ and $y^{(i)} \in \mathbb{R}$.
- Training data is a random sample, and is itself random.
- Fitting a model h_θ where θ is derived from the training procedure.
- We can think of $\theta(\mathcal{D})$ is the parameters of the model via the process of model fitting. $\theta(\mathcal{D})$ is a random variable since deterministic function of random sample $\mathcal{D}$ of size n.
- Each time we draw a different training set, we might get (hopefully only very slightly) different θ.

Bias/Variance Intuition (on one slide).

When are fitting a model h_θ to data $\mathcal{D}$, and we optimize (or can chose) any model in the model family (e.g., linear model family, logistic model, do we include higher order polynomial features, etc.).

Bias/Variance Intuition (on one slide).

When are fitting a model h_θ to data $\mathcal{D}$, and we optimize (or can chose) any model in the model family (e.g., linear model family, logistic model, do we include higher order polynomial features, etc.).

High bias, low variance: (underfitting)

Bias/Variance Intuition (on one slide).

When are fitting a model h_θ to data $\mathcal{D}$, and we optimize (or can chose) any model in the model family (e.g., linear model family, logistic model, do we include higher order polynomial features, etc.).

High bias, low variance: (underfitting)

- if model family is too simple (low complexity), it will never match important characteristics in the data. It is, in such case, biased.

"inductive bias"

const. model

$h(x) = c$

Bias/Variance Intuition (on one slide).

When are fitting a model h_θ to data $\mathcal{D}$, and we optimize (or can chose) any model in the model family (e.g., linear model family, logistic model, do we include higher order polynomial features, etc.).

High bias, low variance: (underfitting)

- if model family is too simple (low complexity), it will never match important characteristics in the data. It is, in such case, biased.
- Even if we train on multiple different data sets, we'll get approximately the same model h_θ since the model family is not that capable or flexible. All are wrong in same way.

Bias/Variance Intuition (on one slide).

When are fitting a model h_θ to data $\mathcal{D}$, and we optimize (or can chose) any model in the model family (e.g., linear model family, logistic model, do we include higher order polynomial features, etc.).

High bias, low variance: (underfitting)

- if model family is too simple (low complexity), it will never match important characteristics in the data. It is, in such case, biased.
- Even if we train on multiple different data sets, we'll get approximately the same model h_θ since the model family is not that capable or flexible. All are wrong in same way.
- Thus, the variance of the random variable $h_{\theta(\mathcal{D})}$ is low!

$\mathrm{var}(h_{\theta(\mathcal{D})})$ small.

a r.v.

Bias/Variance Intuition (on one slide).

When are fitting a model h_θ to data $\mathcal{D}$, and we optimize (or can chose) any model in the model family (e.g., linear model family, logistic model, do we include higher order polynomial features, etc.).

High bias, low variance: (underfitting)

- if model family is too simple (low complexity), it will never match important characteristics in the data. It is, in such case, biased.
- Even if we train on multiple different data sets, we'll get approximately the same model h_θ since the model family is not that capable or flexible. All are wrong in same way.
- Thus, the variance of the random variable $h_{\theta(\mathcal{D})}$ is low!

Low bias, high variance: (overfitting)

Bias/Variance Intuition (on one slide).

When are fitting a model h_θ to data $\mathcal{D}$, and we optimize (or can chose) any model in the model family (e.g., linear model family, logistic model, do we include higher order polynomial features, etc.).

High bias, low variance: (underfitting)

- if model family is too simple (low complexity), it will never match important characteristics in the data. It is, in such case, biased.
- Even if we train on multiple different data sets, we'll get approximately the same model h_θ since the model family is not that capable or flexible. All are wrong in same way.
- Thus, the variance of the random variable $h_{\theta(\mathcal{D})}$ is low!

Low bias, high variance: (overfitting)

- if model family is too complex, it will match unimportant idiosyncrasies in the data. Bias is low, since it matches the data quite well.

Bias/Variance Intuition (on one slide).

When are fitting a model h_θ to data $\mathcal{D}$, and we optimize (or can chose) any model in the model family (e.g., linear model family, logistic model, do we include higher order polynomial features, etc.).

High bias, low variance: (underfitting)

- if model family is too simple (low complexity), it will never match important characteristics in the data. It is, in such case, biased.
- Even if we train on multiple different data sets, we'll get approximately the same model h_θ since the model family is not that capable or flexible. All are wrong in same way.
- Thus, the variance of the random variable $h_{\theta(\mathcal{D})}$ is low!

Low bias, high variance: (overfitting)

- if model family is too complex, it will match unimportant idiosyncrasies in the data. Bias is low, since it matches the data quite well.
- When we train on multiple different data sets, each with its own (often random) idiosyncrasies, we'll get very different models h_θ since the model family can match any idiosyncrasies in the data.

Bias/Variance Intuition (on one slide).

When are fitting a model h_θ to data $\mathcal{D}$, and we optimize (or can chose) any model in the model family (e.g., linear model family, logistic model, do we include higher order polynomial features, etc.).

High bias, low variance: (underfitting)

- if model family is too simple (low complexity), it will never match important characteristics in the data. It is, in such case, biased.
- Even if we train on multiple different data sets, we'll get approximately the same model h_θ since the model family is not that capable or flexible. All are wrong in same way.
- Thus, the variance of the random variable $h_{\theta(\mathcal{D})}$ is low!

Low bias, high variance: (overfitting)

- if model family is too complex, it will match unimportant idiosyncrasies in the data. Bias is low, since it matches the data quite well.
- When we train on multiple different data sets, each with its own (often random) idiosyncrasies, we'll get very different models h_θ since the model family can match any idiosyncrasies in the data.
- Thus, the variance of the random variable $h_{\theta(\mathcal{D})}$ is high!

Bias/Variance

- Recall lecture 1 goal: find $h_\theta : \mathcal{X} \to \mathcal{Y}$ with minimum error.

Bias/Variance

- Recall lecture 1 goal: find $h_\theta : \mathcal{X} \to \mathcal{Y}$ with minimum error.
- Best solution, $\text{argmin}_h E_{p(x,y)}[(y - h(x))^2]$ leads to

$$h^*(x) = \int yp(y|x)dy = E[Y|x] = \text{ best association .} \tag{2.24}$$

assuming we have the distribution p, the resources to compute $E[Y|x]$, and the model family spanned by h includes the functional form of $E[Y|x]$ (i.e., $E[Y|x]$ is realizable, as apposed to the agnostic case which is not realizable). This is best we can do in theory.

Bias/Variance

- Recall lecture 1 goal: find $h_\theta : \mathcal{X} \to \mathcal{Y}$ with minimum error.
- Best solution, $\operatorname{argmin}_h E_{p(x,y)}[(y - h(x))^2]$ leads to

$$h^*(x) = \int y p(y|x) dy = E[Y|x] = \text{ best association .} \tag{2.24}$$

assuming we have the distribution p, the resources to compute $E[Y|x]$, and the model family spanned by h includes the functional form of $E[Y|x]$ (i.e., $E[Y|x]$ is realizable, as apposed to the agnostic case which is not realizable). This is best we can do in theory.

- In practice, we have training data $\mathcal{D}$ and a limited model family $\mathcal{H}$ (e.g., linear models) and instead do:

$$\tilde{h} \in \operatorname*{argmin}_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - h(x^{(i)}))^2 \tag{2.25}$$

Bias/Variance

- Recall lecture 1 goal: find $h_\theta : \mathcal{X} \to \mathcal{Y}$ with minimum error.
- Best solution, $\text{argmin}_h E_{p(x,y)}[(y - h(x))^2]$ leads to

$$h^*(x) = \int y p(y|x) dy = E[Y|x] = \text{ best association .} \tag{2.24}$$

assuming we have the distribution p, the resources to compute $E[Y|x]$, and the model family spanned by h includes the functional form of $E[Y|x]$ (i.e., $E[Y|x]$ is realizable, as apposed to the agnostic case which is not realizable). This is best we can do in theory.

- In practice, we have training data $\mathcal{D}$ and a limited model family $\mathcal{H}$ (e.g., linear models) and instead do:

$$\tilde{h} \in \text{argmin}_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - h(x^{(i)}))^2 \tag{2.25}$$

- Two reasons we might not be good: (a) wrong model family $\mathcal{H}$ and (b) not enough data (n too small).

Validation and Bias/Variance

How do we measure how good we are?

- We could consider measuring on the same training data we've got:

$$\frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \tilde{h}(x^{(i)}))^2 = \frac{1}{|\mathcal{D}|}\sum_{(x,y)\in\mathcal{D}}(y - \tilde{h}(x))^2 \tag{2.26}$$

  Already, intuitively bad for bias/variance reasons we've discussed, each $\mathcal{D}$ gives different solution, each $h_{\theta(\mathcal{D})}$ can look good on its own data.

Validation and Bias/Variance

How do we measure how good we are?

- We could consider measuring on the same training data we've got:

$$\frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \tilde{h}(x^{(i)}))^2 = \frac{1}{|\mathcal{D}|}\sum_{(x,y)\in\mathcal{D}}(y - \tilde{h}(x))^2 \tag{2.26}$$

  Already, intuitively bad for bias/variance reasons we've discussed, each $\mathcal{D}$ gives different solution, each $h_{\theta(\mathcal{D})}$ can look good on its own data.
- Ideal approach, tests all data (e.g., includes future samples we didn't train on).

$$E_{p(x,y)}[(y - \tilde{h}(x))^2] \tag{2.27}$$

  bad since impractical at best and (more likely) impossible.

Validation and Bias/Variance

How do we measure how good we are?

- We could consider measuring on the same training data we've got:

$$\frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \tilde{h}(x^{(i)}))^2 = \frac{1}{|\mathcal{D}|}\sum_{(x,y)\in\mathcal{D}}(y - \tilde{h}(x))^2 \tag{2.26}$$

  Already, intuitively bad for bias/variance reasons we've discussed, each $\mathcal{D}$ gives different solution, each $h_{\theta(\mathcal{D})}$ can look good on its own data.
- Ideal approach, tests all data (e.g., includes future samples we didn't train on).

$$E_{p(x,y)}[(y - \tilde{h}(x))^2] \tag{2.27}$$

  bad since impractical at best and (more likely) impossible.
- Typical approach, draw a separate validation data set $\mathcal{D}_{\text{va}}$, with $\mathcal{D}_{\text{va}} \cap \mathcal{D} = \emptyset$, and to try to get at generalization error, do:

$$\frac{1}{|\mathcal{D}_{\text{va}}|}\sum_{(x,y)\in\mathcal{D}_{\text{va}}}(y - \tilde{h}(x))^2 \tag{2.28}$$

Bias/Variance, Overall Error

- Ideal solution $h^*(x) = E[Y|x]$ and random estimate from data set $h_{\mathcal{D}}(x) \triangleq h_{\theta(\mathcal{D})} = \text{argmin}_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - h(x^{(i)}))^2$

Bias/Variance, Overall Error

- Ideal solution $h^*(x) = E[Y|x]$ and random estimate from data set $h_{\mathcal{D}}(x) \triangleq h_{\theta(\mathcal{D})} = \text{argmin}_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - h(x^{(i)}))^2$
- Lets consider measuring the overall error for any h:

$$\begin{aligned}
\text{error}(h) &= E_{p(x,y)}[(h(x) - y)^2] = \int (h(x) - y)^2 p(y|x) p(x) dx dy && (2.29)\\
&= E_{p(x,y)}[(h(x) - E[Y|x] + E[Y|x] - y)^2] && (2.30)\\
&= E_{p(x,y)}\Big[(h(x) - E[Y|x])^2 && (2.31)\\
&\quad + 2(h(x) - E[Y|x])(E[Y|x] - y) && (2.32)\\
&\quad + (E[Y|x] - y)^2\Big] && (2.33)\\
&= E_{p(x,y)}[(h(x) - E[Y|x])^2] && (2.34)\\
&\quad + 2(E[h(x)] - E[Y|X])(E[Y|x] - E[Y|x]) && (2.35)\\
&\quad + E_{p(x,y)}[(E[Y|x] - y)^2] && (2.36)
\end{aligned}$$

Note, 2nd term cancels out.

2nd Term Cancels Out

- Why does second term cancel out?

2nd Term Cancels Out

- Why does second term cancel out?
- $E[Y|x] = \int yp(y|x)dy =$ best association and is a deterministic function of x, an expected value of the random variable Y under conditional distribution $p(y|x)$ for a given x.

2nd Term Cancels Out

- Why does second term cancel out?
- $E[Y|x] = \int yp(y|x)dy =$ best association and is a deterministic function of x, an expected value of the random variable Y under conditional distribution $p(y|x)$ for a given x. Therefore,

$$E_{p(x,y)}\Big[2(h(x) - E[Y|x])(E[Y|x] - y)\Big] \tag{2.37}$$

$$= 2\int\int p(x)p(y|x)\Big[(h(x) - E[Y|x])(E[Y|x] - y)\Big]dydx \tag{2.38}$$

$$= 2\int p(x)(h(x) - E[Y|x])\Big[\int p(y|x)(E[Y|x] - y)dy\Big]dx \tag{2.39}$$

$$= 2\int p(x)(h(x) - E[Y|x])\Big[(E[Y|x] - E[Y|x])\Big]dx = 0 \tag{2.40}$$

2nd Term Cancels Out

- Why does second term cancel out?
- $E[Y|x] = \int yp(y|x)dy =$ best association and is a deterministic function of x, an expected value of the random variable Y under conditional distribution $p(y|x)$ for a given x. Therefore,

$$E_{p(x,y)}\Big[2(h(x) - E[Y|x])(E[Y|x] - y)\Big] \tag{2.37}$$

$$= 2\int\int p(x)p(y|x)\Big[(h(x) - E[Y|x])(E[Y|x] - y)\Big]dydx \tag{2.38}$$

$$= 2\int p(x)(h(x) - E[Y|x])\Big[\int p(y|x)(E[Y|x] - y)dy\Big]dx \tag{2.39}$$

$$= 2\int p(x)(h(x) - E[Y|x])\Big[(E[Y|x] - E[Y|x])\Big]dx = 0 \tag{2.40}$$

- Aside: $E[Y|X] = \int yp(y|X)dy$ is a deterministic function of r.v. X, and hence $E[Y|X]$ is itself a r.v. with a mean $E[E[Y|X]]$.

2nd Term Cancels Out

- Why does second term cancel out?
- $E[Y|x] = \int yp(y|x)dy =$ best association and is a deterministic function of x, an expected value of the random variable Y under conditional distribution $p(y|x)$ for a given x. Therefore,

$$E_{p(x,y)}\Big[2(h(x) - E[Y|x])(E[Y|x] - y)\Big] \tag{2.37}$$

$$= 2\int\int p(x)p(y|x)\Big[(h(x) - E[Y|x])(E[Y|x] - y)\Big]dydx \tag{2.38}$$

$$= 2\int p(x)(h(x) - E[Y|x])\Big[\int p(y|x)(E[Y|x] - y)dy\Big]dx \tag{2.39}$$

$$= 2\int p(x)(h(x) - E[Y|x])\Big[(E[Y|x] - E[Y|x])\Big]dx = 0 \tag{2.40}$$

- Asside: $E[Y|X] = \int yp(y|X)dy$ is a deterministic function of r.v. X, and hence $E[Y|X]$ is itself a r.v. with a mean $E[E[Y|X]]$.
- Thus, $E[E[Y|X]] = \int E[Y|x]p(x)dx = \int \left[\int yp(y|x)p(x)dy\right]dx = \int y\left[\int p(y|x)p(x)dx\right]dy = \int yp(y)dy = E[Y]$.

Bias/Variance, the r.v. $h_{\mathcal{D}}(x)$

- Thus the error has only two terms:

$$\text{error}(h) = E_{p(x,y)}[(h(x) - E[Y|x])^2] + E_{p(x,y)}[(E[Y|x] - y)^2] \quad (2.41)$$

Bias/Variance, the r.v. $h_{\mathcal{D}}(x)$

- Thus the error has only two terms:

$$\text{error}(h) = E_{p(x,y)}[(h(x) - E[Y|x])^2] + E_{p(x,y)}[(E[Y|x] - y)^2] \qquad (2.41)$$

- Second term has nothing to do with the model h whose error we are measuring, and is the inherent error, due to the random process and label noise. We can neglect this term in our study of bias/variance.

Bias/Variance, the r.v. $h_{\mathcal{D}}(x)$

- Thus the error has only two terms:

$$\text{error}(h) = E_{p(x,y)}[(h(x) - E[Y|x])^2] + E_{p(x,y)}[(E[Y|x] - y)^2] \qquad (2.41)$$

- Second term has nothing to do with the model h whose error we are measuring, and is the inherent error, due to the random process and label noise. We can neglect this term in our study of bias/variance.
- The first term can be simplified to $E_{p(x)}[(h(x) - E[Y|x])^2]$ and is the learning error (or MSE). It is zero when $h(x) = h^*(x) = E[Y|x]$.

Bias/Variance, the r.v. $h_{\mathcal{D}}(x)$

- Thus the error has only two terms:

$$\text{error}(h) = E_{p(x,y)}[(h(x) - E[Y|x])^2] + E_{p(x,y)}[(E[Y|x] - y)^2] \qquad (2.41)$$

- Second term has nothing to do with the model h whose error we are measuring, and is the inherent error, due to the random process and label noise. We can neglect this term in our study of bias/variance.
- The first term can be simplified to $E_{p(x)}[(h(x) - E[Y|x])^2]$ and is the learning error (or MSE). It is zero when $h(x) = h^*(x) = E[Y|x]$.
- $h_{\mathcal{D}}(x)$ is a (random) learnt model from (random variable) data set $\mathcal{D}$. Hence, $h_{\mathcal{D}}(x)$ is a random variable (a deterministic function, the learning/optimization process, of random variable $\mathcal{D}$ data set) and it has a mean $E_{\mathcal{D}}[h_{\mathcal{D}}(x)]$ and a variance $E_{\mathcal{D}}\left[\left(h_{\mathcal{D}}(x) - E_{\mathcal{D}}[h_{\mathcal{D}}(x)]\right)^2\right]$.

Bias/Variance, the r.v. $h_{\mathcal{D}}(x)$

- Thus the error has only two terms:

$$\text{error}(h) = E_{p(x,y)}[(h(x) - E[Y|x])^2] + E_{p(x,y)}[(E[Y|x] - y)^2] \quad (2.41)$$

- Second term has nothing to do with the model h whose error we are measuring, and is the inherent error, due to the random process and label noise. We can neglect this term in our study of bias/variance.
- The first term can be simplified to $E_{p(x)}[(h(x) - E[Y|x])^2]$ and is the learning error (or MSE). It is zero when $h(x) = h^*(x) = E[Y|x]$.
- $h_{\mathcal{D}}(x)$ is a (random) learnt model from (random variable) data set $\mathcal{D}$. Hence, $h_{\mathcal{D}}(x)$ is a random variable (a deterministic function, the learning/optimization process, of random variable $\mathcal{D}$ data set) and it has a mean $E_{\mathcal{D}}[h_{\mathcal{D}}(x)]$ and a variance $E_{\mathcal{D}}\left[\left(h_{\mathcal{D}}(x) - E_{\mathcal{D}}[h_{\mathcal{D}}(x)]\right)^2\right]$.
- To clarify notation, we'll use $\mathcal{D}$ and $\bar{\mathcal{D}}$ to express things like $h_{\mathcal{D}}(x)$'s variance $E_{\mathcal{D}}\left[\left(h_{\mathcal{D}}(x) - E_{\bar{\mathcal{D}}}[h_{\bar{\mathcal{D}}}(x)]\right)^2\right]$ but $\mathcal{D}$ and $\bar{\mathcal{D}}$ are two independent and identically distributed (iid) random variables over datasets.

Bias/Variance Breakdown

- We further analyze only this first term, and take ensemble averages over $\mathcal{D}$, the random data sample. $h_{\mathcal{D}}(x)$ is a (random) learnt model from (random variable) data set $\mathcal{D}$.

Bias/Variance Breakdown

- We further analyze only this first term, and take ensemble averages over $\mathcal{D}$, the random data sample. $h_{\mathcal{D}}(x)$ is a (random) learnt model from (random variable) data set $\mathcal{D}$. For a given x, we have

$$
\begin{aligned}
E_{\mathcal{D}}[(h_{\mathcal{D}}(x) - E[Y|x])^2] &= E_{\mathcal{D}}[(h_{\mathcal{D}}(x) - E_{\bar{\mathcal{D}}}[h_{\bar{\mathcal{D}}}(x)] + E_{\bar{\mathcal{D}}}[h_{\bar{\mathcal{D}}}(x)] - E[Y|x])^2] \\
&= E_{\mathcal{D}}\Big[(h_{\mathcal{D}}(x) - E_{\bar{\mathcal{D}}}[h_{\bar{\mathcal{D}}}(x)])^2 && (2.42) \\
&\quad + 2(h_{\mathcal{D}}(x) - E_{\bar{\mathcal{D}}}[h_{\bar{\mathcal{D}}}(x)])(E_{\bar{\mathcal{D}}}[h_{\bar{\mathcal{D}}}(x)] - E[Y|x]) \\
&\quad + (E_{\bar{\mathcal{D}}}[h_{\bar{\mathcal{D}}}(x)] - E[Y|x])^2\Big] && (2.43) \\
&= E_{\mathcal{D}}[(h_{\mathcal{D}}(x) - E_{\bar{\mathcal{D}}}[h_{\bar{\mathcal{D}}}(x)])^2] && (2.44) \\
&\quad + (E_{\bar{\mathcal{D}}}[h_{\bar{\mathcal{D}}}(x)] - E[Y|x])^2 && (2.45) \\
&= \text{variance}(x) + \text{ bias squared } (x) && (2.46)
\end{aligned}
$$

Bias/Variance Breakdown

- We further analyze only this first term, and take ensemble averages over $\mathcal{D}$, the random data sample. $h_{\mathcal{D}}(x)$ is a (random) learnt model from (random variable) data set $\mathcal{D}$. For a given x, we have

$$
\begin{aligned}
E_{\mathcal{D}}[(h_{\mathcal{D}}(x) - E[Y|x])^2] &= E_{\mathcal{D}}[(h_{\mathcal{D}}(x) - E_{\bar{\mathcal{D}}}[h_{\bar{\mathcal{D}}}(x)] + E_{\bar{\mathcal{D}}}[h_{\bar{\mathcal{D}}}(x)] - E[Y|x])^2] \\
&= E_{\mathcal{D}}\Big[(h_{\mathcal{D}}(x) - E_{\bar{\mathcal{D}}}[h_{\bar{\mathcal{D}}}(x)])^2 && (2.42) \\
&\quad + 2(h_{\mathcal{D}}(x) - E_{\bar{\mathcal{D}}}[h_{\bar{\mathcal{D}}}(x)])(E_{\bar{\mathcal{D}}}[h_{\bar{\mathcal{D}}}(x)] - E[Y|x]) \\
&\quad + (E_{\bar{\mathcal{D}}}[h_{\bar{\mathcal{D}}}(x)] - E[Y|x])^2\Big] && (2.43) \\
&= E_{\mathcal{D}}[(h_{\mathcal{D}}(x) - E_{\bar{\mathcal{D}}}[h_{\bar{\mathcal{D}}}(x)])^2] && (2.44) \\
&\quad + (E_{\bar{\mathcal{D}}}[h_{\bar{\mathcal{D}}}(x)] - E[Y|x])^2 && (2.45) \\
&= \text{variance}(x) + \text{ bias squared } (x) && (2.46)
\end{aligned}
$$

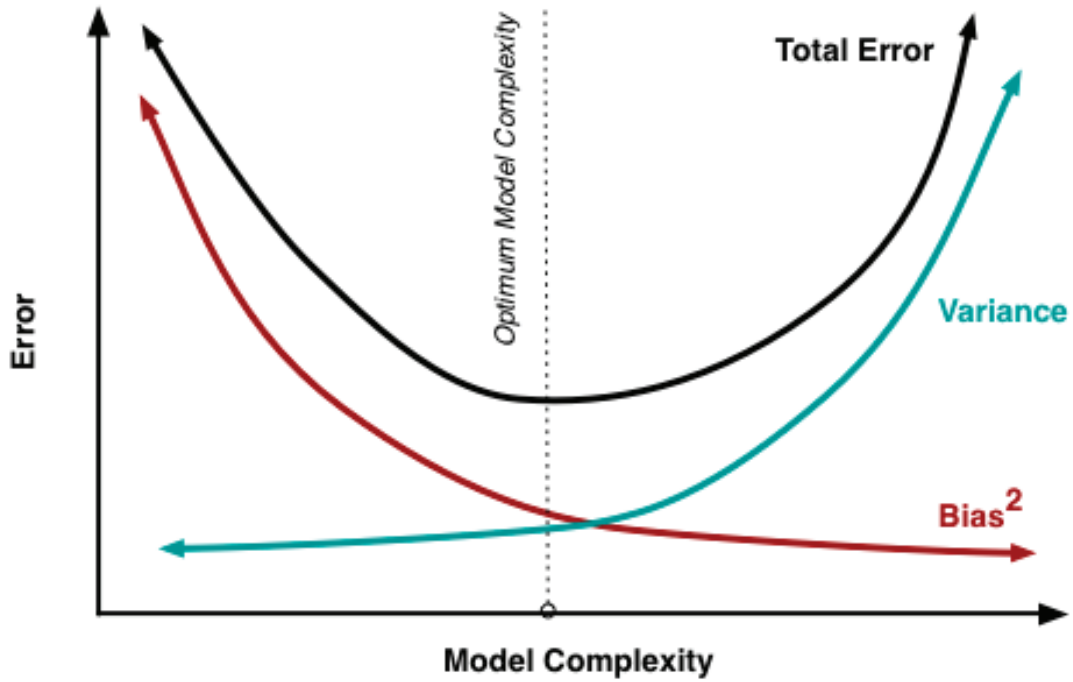
- We then take $E_{p(x)}[\cdot]$ to get overall bias and variance over all x.

Error (MSE) and Bias/Variance Breakdown

$$\text{MSE} = E_{p(x)}[E_{\mathcal{D}}[(h_{\mathcal{D}}(x) - E[Y|x])^2]] \tag{2.47}$$

$$= E_{p(x)}[E_{\mathcal{D}}[(h_{\mathcal{D}}(x) - E_{\bar{\mathcal{D}}}[h_{\bar{\mathcal{D}}}(x)])^2]] + E_{p(x)}[(E_{\bar{\mathcal{D}}}[h_{\bar{\mathcal{D}}}(x)] - E[Y|x])^2]$$

$$= \text{variance} + \text{ bias squared} \tag{2.48}$$



Bias/Variance, Unbiased estimator

- To be an unbiased estimation procedure means that $E_{\mathcal{D}}[h_{\mathcal{D}}(x)] = E[Y|x]$, or that the bias is zero.

Bias/Variance, Unbiased estimator

- To be an unbiased estimation procedure means that $E_{\mathcal{D}}[h_{\mathcal{D}}(x)] = E[Y|x]$, or that the bias is zero.
- We do sometimes have zero bias and variance that depends on the size of the data.

Bias/Variance for Linear Least Squares

- Linear model with noise $y = h_{\theta^*}(x) + \epsilon = \theta^{*\mathsf{T}} x + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

Bias/Variance for Linear Least Squares

- Linear model with noise $y = h_{\theta^*}(x) + \epsilon = \theta^{*\mathsf{T}} x + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Suppose this is true generative process for some θ^*

Bias/Variance for Linear Least Squares

- Linear model with noise $y = h_{\theta^*}(x) + \epsilon = \theta^{*\mathsf{T}} x + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Suppose this is true generative process for some θ^*
- Given training data $\mathcal{D}$ and corresponding $n \times m$ design matrix X and length-n column vector $\vec{y}$, we have relationship $\vec{y} = X\theta + \vec{\epsilon}$, $\vec{\epsilon}$ is length-n vector of Gaussians, $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$.

Bias/Variance for Linear Least Squares

- Linear model with noise $y = h_{\theta^*}(x) + \epsilon = \theta^{*\intercal} x + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Suppose this is true generative process for some θ^*
- Given training data $\mathcal{D}$ and corresponding $n \times m$ design matrix X and length-n column vector $\vec{y}$, we have relationship $\vec{y} = X\theta + \vec{\epsilon}$, $\vec{\epsilon}$ is length-n vector of Gaussians, $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$.
- MLE parameter estimate $\tilde{\theta} = (X^\intercal X)^{-1} X^\intercal \vec{y} = \theta^* + (X^\intercal X)^{-1} X^\intercal \vec{\epsilon}$, a noisy (r.v.) version of θ^*.

Bias/Variance for Linear Least Squares

- Linear model with noise $y = h_{\theta^*}(x) + \epsilon = \theta^{*\intercal} x + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Suppose this is true generative process for some θ^*
- Given training data $\mathcal{D}$ and corresponding $n \times m$ design matrix X and length-n column vector $\vec{y}$, we have relationship $\vec{y} = X\theta + \vec{\epsilon}$, $\vec{\epsilon}$ is length-n vector of Gaussians, $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$.
- MLE parameter estimate $\tilde{\theta} = (X^\intercal X)^{-1} X^\intercal \vec{y} = \theta^* + (X^\intercal X)^{-1} X^\intercal \vec{\epsilon}$, a noisy (r.v.) version of θ^*.
- Best estimate $E[Y|x] = \theta^{*\intercal} x$

Bias/Variance for Linear Least Squares

- Linear model with noise $y = h_{\theta^*}(x) + \epsilon = \theta^{*\intercal} x + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Suppose this is true generative process for some θ^*
- Given training data $\mathcal{D}$ and corresponding $n \times m$ design matrix X and length-n column vector $\vec{y}$, we have relationship $\vec{y} = X\theta + \vec{\epsilon}$, $\vec{\epsilon}$ is length-n vector of Gaussians, $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$.
- MLE parameter estimate $\tilde{\theta} = (X^\intercal X)^{-1} X^\intercal \vec{y} = \theta^* + (X^\intercal X)^{-1} X^\intercal \vec{\epsilon}$, a noisy (r.v.) version of θ^*.
- Best estimate $E[Y|x] = \theta^{*\intercal} x$
- Recall error, which has only two terms, the model error (which can be broken into the bias squared and variance) and the inherent error.

$$\text{error}(h) = E_{p(x,y)}[(h(x) - E[Y|x])^2] + E_{p(x,y)}[(E[Y|x] - y)^2] \tag{2.49}$$

Bias/Variance for Linear Least Squares

- Linear model with noise $y = h_{\theta^*}(x) + \epsilon = \theta^{*\intercal} x + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Suppose this is true generative process for some θ^*
- Given training data $\mathcal{D}$ and corresponding $n \times m$ design matrix X and length-n column vector $\vec{y}$, we have relationship $\vec{y} = X\theta + \vec{\epsilon}$, $\vec{\epsilon}$ is length-n vector of Gaussians, $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$.
- MLE parameter estimate $\tilde{\theta} = (X^\intercal X)^{-1} X^\intercal \vec{y} = \theta^* + (X^\intercal X)^{-1} X^\intercal \vec{\epsilon}$, a noisy (r.v.) version of θ^*.
- Best estimate $E[Y|x] = \theta^{*\intercal} x$
- Recall error, which has only two terms, the model error (which can be broken into the bias squared and variance) and the inherent error.

$$\text{error}(h) = E_{p(x,y)}[(h(x) - E[Y|x])^2] + E_{p(x,y)}[(E[Y|x] - y)^2] \tag{2.49}$$

- In current case, inherent error is:

$$E_{p(x,y)}[(E[Y|x] - y)^2] = \sigma^2 \tag{2.50}$$

Bias/Variance for Linear Least Squares

- Random model $h_{\tilde{\theta}}$ based on random sample $\mathcal{D}$, so $E_{\mathcal{D}}[h_{\tilde{\theta}}] = E_{\mathcal{D}}[h_{\theta^* + (X^\intercal X)^{-1} X^\intercal \vec{\epsilon}}] = h_{\theta^*}$ since $E_{\mathcal{D}}[\epsilon] = 0$.

Bias/Variance for Linear Least Squares

- Random model $h_{\tilde{\theta}}$ based on random sample $\mathcal{D}$, so $E_{\mathcal{D}}[h_{\tilde{\theta}}] = E_{\mathcal{D}}[h_{\theta^* + (X^\intercal X)^{-1} X^\intercal \vec{\epsilon}}] = h_{\theta^*}$ since $E_{\mathcal{D}}[\epsilon] = 0$.
- Thus, bias squared at x:

$$(E_{\bar{\mathcal{D}}}[h_{\bar{\mathcal{D}}}(x)] - E[Y|x])^2 = 0 \tag{2.51}$$

Bias/Variance for Linear Least Squares

- Random model $h_{\tilde{\theta}}$ based on random sample $\mathcal{D}$, so $E_{\mathcal{D}}[h_{\tilde{\theta}}] = E_{\mathcal{D}}[h_{\theta^* + (X^\intercal X)^{-1} X^\intercal \vec{\epsilon}}] = h_{\theta^*}$ since $E_{\mathcal{D}}[\epsilon] = 0$.
- Thus, bias squared at x:

$$(E_{\bar{\mathcal{D}}}[h_{\bar{\mathcal{D}}}(x)] - E[Y|x])^2 = 0 \tag{2.51}$$

- Variance increases with m and decreases with n (sample size)

$$E_{\mathcal{D}}[(h_{\mathcal{D}}(x) - E_{\bar{\mathcal{D}}}[h_{\bar{\mathcal{D}}}(x)])^2] = \frac{\sigma^2 m}{n} \tag{2.52}$$

Bias/Variance for Linear Least Squares

- Random model $h_{\tilde{\theta}}$ based on random sample $\mathcal{D}$, so $E_{\mathcal{D}}[h_{\tilde{\theta}}] = E_{\mathcal{D}}[h_{\theta^* + (X^\intercal X)^{-1} X^\intercal \vec{\epsilon}}] = h_{\theta^*}$ since $E_{\mathcal{D}}[\epsilon] = 0$.
- Thus, bias squared at x:

$$(E_{\bar{\mathcal{D}}}[h_{\bar{\mathcal{D}}}(x)] - E[Y|x])^2 = 0 \tag{2.51}$$

- Variance increases with m and decreases with n (sample size)

$$E_{\mathcal{D}}[(h_{\mathcal{D}}(x) - E_{\bar{\mathcal{D}}}[h_{\bar{\mathcal{D}}}(x)])^2] = \frac{\sigma^2 m}{n} \tag{2.52}$$

- A famous theorem (Gauss-Markov Theorem) states that among all unbiased estimators, the linear least squares (LLS) estimator has the smallest variance and hence has the smallest smallest (mean squared) error of all unbiased linear estimators! I.e., $\text{Var}(h_{\theta_{\text{LLS}}}(x)) \leq \text{Var}(h_{\theta_{\text{any unbiased}}}(x))$.