Nature, May, 28, 2015

REVIEW

doi:10.1038/nature14539

Deep learning

Yann LeCun[1,2], Yoshua Bengio[3] & Geoffrey Hinton[4,5]

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have brought about breakthroughs in processing images, video, speech and audio, whereas recurrent nets have shone light on sequential data such as text and speech.

Machine-learning technology powers many aspects of modern society: from web searches to content filtering on social networks to recommendations on e-commerce websites, and it is increasingly present in consumer products such as cameras and smartphones. Machine-learning systems are used to identify objects in images, transcribe speech into text, match news items, posts or products with users' interests, and select relevant results of search. Increasingly, these applications make use of a class of techniques called deep learning.

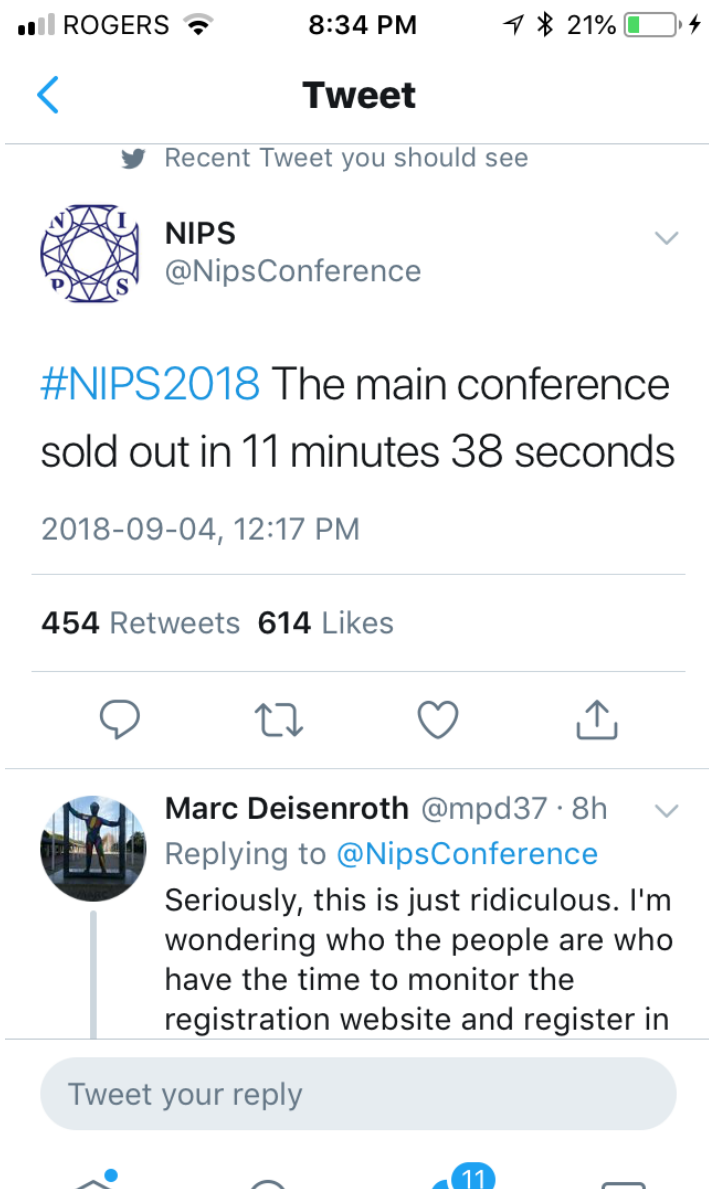
Conventional machine-learning techniques were limited in their ability to process natural data in their raw form. For decades, constructing a pattern-recognition or machine-learning system required careful engineering and considerable domain expertise to design a feature extractor that transformed the raw data (such as the pixel values of an image) into a suitable internal representation or feature vector from which the learning subsystem, often a classifier, could detect or classify patterns in the input.

Representation learning is a set of methods that allows a machine to be fed with raw data and to automatically discover the representations needed for detection or classification. Deep-learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. With the composition of enough such transformations, very complex functions can be learned. For classification tasks, higher layers of representation amplify aspects of the input that are important for discrimination and suppress irrelevant variations. An image, for example, comes in the form of an array of pixel values, and the learned features in the first layer of representation typically represent the presence or absence of edges at particular orientations and locations in the image. The second layer typically detects motifs by spotting particular arrangements of edges, regardless of small variations in the edge positions. The third layer may assemble motifs into larger combinations that correspond to parts of familiar objects, and subsequent layers would detect objects as combinations of these parts. The key aspect of deep learning is that these layers of features are not designed by human engineers: they are learned from data using a general-purpose learning procedure.

Deep learning is making major advances in solving problems that have resisted the best attempts of the artificial intelligence community for many years. It has turned out to be very good at discovering intricate structures in high-dimensional data and is therefore applicable to many domains of science, business and government. In addition to beating records in image recognition[1–4] and speech recognition[5–7], it has beaten other machine-learning techniques at predicting the activity of potential drug molecules[8], analysing particle accelerator data[9,10], reconstructing brain circuits[11], and predicting the effects of mutations in non-coding DNA on gene expression and disease[12,13]. Perhaps more surprisingly, deep learning has produced extremely promising results for various tasks in natural language understanding[14], particularly topic classification, sentiment analysis, question answering[15] and language translation[16,17].

Ke Jie vs AlphaGo: **0:3**, May, 27, 2017

Twitter, September 4, 2018



ChatGPT, November 30, 2022

Default (GPT-3.5)

Alex's father has three sons. He named his first son Jason and second son Jack. What's the name of his third son?

The name of the third son is Alex, as mentioned at the beginning of the question: "Alex's father has three sons."

Deep Learning

Deep Learning

$\neq$

Machine Learning

Deep Learning

$\neq$

Machine Learning

$\neq$

Artificial Intelligence

CSI5138
Introduction to Deep Learning and Reinforcement Learning
Module 01: Machine Learning Basics

Yongyi Mao

University of Ottawa

2023/09/07

Machine Learning: A Definition

Quote (Tom M. Mitchell, 97)

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

- A program that predicts stock prices from the market history
- A program that approves/declines a loan application based on the applicant's information
- A program that discovers subclasses of lung cancers from a collection of X-ray images
- A program that plays chess
- ...

Note

The "task" is defined by the input/output specification of the program.

Main Classes of ML Problems

- Supervised Learning
  - Given $\mathcal{D} := \{(x_i, y_i) : i = 1, 2, \ldots, N\}$, develop a program that predicts Y from X, or finds how Y depends on X
  - Examples: Regression, Classification, ...
- Unsupervised Learning
  - Given $\mathcal{D} := \{x_i : i = 1, 2, \ldots, N\}$, develop a program that finds the structure in X, or generates an (new) X that conforms to the structure.
  - Examples: Clustering, Density Estimation, generating realistic images, ...
- Reinforcement Learning
  - Suppose that there is an "environment" which can interact with an agent by changing the "state" and generating a reward for the agent. Develop an agent program that maximizes some accumulated reward it receives.
  - Example: Chess-playing program ...

Concept of Model

- A *model* is a *restricted* family $\mathcal{H}$ of hypotheses
  - about how Y depends on X (in supervised learning)
    - How many hypotheses are there for binary-classifying a 10×10 black-and-white images? Create a model for this!
  - about the structure of X (in unsupervised learning)
  - about the policy of the agent and/or the behaviour of environment (in reinforcement learning)
- Ambiguity:
  - model $= \mathcal{H}$?
  - model $= \mathcal{H}$ + algorithm finding $h^* \in \mathcal{H}$?
- Parametric models: models having a fixed number of parameters, independent of sample size
- Non-Parametric models: the number of parameters increases with sample size
- We consider primarily parametric models in this course

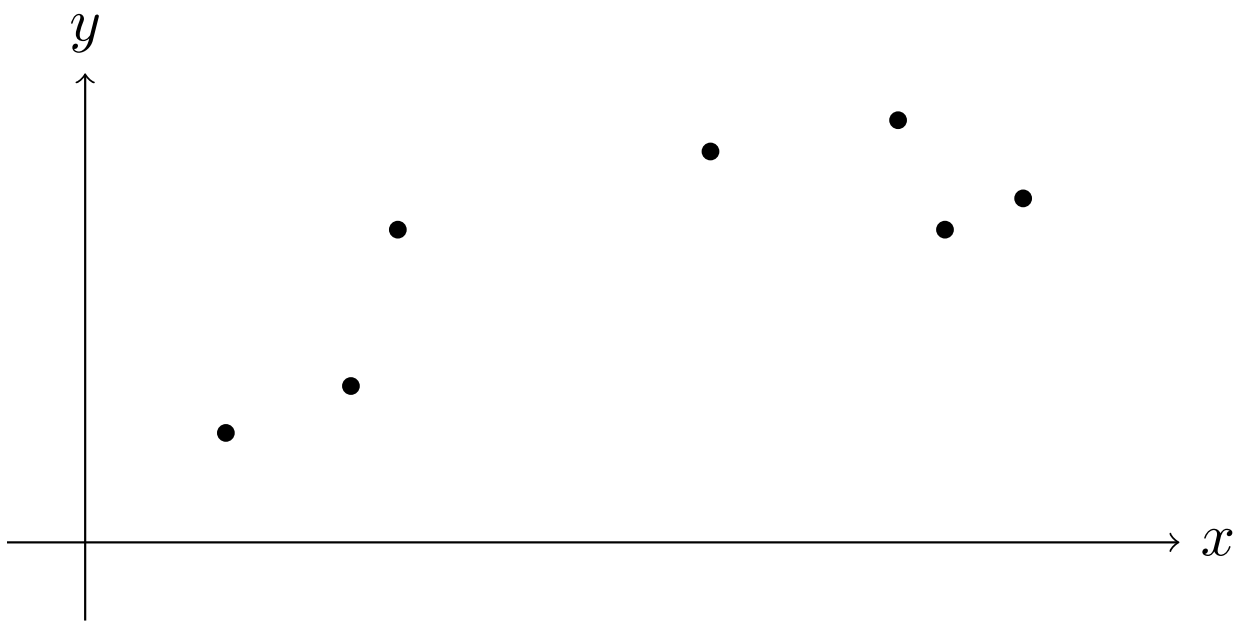
Loss Functions

- A model is usually characterized by a loss function $\mathcal{L}(\theta)$ over the space Θ of model parameters θ.
  - Each $\theta \in \Theta$ corresponds to an hypothesis in $\mathcal{H}$.
    - This correspondence can be one-to-one, namely that each hypothesis corresponds to only one θ.
    - Sometimes multiple θ, usually continuously many, may all correspond to a single hypothesis in $\mathcal{H}$.
  - Loss measures some notion of error.
  - Loss should be chosen "correlated" to the performance measure P. The higher the correlation, the better.
- The problem of learning then reduces to finding

$$\widehat{\theta} := \arg\min_{\theta \in \Theta} \mathcal{L}(\theta)$$

Example: Regression

- X takes value in $\mathbb{R}^K$
- Y takes value in $\mathbb{R}$
- Given $\mathcal{D} := \{(x_i, y_i) : i = 1, 2, \ldots, N\}$
- Predict Y for an arbitrary X



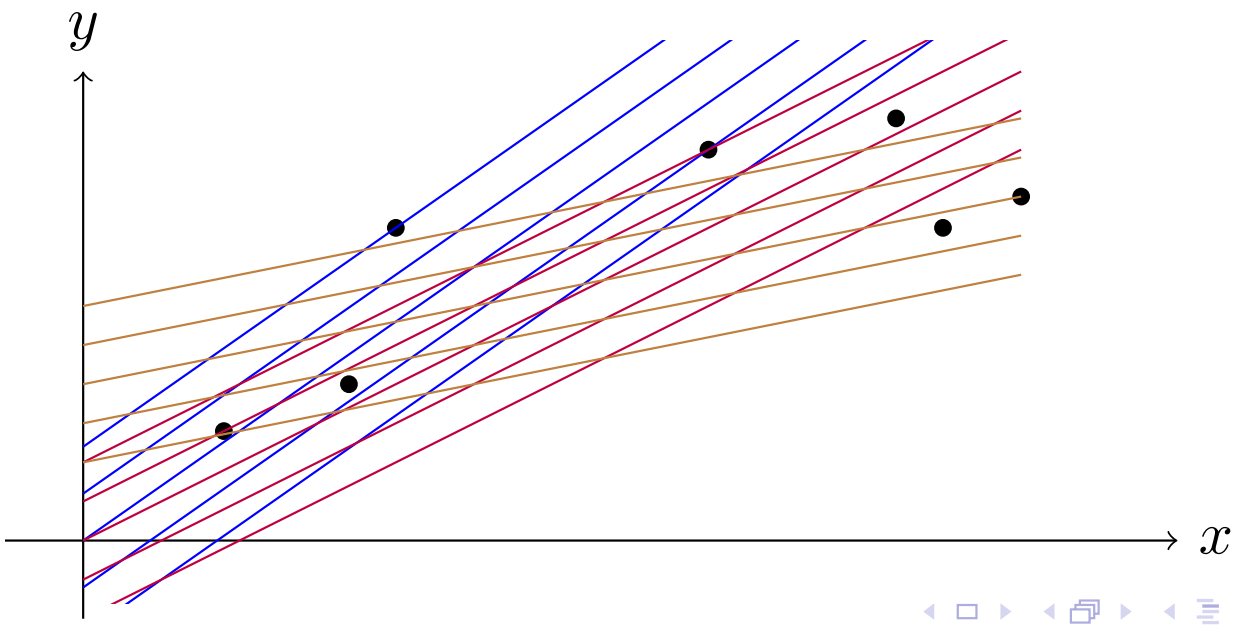
- The family $\mathcal{H}$ of hypotheses: Y depends on X according to a function $Y \approx f(X;\theta)$.
- Linear Regression Model:

$$Y \approx \theta^T X$$

  - More precisely, $Y \approx \theta^T X + b$, which can be taken as

$$Y \approx \begin{bmatrix} \theta \\ b \end{bmatrix}^T \begin{bmatrix} X \\ 1 \end{bmatrix}$$

    but we will absorb b into θ and 1 into X.



- Choose "mean square error (MSE)" loss:

$$\mathcal{L}(\theta) := \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i;\theta))^2 = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \theta^T x_i\right)^2$$

- Minimizing the MSE loss is equivalent to maximizing the data likelihood under a Gaussian probability model.

Note (Maximum Likelihood Principle)

Under the maximum likelihood (ML) principle, the best parameter of a given probability model is declared as the one that maximizes the data likelihood under the model.

- What do I mean by "principle"?

- A Gaussian model for regression
  - (X, Y) is a pair of random variables, in which $p_{Y|X}$ is specified via

$$Y = f(X; \theta) + Z$$

    where Z is scalar Gaussian random variable with zero mean and variance σ^2 and Z is independent of X.
  - The data $\mathcal{D}$ is drawn i.i.d. from p_{XY}.
- Then the ML estimate $\widehat{\theta}_{\text{ML}}$ of θ is $\widehat{\theta} := \arg\min_\theta \mathcal{L}(\theta)$.
  Proof:

$$\begin{aligned}
\widehat{\theta}_{\text{ML}} &:= \arg\max_\theta \log p(\mathcal{D}|\theta) \\
&= \arg\max_\theta \log \prod_{i=1}^N p_{Y|X}(y_i|x_i;\theta) \\
&= \arg\max_\theta \sum_{i=1}^N \log p_{Y|X}(y_i|x_i;\theta) \\
&= \arg\max_\theta \sum_{i=1}^N -\frac{(y_i - f(x_i;\theta))^2}{2\sigma^2} \\
&= \arg\min_\theta \mathcal{L}(\theta) = \widehat{\theta}
\end{aligned}$$

- Minimize $\mathcal{L}(\theta)$ in closed form
  - Let $\mathbf{X} := \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{bmatrix}$ and $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}^T$.
  - Minimizing $\mathcal{L}(\theta)$ is minimizing $\|\mathbf{y} - \mathbf{X}\theta\|^2$
  - $\mathbf{X}\theta$ lives in the column space, SPAN($\mathbf{X}$), of $\mathbf{X}$ for any θ.
  - $\mathbf{X}\widehat{\theta} := \widehat{\mathbf{y}}$ must satisfy that $\mathbf{y} - \widehat{\mathbf{y}}$ is orthogonal to SPAN($\mathbf{X}$).
  - Then $\mathbf{X}^T(\mathbf{y} - \widehat{\mathbf{y}}) = 0$.
  - $\Rightarrow \mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{X}\widehat{\theta} = 0$, or, $\mathbf{X}^T\mathbf{y} = \mathbf{X}^T\mathbf{X}\widehat{\theta}$. Then we have

$$\widehat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

- Minimize $\mathcal{L}(\theta)$ by Gradient Descent (GD)
  1. Start from a random θ
  2. Update θ according to

$$\begin{aligned} \theta^{\text{new}} &:= \theta^{\text{old}} - \lambda \frac{d\mathcal{L}}{d\theta}(\theta^{\text{old}}) \\ &= \theta^{\text{old}} + \lambda \frac{1}{N} \sum_{i=1}^{N} 2(y_i - {\theta^{\text{old}}}^T x_i) x_i \end{aligned}$$

     where λ is a small step size, called the learning rate
  3. Repeat (2)

- Minimize $\mathcal{L}(\theta)$ by Stochastic Gradient Descent (SGD): replace gradient by its stochastic estimation using a single data point
  1. Start from a random θ
  2. Draw a random $(x, y) \in \mathcal{D}$ and update θ according to

$$\theta^{\text{new}} \quad := \quad \theta^{\text{old}} + \lambda \cdot 2(y - \theta^{\text{old}\,T} x)x$$

  3. Repeat (2)

- Minimize $\mathcal{L}(\theta)$ by Mini-Batched SGD:
  replace gradient by its stochastic estimation using a mini-batch of data points
  1. Start from a random θ
  2. Draw a random mini-batch $\mathcal{B} \subset \mathcal{D}$ and update θ according to

$$\theta^{\text{new}} := \theta^{\text{old}} + \lambda \cdot \frac{1}{|\mathcal{B}|} \sum_{(x,y)\in\mathcal{B}} 2(y - \theta^{\text{old}^T} x)x$$

  3. Repeat (2)
- In most literature, "SGD"="Mini-Batched SGD", and we will adopt this convention hereafter.

Some Remarks

- When the gradient of $\mathcal{L}(\theta)$ decomposes into a sum (or average) over data points, SGD and mini-batched SGD are in general applicable.
- Smaller learning rate
  - slower convergence, i.e., longer training time
  - the final loss value closer to a local optimum
  - higher chance of getting stuck at nearby local optima
- Adaptive learning rate?
- Larger batch size
  - less noisy gradient (relative to GD gradient), smoother dropping of training loss
  - better utilizing parallel-computing resources (computing within a batch can be paralleled)
  - more computation per update
- Some argue that SGD has an implicit bias in helping generalization, where learning rate and batch size matter.

What have we done?

1. (Modelling) We chose a family of hypotheses
   - Should we choose a different function class for $f(x;\theta)$, say, as a higher-order polynomials? Why and why not? – To discuss
2. (Optimization formulation) We defined an optimization problem, taking faith in some "good principle".
3. (Solution) We solved the optimization problem.

Note

When using the learned model to predict Y for a new X, we (under the ML/frequentist principle) declare

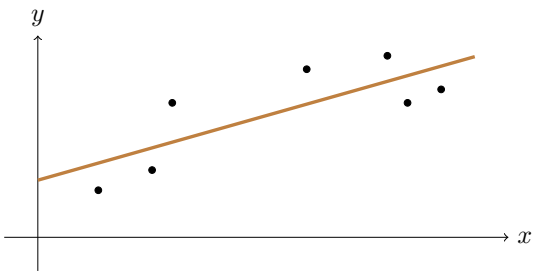
$$Y = f(X;\widehat{\theta})$$

Evaluating a Learned Model

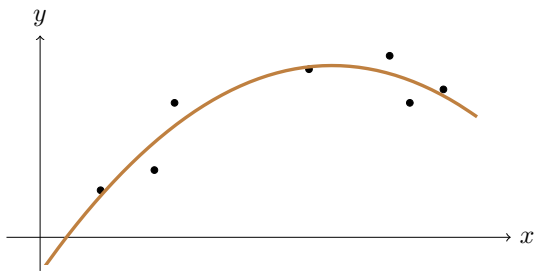
- In-sample performance/"training performance", empirical risk: performance/risk on $\mathcal{D}$, the data we used to trained the model
- Out-of-sample performance/"testing performance", true /population risk:
  performance on unseen examples following the same statistics as $\mathcal{D}$
- Our ultimate interest is the out-of-sample performance of the learned model.
- performance metrics:
  - problem-specific metrics: precision/recall/F1, accuracy, rank, BLEU, ..., subjective metrics
  - the designed loss/error function
- Standard practice: cross-validation
  divide the available data into training, validation and testing sets
  - Use the training set as $\mathcal{D}$ to train the model
  - Use the validation set to select hyper-parameters, to stop SGD iterations, to check if the model performs well enough, etc.
  - Use the testing set to evaluate model performance

Fitting and Generalization

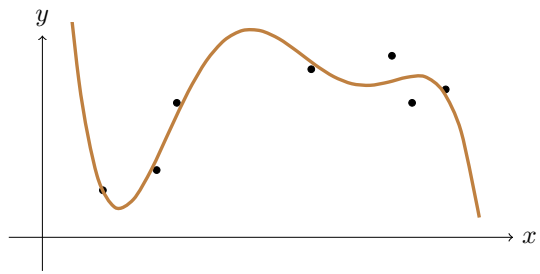
- Should we choose a complex model or a simple model?



degree-1 polynomial

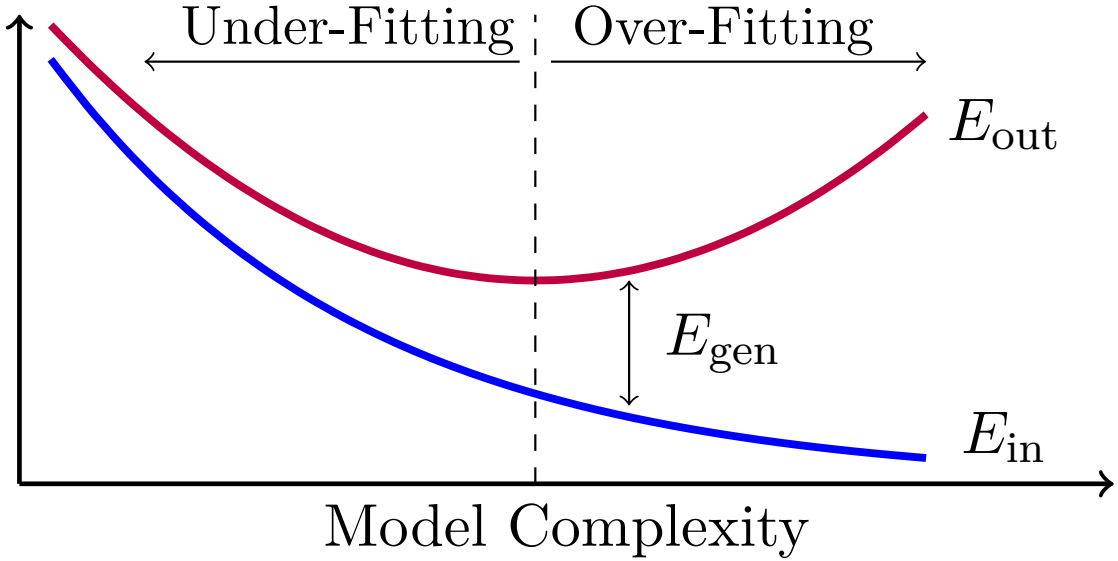


degree-2 polynomial



degree-5 polynomial

- In-Sample Error ("training error") E_{in}
- Out-of-Sample Error ("testing error") E_{out}
- Ideally, we wish to have $E_{\text{in}} \approx 0$ and $E_{\text{out}} \approx E_{\text{in}}$
- Generalization Gap $E_{\text{gen}} := E_{\text{out}} - E_{\text{in}}$



- E_{in} decreases with increasing model complexity.
- There exists an optimal complexity, marking the division of under-fitting and over-fitting models
  - In the under-fitting regime, E_{out} decreases with model complexity
  - In the over-fitting regime, E_{out} increases with model complexity
- E_{gen} increases with model complexity

Bias and Variance

- Consider the regression problem in which

$$Y = F(X)$$

- F is unknown.
- Mean square error is chosen as performance metric.
- For a given model (i.e, function class) $\mathcal{H}$, suppose that a learning algorithm returns $\widehat{f}^{(\mathcal{D})} \in \mathcal{H}$.
- Then define an expected version of E_{out} as

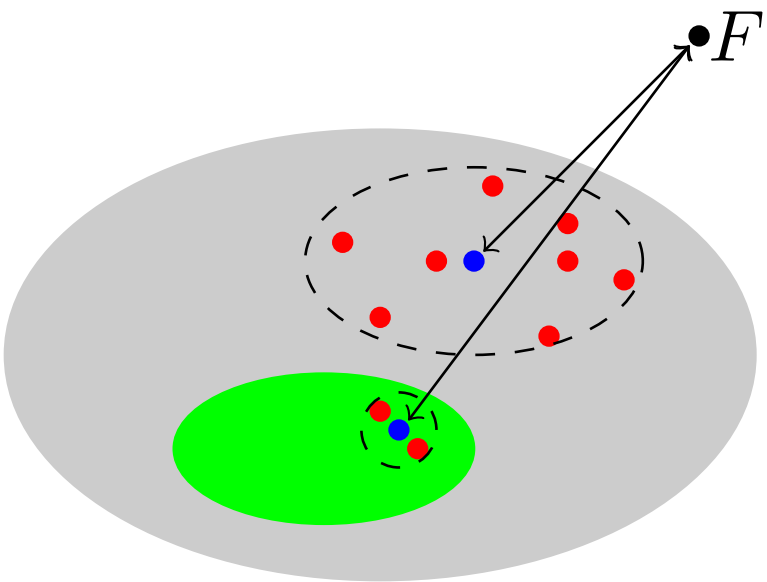
$$E_{\text{out}} = \mathbb{E}_{\mathcal{D}}\mathbb{E}_X \left(\widehat{f}^{(\mathcal{D})}(X) - F(X) \right)^2$$

- We will show that E_{out} decomposes into "bias" and "variance".

- Let function $\overline{f}$ be defined by $\overline{f}(x) := \mathbb{E}_{\mathcal{D}} \widehat{f}^{(\mathcal{D})}(x)$
  - $\overline{f}$ may not live in $\mathcal{H}$.

$$
\begin{aligned}
E_{\text{out}} &= \mathbb{E}_{\mathcal{D}}\mathbb{E}_X \left(\widehat{f}^{(\mathcal{D})}(X) - F(X)\right)^2 \\
&= \mathbb{E}_X\mathbb{E}_{\mathcal{D}} \left(\widehat{f}^{(\mathcal{D})}(X) - F(X)\right)^2 \\
&= \mathbb{E}_X\mathbb{E}_{\mathcal{D}} \left(\widehat{f}^{(\mathcal{D})}(X) - \overline{f}(X) + \overline{f}(X) - F(X)\right)^2 \\
&= \mathbb{E}_X\mathbb{E}_{\mathcal{D}} \left[\left(\widehat{f}^{(\mathcal{D})}(X) - \overline{f}(X)\right)^2 + \left(\overline{f}(X) - F(X)\right)^2 \right. \\
&\quad \left. + 2\left(\widehat{f}^{(\mathcal{D})}(X) - \overline{f}(X)\right)\left(\overline{f}(X) - F(X)\right)\right] \\
&= \mathbb{E}_X\mathbb{E}_{\mathcal{D}} \left[\left(\widehat{f}^{(\mathcal{D})}(X) - \overline{f}(X)\right)^2 + \left(\overline{f}(X) - F(X)\right)^2\right] \\
&= \mathbb{E}_X\mathbb{E}_{\mathcal{D}} \left(\widehat{f}^{(\mathcal{D})}(X) - \overline{f}(X)\right)^2 + \mathbb{E}_X \left(\overline{f}(X) - F(X)\right)^2 \\
&:= \text{variance} + \text{bias}
\end{aligned}
$$

- "Bias": the error of $\overline{f}$ relative to the ground truth F
- "Variance": the average deviation of $\widehat{f}^{(\mathcal{D})}$ relative to $\overline{f}$
- Bias-variance tradeoff
  - Low complexity model: high bias, low variance
  - High complexity model: low bias, high variance

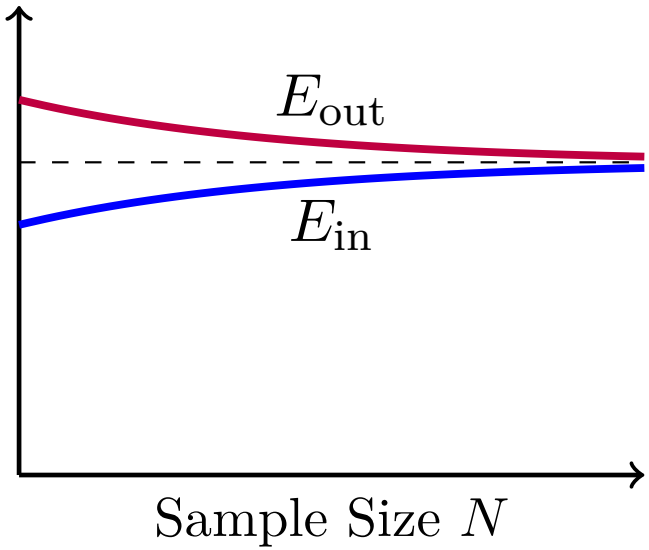


- Grey: High-Complexity Model
- Green: Low-compexity Model
- Blue dot: $\overline{f}$
- Red dot: $\widehat{f}^{(\mathcal{D})}$
- In general, the green may or may not live inside the grey; but for the polynomial regression problem, it does.
- Such a picture is quite "typical", but not always true. The "fitness" of the model's Inductive bias to the data structure also matters.

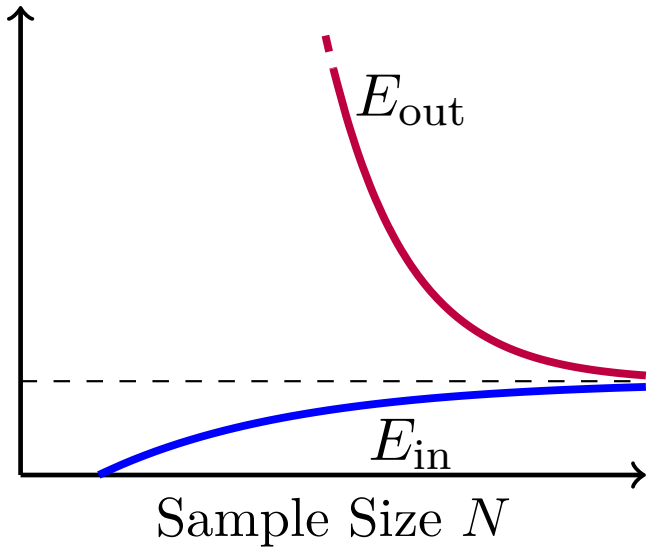
Inductive Bias

- Assumptions made in a model
  - Modelling the quality of university:
    - "the quality of a university depends on its professors"
    - "the quality of a university depends on its staff"
    - "the quality of a university depends on its infrastructure".
  - Regression:
    - "each $h \in \mathcal{H}$ is a polynomial"
    - "each $h \in \mathcal{H}$ is a sinusoidal"
  - Modelling $X = (X[1], X[2], \ldots, X[d])$
    - "$(X[1], X[2], \ldots X[d])$ are independent"
    - "$(X[1], X[2], \ldots X[d])$ form a Markov chain"
- Favoured type solutions returned from a learning algorithm
  - Run SGD with learning rate 0.1.
  - Run SGD with learning rate 0.0001.

E_{in}, E_{out}, and E_{gen} vs Sample Size N and Model Complexity

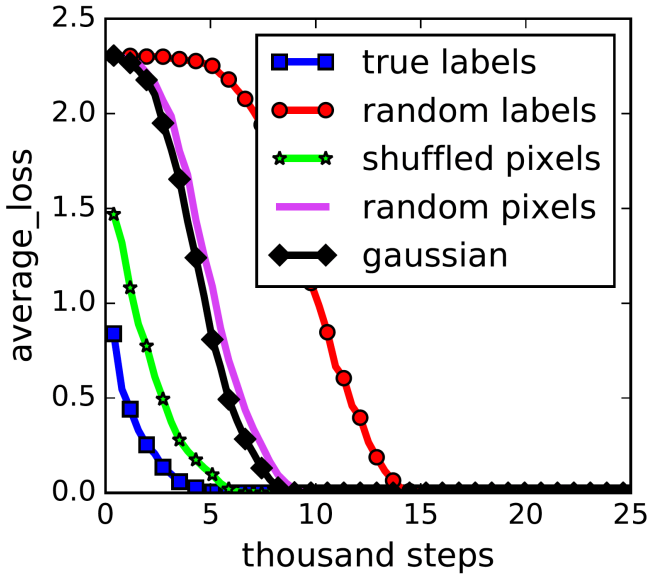


Simple Model

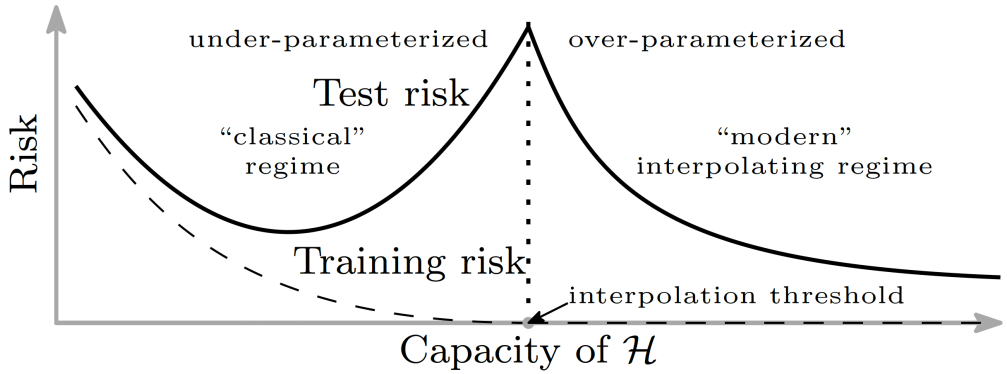


Complex Model

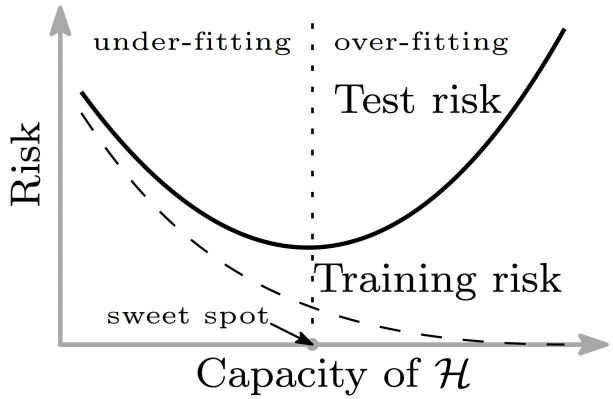
The generalization mystery of deep learning



- Zhang et al, "Understanding deep learning requires rethinking generalization ", 2017
- Classical statistical learning theory:
  "High-capacity models do not generalize, or tend to overfit."
- But deep neural nets have extremely large capacity (# of para. $>$ $|S|$) yet generalize well



- Belkin et al, "Reconciling modern machine learning practice and the bias-variance trade-off", 2019
- "double descent" contradicts the classical understanding.



Regularization

- Regularization refers to techniques that reduce over-fitting when learning with complex models.
  - "soft" reduction of model complexity
  - constraining parameter space to make the optimization in favour of simpler or smoother hypotheses
- Regularization by optimizing a regularized loss function instead:

$$\mathcal{L}_{\text{Reg}}(\theta) := \mathcal{L}(\theta) + \Omega(\theta)$$

- Some regard early stopping as a form a regularization.

Regularized Loss: $\mathcal{L}_{\text{Reg}}(\theta) := \mathcal{L}(\theta) + \Omega(\theta)$

- Underlying assumption:
  Lower magnitudes of θ correspond to simpler models or smoother hypotheses
- Most popular: L2-Regularizer (a.k.a, "weight decay"):

$$\Omega(\theta) := \lambda_{\text{Reg}} \|\theta\|_2^2$$

  where λ_{Reg} is a hyper-parameter
  - Larger λ_{Reg} make smaller $\|\theta\|_2$ preferred.
- When minimizing $\mathcal{L}(\theta)$ corresponds to an ML formulation, minimizing $\mathcal{L}_{\text{Reg}}(\theta)$ with L2-regularizer corresponds to an maximum *a posteriori* (MAP) formulation with a Gaussian prior.

Note (MAP Principle)

Suppose that θ is drawn from a distribution p_θ on Θ. The MAP estimate $\widehat{\theta}_{\text{MAP}}$ of θ under a probability model is the value of θ that maximizes the $p(\theta|\mathcal{D})$.

$$\begin{aligned}
\widehat{\theta}_{\text{MAP}} &:= \arg\max_\theta \log p(\theta|\mathcal{D}) \\
&= \arg\max_\theta \log \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} \\
&= \arg\max_\theta \left(\log p(\mathcal{D}|\theta) + \log p(\theta)\right) \\
&= \arg\min_\theta \left(-\log p(\mathcal{D}|\theta) - \log p(\theta)\right) \\
&= \arg\min_\theta \left(\mathcal{L}(\theta) - \log p(\theta)\right)
\end{aligned}$$

Note

The term $-\log p(\theta)$ serves as a regularizer $\Omega(\theta)$. When $p(\theta)$ is a spherical Gaussian, $-\log p(\theta)$ reduces to the weight-decay regularizer.

Another View of Regularization

- $\mathcal{L}_{\text{Reg}}(\theta)$ may be seen as a better proxy of E_{out} than $\mathcal{L}(\theta)$, so that minimizing $\mathcal{L}_{\text{Reg}}(\theta)$ has an effect of minimizing E_{out}.
  - $\mathcal{L}(\theta)$ is indicator of E_{in}.
  - $\Omega(\theta)$ tries to capture the generalization gap E_{gen}.
    - Recall: E_{gen} is small for simple model and large for complex model
    - Check: $\Omega(\theta)$ is small for simple model and large for complex model

Data-Dependent Regularization

- In the cases above, the term $\Omega(\cdot)$ is usually designed independent of the training data.
- More recently regularization schemes
  - Data augmentation
  - MixUp:
    - Zhang et al, "mixup: Beyond Empirical Risk Minimization", 2017
  - Adversarial training:
    - Goodfellow et al, "Explaining and Harnessing Adversarial Examples", 2015
  - Label smoothing
    - Szegedy et al, "Rethinking the Inception Architecture for Computer Vision", 2016
  - Adversarial Model Perturbation (AMP)/Sharpness Aware Minimization (SAM):
    - Zheng, et al, "Regularizing Neural Networks via Adversarial Model Perturbation", 2020
    - Foret, et al, "Sharpness-Aware Minimization for Efficiently Improving Generalization", 2020

Three Interacting Dimensions of Machine Learning

- Modelling
  - Does the hypothesis family $\mathcal{H}$ have the correct inductive bias for the learning task?
- Optimization
  - Do we have an efficient means of finding a good solution in $\mathcal{H}$ (i.e., one with small enough loss)?
- Generalization
  - Does the solution generalize?

After-Class

- Read Goodfellow text: Chapter 5 (and all chapters before it if you do not already know them)
- Homework 1