Deep Learning

Lecture 1 – Introduction

Kumar Bipin

BE, MS, PhD (MMMTU, IISc, IIIT-Hyderabad)

Robotics, Computer Vision, Deep Learning, Machine Learning, System Software

Thies, Elgharib, Tewari, Theobalt and Niessner: Neural Voice Puppetry: Audio-driven Facial Reenactment. ECCV, 2020.

Agenda

1.1 Organization

1.2 History of Deep Learning

1.3 Machine Learning Basics

1.1
Organization

Course Materials and Prerequisites

Prerequisites

Linear Algebra:

- Vectors: $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$
- Matrices: $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m \times n}$
- Operations: $\mathbf{A}^T$, $\mathbf{A}^{-1}$, $\mathrm{Tr}(\mathbf{A})$, $\det(\mathbf{A})$, $\mathbf{A} + \mathbf{B}$, $\mathbf{AB}$, $\mathbf{Ax}$, $\mathbf{x}^\top \mathbf{y}$
- Norms: $\|\mathbf{x}\|_1$, $\|\mathbf{x}\|_2$, $\|\mathbf{x}\|_\infty$, $\|\mathbf{A}\|_F$
- SVD: $\mathbf{A} = \mathbf{UDV}^\top$

Prerequisites

Probability and Information Theory:

- Probability distributions: $P(X = x)$
- Marginal/conditional: $p(x) = \int p(x, y)dy\,,\ p(x, y) = p(x|y)p(y)$
- Bayes rule: $p(x|y) = p(y|x)p(x)/p(y)$
- Conditional independence: $x \perp\!\!\!\perp y \mid z \Leftrightarrow p(x, y|z) = p(x|z)p(y|z)$
- Expectation: $\mathbb{E}_{x\sim p}\left[f(x)\right] = \int_x p(x)f(x)dx$
- Variance: $\mathrm{Var}(f(x)) = \mathbb{E}\left[(f(x) - \mathbb{E}[f(x)])^2\right]$
- Distributions: Bernoulli, Categorical, Gaussian, Laplace
- Entropy: $H(x)$, KL Divergence: $D_{KL}(p\|q)$

Thank You!

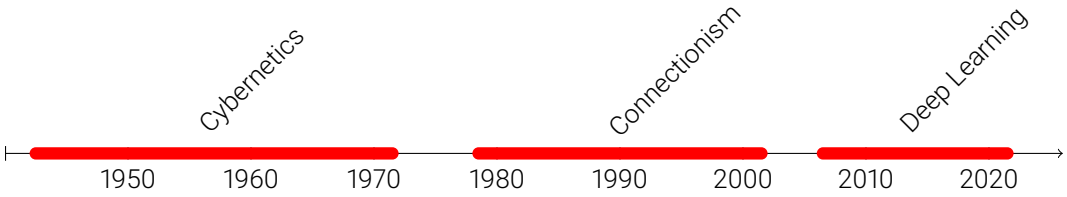
Looking forward to our discussions

1.2
History of Deep Learning

A Brief History of Deep Learning

Three waves of development:

- 1940-1970: “Cybernetics” (Golden Age)
  - Simple computational models of biological learning, simple learning rules
- 1980-2000: “Connectionism” (Dark Age)
  - Intelligent behavior through large number of simple units, Backpropagation
- 2006-now: “Deep Learning” (Revolution Age)
  - Deeper networks, larger datasets, more computation, state-of-the-art in many areas



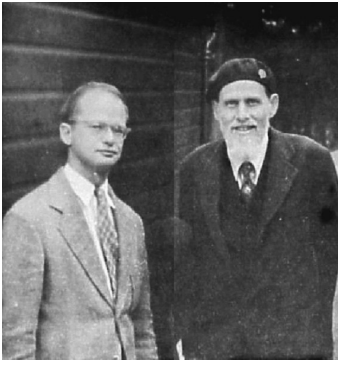
A Brief History of Deep Learning

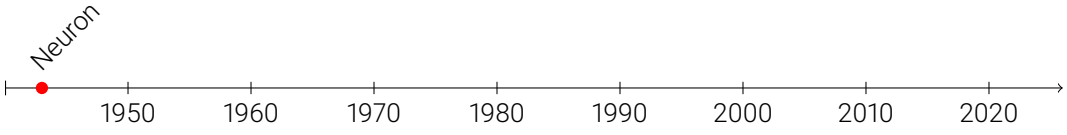
1943: McCullock and Pitts

- Early model for neural activation
- Linear threshold neuron (binary):

$$f_{\mathbf{w}}(\mathbf{x}) = \begin{cases} +1 & \text{if } \mathbf{w}^T\mathbf{x} \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

- More powerful than AND/OR gates
- But no procedure to learn weights





McCulloch and Pitts: A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 1943.

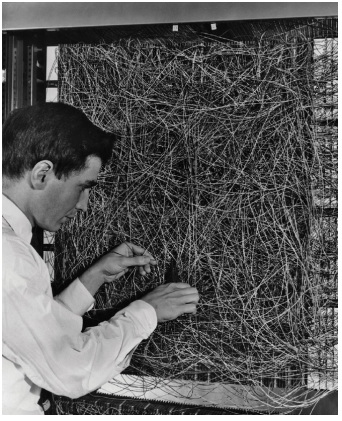
A Brief History of Deep Learning

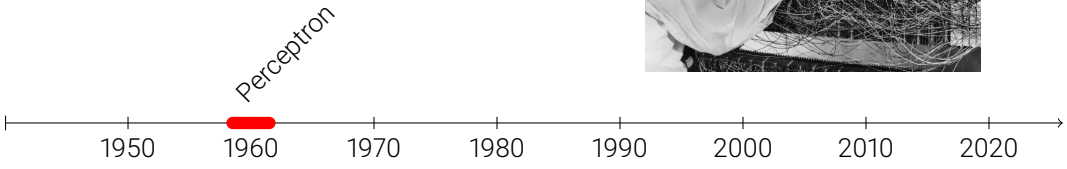
1958-1962: Rosenblatt's Perceptron

- First algorithm and implementation to train single linear threshold neuron
- Optimization of perceptron criterion:

$$\mathcal{L}(\mathbf{w}) = -\sum_{n \in \mathcal{M}} \mathbf{w}^T \mathbf{x}_n \, y_n$$

- Novikoff proved convergence



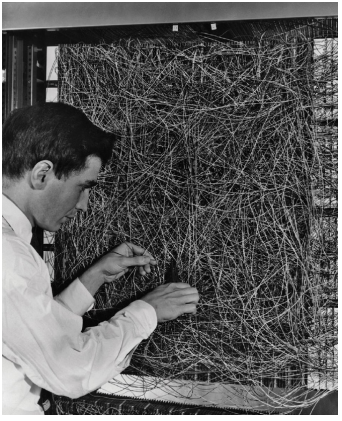


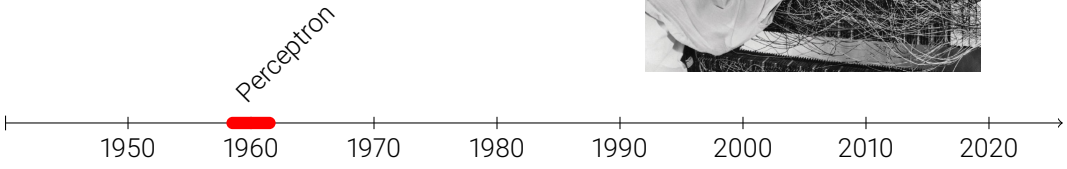
Rosenblatt: The perceptron - a probabilistic model for information storage and organization in the brain. Psychological Review, 1958.

A Brief History of Deep Learning

1958-1962: Rosenblatt's Perceptron

- First algorithm and implementation to train single linear threshold neuron
- Overhyped: Rosenblatt claimed that the perceptron will lead to computers that walk, talk, see, write, reproduce and are conscious of their existence



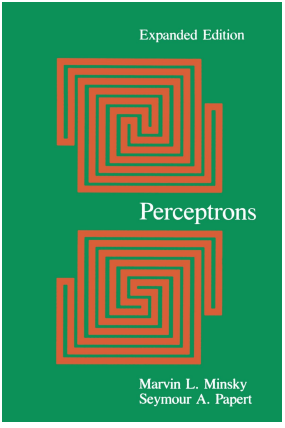


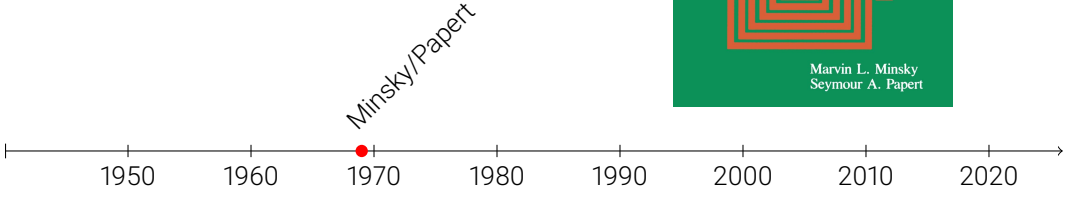
Rosenblatt: The perceptron - a probabilistic model for information storage and organization in the brain. Psychological Review, 1958.

A Brief History of Deep Learning

1969: Minsky and Papert publish book

- Several discouraging results
- Showed that single-layer perceptrons cannot solve some very simple problems (XOR problem, counting)
- Symbolic AI research dominates 70s



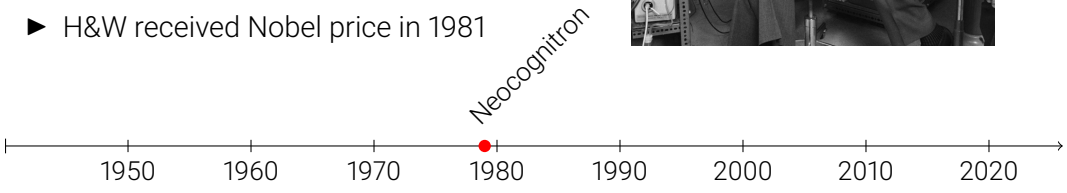


Minsky and Papert: Perceptrons: An introduction to computational geometry. MIT Press, 1969.

A Brief History of Deep Learning

1979: Fukushima's Neocognitron

- Inspired by Hubel and Wiesel experiments in the 1950s
- Study of visual cortex in cats
- Found that cells are sensitive to orientation of edges but insensitive to their position (simple vs. complex)
- H&W received Nobel price in 1981

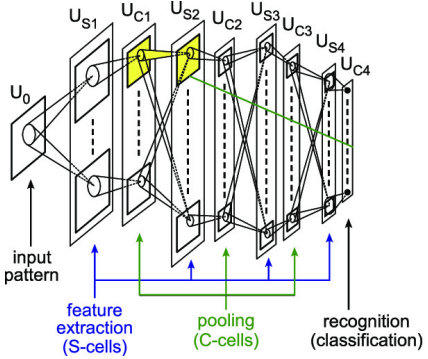


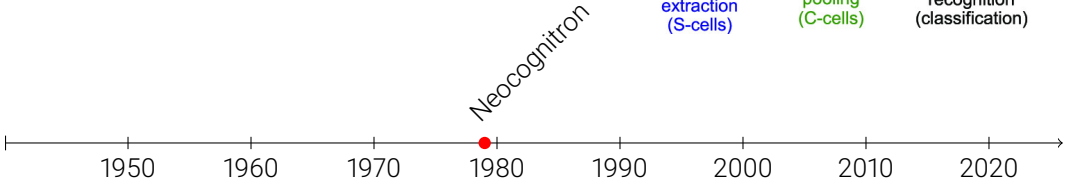
Fukushima: Neural network model for a mechanism of pattern recognition unaffected by shift in position. IECE (in Japanese), 1979.

A Brief History of Deep Learning

1979: Fukushima's Neocognitron

- Multi-layer processing to create intelligent behavior
- Simple (S) and complex (C) cells implement convolution and pooling
- Reinforcement based learning
- Inspiration for modern ConvNets



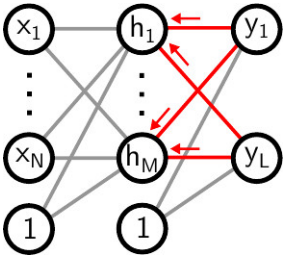


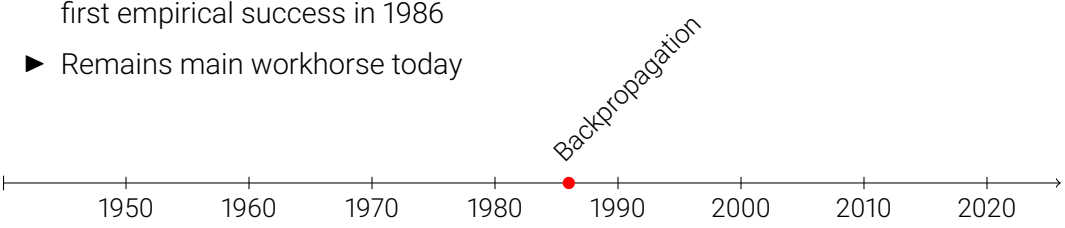
Fukushima: Neural network model for a mechanism of pattern recognition unaffected by shift in position. IECE (in Japanese), 1979.

A Brief History of Deep Learning

1986: Backpropagation Algorithm

- Efficient calculation of gradients in a deep network wrt. network weights
- Enables application of gradient based learning to deep networks
- Known since 1961, but first empirical success in 1986
- Remains main workhorse today



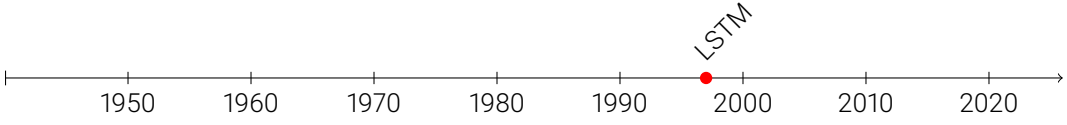


Rumelhart, Hinton and Williams: Learning representations by back-propagating errors. Nature, 1986.

A Brief History of Deep Learning

1997: Long Short-Term Memory

- In 1991, Hochreiter demonstrated the problem of vanishing/exploding gradients in his Diploma Thesis
- Led to development of long-short term memory for sequence modeling
- Uses feedback and forget/keep gate

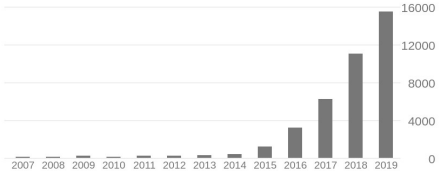


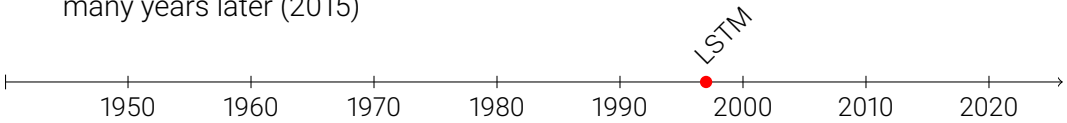
Hochreiter, Schmidhuber: Long short-term memory. Neural Computation, 1997.

A Brief History of Deep Learning

1997: Long Short-Term Memory

- In 1991, Hochreiter demonstrated the problem of vanishing/exploding gradients in his Diploma Thesis
- Led to development of long-short term memory for sequence modeling
- Uses feedback and forget/keep gate
- Revolutionized NLP (e.g. at Google) many years later (2015)



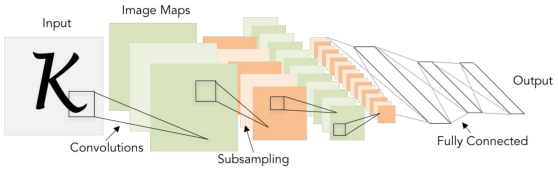


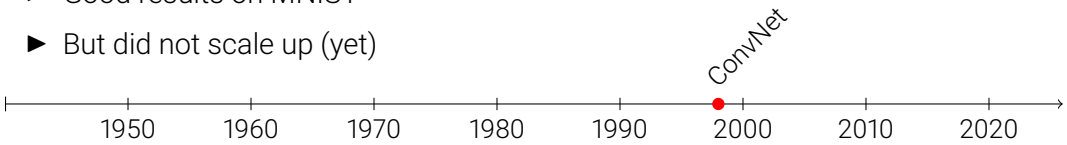
Hochreiter, Schmidhuber: Long short-term memory. Neural Computation, 1997.

A Brief History of Deep Learning

1998: Convolutional Neural Networks

- Similar to Neocognitron, but trained end-to-end using backpropagation
- Implements spatial invariance via convolutions and max-pooling
- Weight sharing reduces parameters
- Tanh/Softmax activations
- Good results on MNIST
- But did not scale up (yet)





LeCun, Bottou, Bengio, Haffner: Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998.

A Brief History of Deep Learning

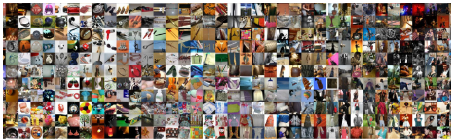
2009-2012: ImageNet and AlexNet

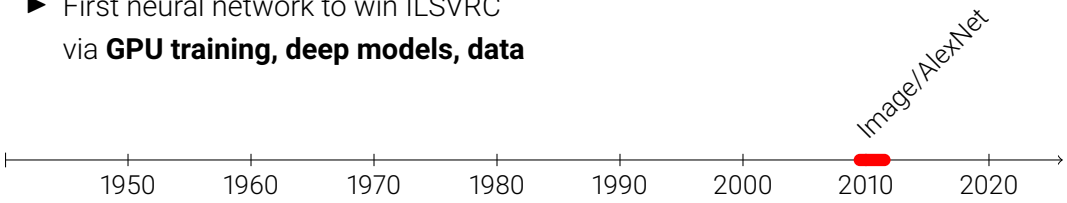
ImageNet

- Recognition benchmark (ILSVRC)
- 10 million annotated images
- 1000 categories

AlexNet

- First neural network to win ILSVRC via **GPU training, deep models, data**





Krizhevsky, Sutskever, Hinton. ImageNet classification with deep convolutional neural networks. NIPS, 2012.

A Brief History of Deep Learning

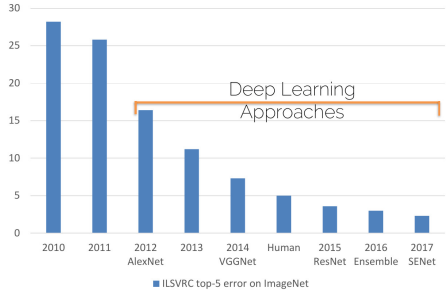
2009-2012: ImageNet and AlexNet

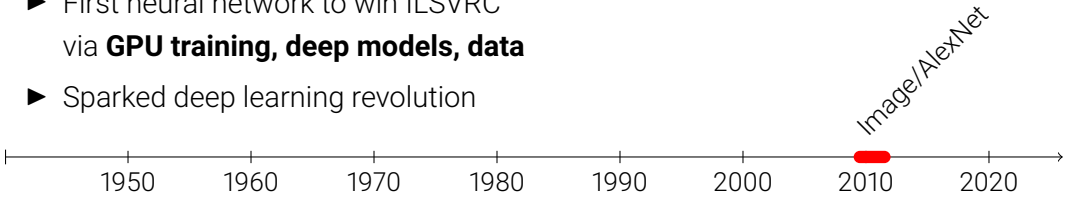
ImageNet

- Recognition benchmark (ILSVRC)
- 10 million annotated images
- 1000 categories

AlexNet

- First neural network to win ILSVRC via **GPU training, deep models, data**
- Sparked deep learning revolution



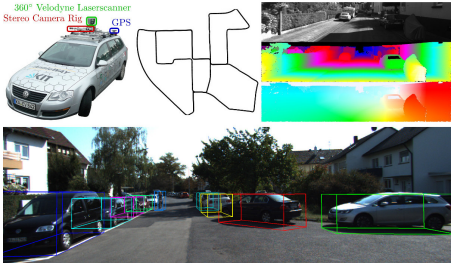


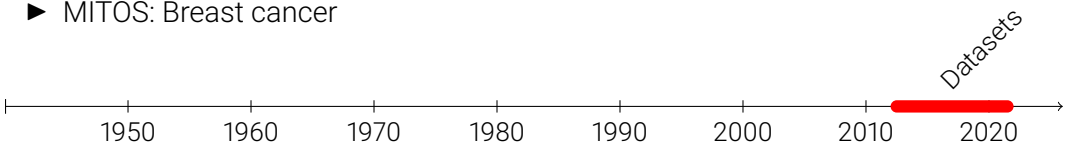
Krizhevsky, Sutskever, Hinton. ImageNet classification with deep convolutional neural networks. NIPS, 2012.

A Brief History of Deep Learning

2012-now: Golden Age of Datasets

- KITTI, Cityscapes: Self-driving
- PASCAL, MS COCO: Recognition
- ShapeNet, ScanNet: 3D DL
- GLUE: Language understanding
- Visual Genome: Vision/Language
- VisualQA: Question Answering
- MITOS: Breast cancer



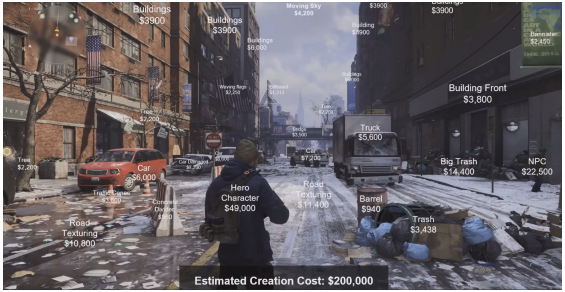


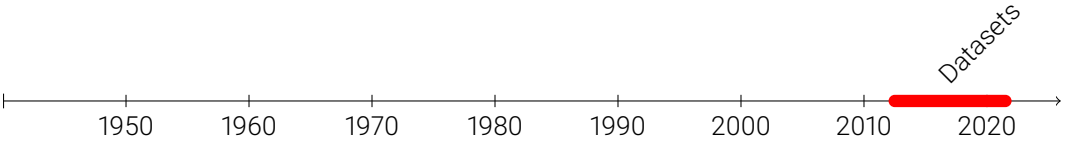
Geiger, Lenz and Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. CVPR, 2012.

A Brief History of Deep Learning

2012-now: Synthetic Data

- Annotating real data is expensive
- Led to surge of synthetic datasets
- Creating 3D assets is also costly



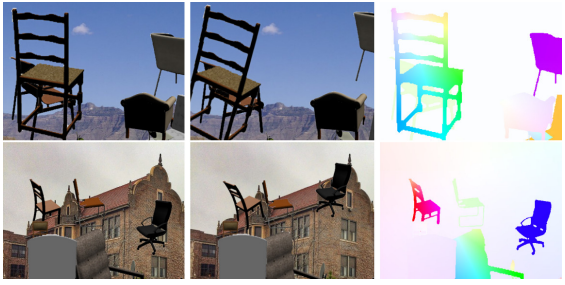


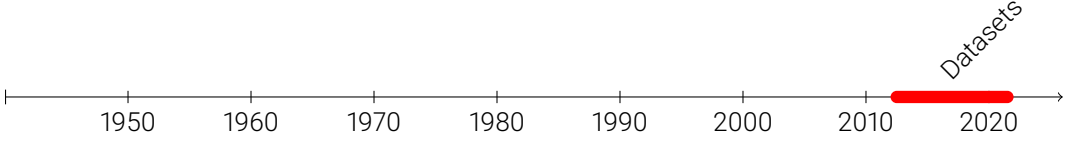
Dosovitskiy et al.: FlowNet: Learning Optical Flow with Convolutional Networks. ICCV, 2015.

A Brief History of Deep Learning

2012-now: Synthetic Data

- Annotating real data is expensive
- Led to surge of synthetic datasets
- Creating 3D assets is also costly
- But even very simple 3D datasets proved tremendously useful for pre-training (e.g., in optical flow)



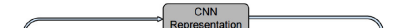


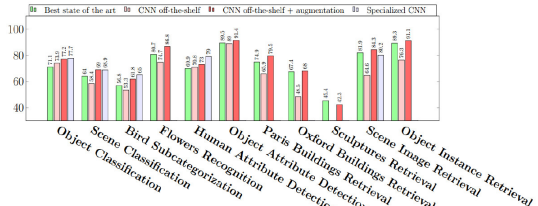
Dosovitskiy et al.: FlowNet: Learning Optical Flow with Convolutional Networks. ICCV, 2015.

A Brief History of Deep Learning

2014: Generalization

- Empirical demonstration that deep representations generalize well despite large number of parameters
- Pre-train CNN on large amounts of data on generic task (e.g., ImageNet)
- Fine-tune (re-train) only last layers on few data of a new task
- State-of-the-art performance



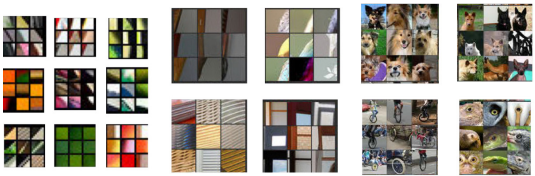


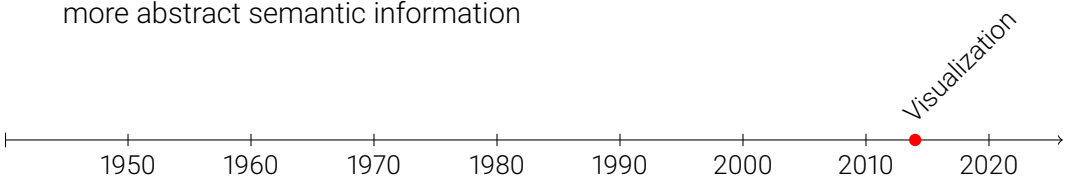
Razavian, Azizpour, Sullivan, Carlsson: CNN Features Off-the-Shelf: An Astounding Baseline for Recognition. CVPR Workshops, 2014.

A Brief History of Deep Learning

2014: Visualization

- Goal: provide insights into what the network (black box) has learned
- Visualized image regions that most strongly activate various neurons at different layers of the network
- Found that higher levels capture more abstract semantic information





Zeiler and Fergus: CNN Features Off-the-Shelf: An Astounding Baseline for Recognition. CVPR Workshops, 2014.

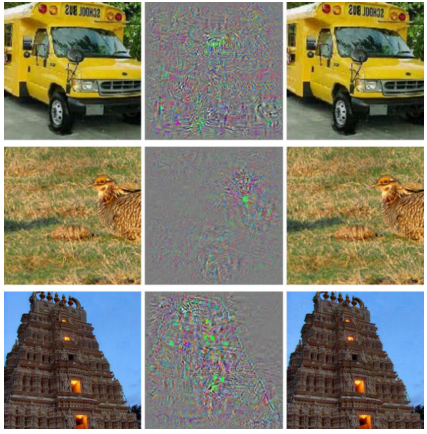
A Brief History of Deep Learning

2014: Adversarial Examples

- Accurate image classifiers can be fooled by imperceptible changes
- Adversarial example:

$$x + \underset{\Delta x}{\operatorname{argmin}} \{\|\Delta x\|_2 : f(x+\Delta x) \neq f(x)\}$$

- All images classified as “ostrich”



Szegedy et al.: Intriguing properties of neural networks. ICLR, 2014.

A Brief History of Deep Learning

2014: Domination of Deep Learning

- Machine translation (Seq2Seq)

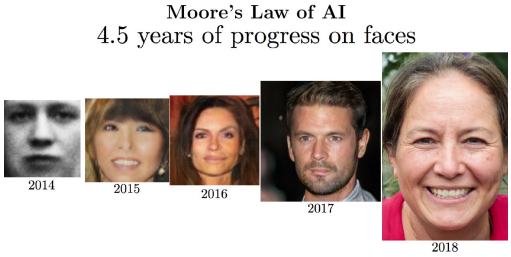
Type	Sentence
Our model	Ulrich UNK , membre du conseil d' administration du constructeur automobile Audi , affirme qu' il s' agit d' une pratique courante depuis des années pour que les téléphones portables puissent être collectés avant les réunions du conseil d' administration afin qu' ils ne soient pas utilisés comme appareils d' écoute à distance .
Truth	Ulrich Hackenberg , membre du conseil d' administration du constructeur automobile Audi , déclare que la collecte des téléphones portables avant les réunions du conseil , afin qu' ils ne puissent pas être utilisés comme appareils d' écoute à distance , est une pratique courante depuis des années .
Our model	" Les téléphones cellulaires , qui sont vraiment une question , non seulement parce qu' ils pourraient potentiellement causer des interférences avec les appareils de navigation , mais nous savons , selon la FCC , qu' ils pourraient interférer avec les tours de téléphone cellulaire lorsqu' ils sont dans l' air " , dit UNK .
Truth	" Les téléphones portables sont véritablement un problème , non seulement parce qu' ils pourraient éventuellement créer des interférences avec les instruments de navigation , mais parce que nous savons , d' après la FCC , qu' ils pourraient perturber les antennes-relais de téléphonie mobile s' ils sont utilisés à bord " , a déclaré Rosenker .
Our model	Avec la crémation , il y a un " sentiment de violence contre le corps d' un être cher " , qui sera " réduit à une pile de cendres " en très peu de temps au lieu d' un processus de décomposition " qui accompagnera les étapes du deuil " .
Truth	Il y a , avec la crémation , " une violence faite au corps aimé " , qui va être " réduit à un tas de cendres " en très peu de temps , et non après un processus de décomposition , qui " accompagnerait les phases du deuil " .

Sutskever, Vinyals, Quoc: Sequence to Sequence Learning with Neural Networks. NIPS, 2014.

A Brief History of Deep Learning

2014: Domination of Deep Learning

- Machine translation (Seq2Seq)
- Deep generative models (VAEs, GANs) produce compelling images



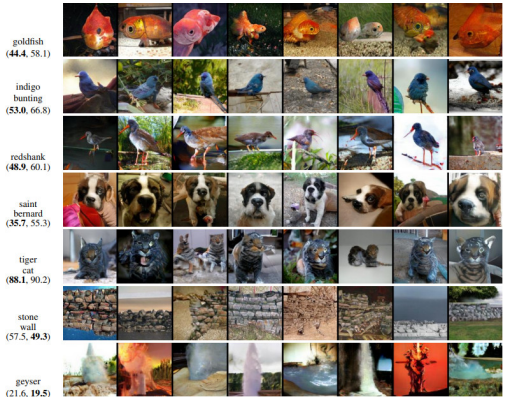
1950 1960 1970 1980 1990 2000 2010 2020

Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio: Generative Adversarial Networks. NIPS, 2014.

A Brief History of Deep Learning

2014: Domination of Deep Learning

- Machine translation (Seq2Seq)
- Deep generative models (VAEs, GANs) produce compelling images

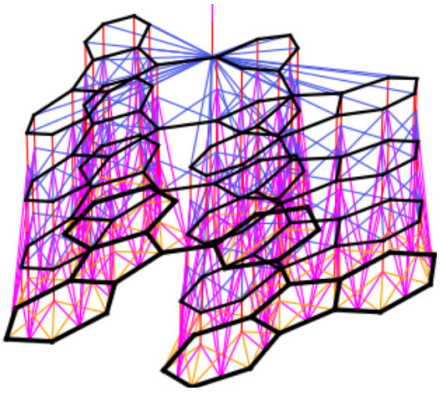


Zhang, Goodfellow, Metaxas, Odena: Self-Attention Generative Adversarial Networks. ICML, 2019.

A Brief History of Deep Learning

2014: Domination of Deep Learning

- Machine translation (Seq2Seq)
- Deep generative models (VAEs, GANs) produce compelling images
- Graph Neural Networks (GNNs) revolutionize the prediction of molecular properties

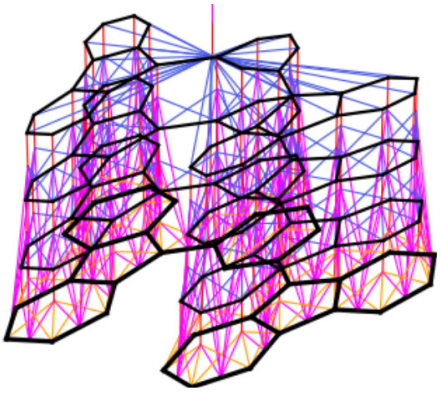


Duvenaud et al.: Convolutional Networks on Graphs for Learning Molecular Fingerprints. NIPS 2015.

A Brief History of Deep Learning

2014: Domination of Deep Learning

- Machine translation (Seq2Seq)
- Deep generative models (VAEs, GANs) produce compelling images
- Graph Neural Networks (GNNs) revolutionize the prediction of molecular properties
- Dramatic gains in vision and speech (Moore's Law of AI)

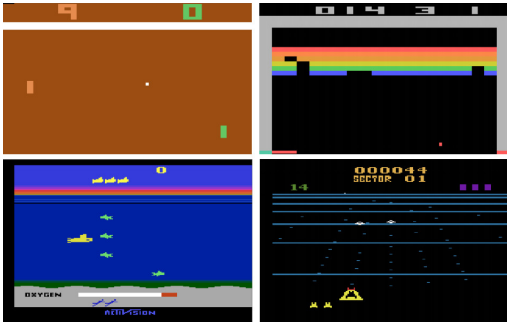


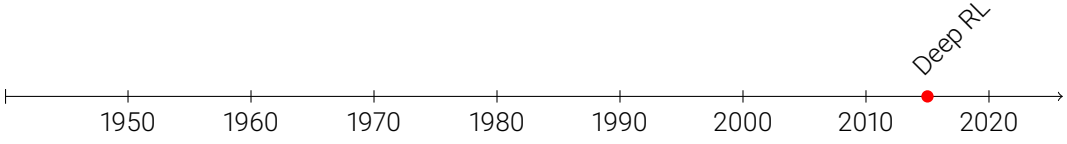
Duvenaud et al.: Convolutional Networks on Graphs for Learning Molecular Fingerprints. NIPS 2015.

A Brief History of Deep Learning

2015: Deep Reinforcement Learning

- Learning a policy (state→action) through random exploration and reward signals (e.g., game score)
- No other supervision
- Success on many Atari games
- But some games remain hard



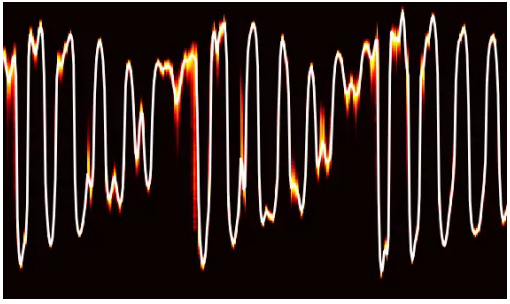


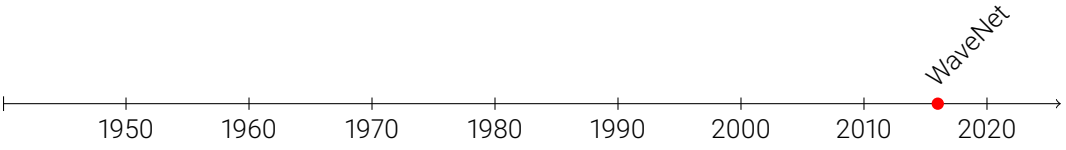
Mnih et al.: Human-level control through deep reinforcement learning. Nature, 2015.

A Brief History of Deep Learning

2016: WaveNet

- Deep generative model of raw audio waveforms
- Generates **speech** which mimics human voice
- Generates **music**



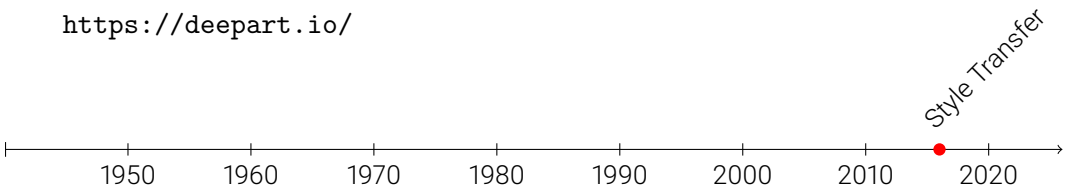


Oord et al.: WaveNet: A Generative Model for Raw Audio. Arxiv, 2016.

A Brief History of Deep Learning

2016: Style Transfer

- Manipulate photograph to adopt style of a another image (painting)
- Uses deep network pre-trained on ImageNet for disentangling content from style
- It is fun! Try yourself: `https://deepart.io/`

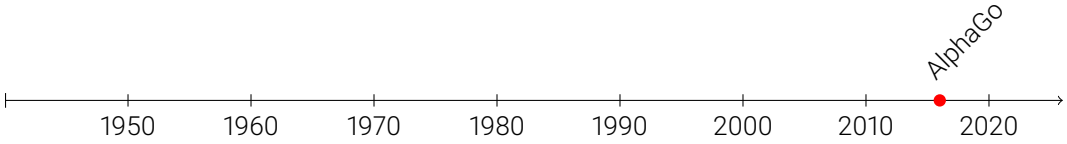


Gatys, Ecker and Bethge: Image Style Transfer Using Convolutional Neural Networks. CVPR, 2016.

A Brief History of Deep Learning

2016: AlphaGo defeats Lee Sedol

- Developed by DeepMind
- Combines deep learning with Monte Carlo tree search
- First computer program to defeat professional player
- AlphaZero (2017) learns via self-play and masters multiple games

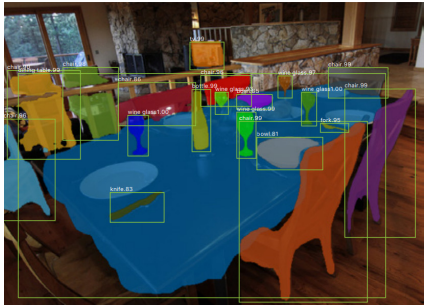


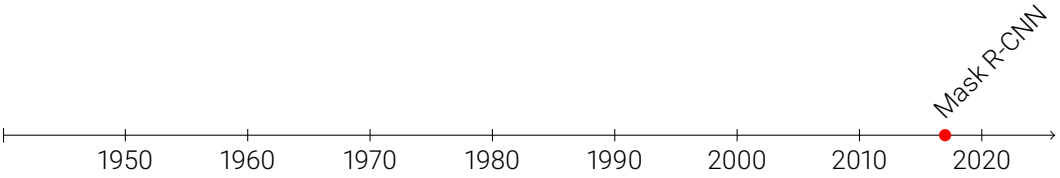
Silver et al.: Mastering the game of Go without human knowledge. Nature, 2017.

A Brief History of Deep Learning

2017: Mask R-CNN

- Deep neural network for joint object detection and instance segmentation
- Outputs “structured object”, not only a single number (class label)
- State-of-the-art on MS-COCO



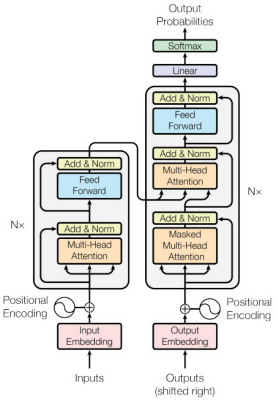


He, Gkioxari, Dollár and Ross Girshick: Mask R-CNN. ICCV, 2017.

A Brief History of Deep Learning

2017-2018: Transformers and BERT

- Transformers: Attention replaces recurrence and convolutions



BERT/GLUE

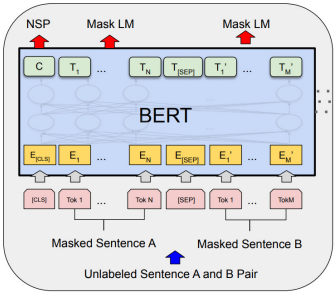
1950 1960 1970 1980 1990 2000 2010 2020

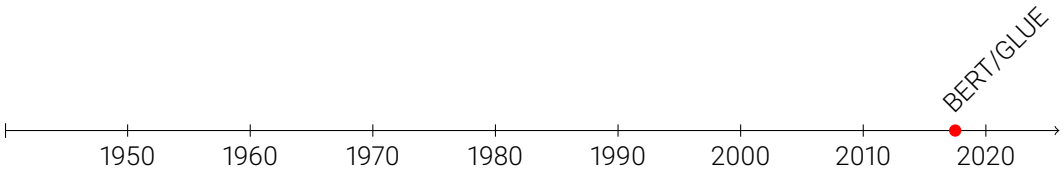
Vaswani et al.: Attention is All you Need. NIPS 2017.

A Brief History of Deep Learning

2017-2018: Transformers and BERT

- Transformers: Attention replaces recurrence and convolutions
- BERT: Pre-training of language models on unlabeled text





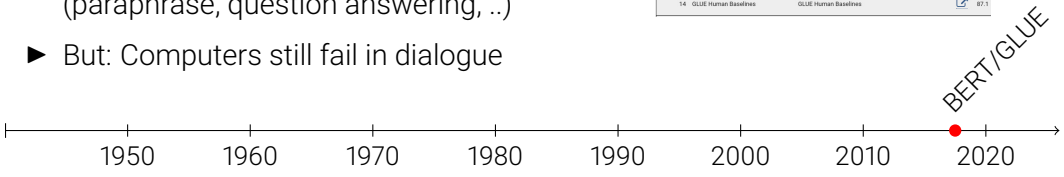
Devlin, Chang, Lee and Toutanova: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Arxiv, 2018.

A Brief History of Deep Learning

2017-2018: Transformers and BERT

- Transformers: Attention replaces recurrence and convolutions
- BERT: Pre-training of language models on unlabeled text
- GLUE: Superhuman performance on some language understanding tasks (paraphrase, question answering, ..)
- But: Computers still fail in dialogue

	Rank	Name	Model	URL	Score
	1	HFL iFLYTEK	MacALBERT + DKM		90.7
+	2	Alibaba DAMO NLP	StructBERT + TAPT		90.6
+	3	PING-AN Omni-Sinitic	ALBERT + DAAF + NAS		90.6
	4	ERNIE Team - Baidu	ERNIE		90.4
	5	T5 Team - Google	T5		90.3
	6	Microsoft D365 AI & MSR AI & GATECHMT-DNN-SMART			89.9
+	7	Zihang Dai	Funnel-Transformer (Ensemble B10-10-10H1024)		89.7
+	8	ELECTRA Team	ELECTRA-Large + Standard Tricks		89.4
+	9	Huawei Noah's Ark Lab	NEZHA-Large		89.1
+	10	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)		88.4
	11	Junjie Yang	HIRE-RoBERTa		88.3
	12	Facebook AI	RoBERTa		88.1
+	13	Microsoft D365 AI & MSR AI	MT-DNN-ensemble		87.6
	14	GLUE Human Baselines	GLUE Human Baselines		87.1



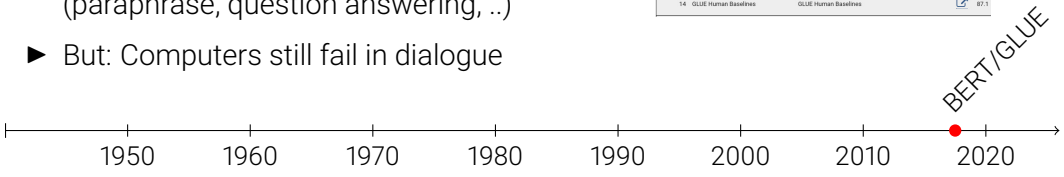
Wang et al.: GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. ICLR, 2019.

A Brief History of Deep Learning

2017-2018: Transformers and BERT

- Transformers: Attention replaces recurrence and convolutions
- BERT: Pre-training of language models on unlabeled text
- GLUE: Superhuman performance on some language understanding tasks (paraphrase, question answering, ..)
- But: Computers still fail in dialogue

	Rank	Name	Model	URL	Score
	1	HFL iFLYTEK	MacALBERT + DKM		90.7
+	2	Alibaba DAMO NLP	StructBERT + TAPT		90.6
+	3	PING-AN Omni-Sinitic	ALBERT + DAAF + NAS		90.6
	4	ERNIE Team - Baidu	ERNIE		90.4
	5	T5 Team - Google	T5		90.3
	6	Microsoft D365 AI & MSR AI & GATECHMT-DNN-SMART			89.9
+	7	Zihang Dai	Funnel-Transformer (Ensemble B10-10-10H1024)		89.7
+	8	ELECTRA Team	ELECTRA-Large + Standard Tricks		89.4
+	9	Huawei Noah's Ark Lab	NEZHA-Large		89.1
+	10	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)		88.4
	11	Junjie Yang	HIRE-RoBERTa		88.3
	12	Facebook AI	RoBERTa		88.1
+	13	Microsoft D365 AI & MSR AI	MT-DNN-ensemble		87.6
	14	GLUE Human Baselines	GLUE Human Baselines		87.1



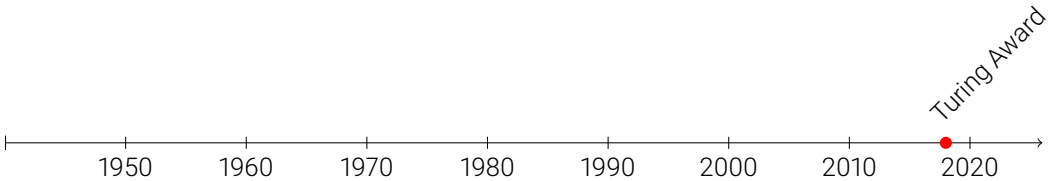
Wang et al.: GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. ICLR, 2019.

A Brief History of Deep Learning

2018: Turing Award

In 2018, the “nobel price of computing” has been awarded to:

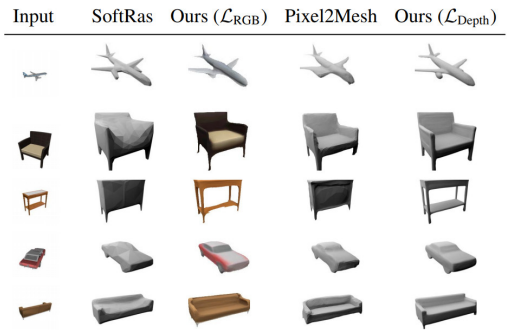
- Yoshua Bengio
- Geoffrey Hinton
- Yann LeCun

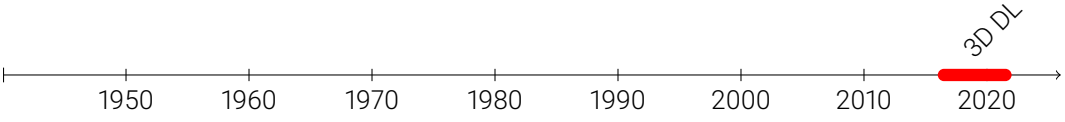


A Brief History of Deep Learning

2016-2020: 3D Deep Learning

- First models to successfully output 3D representations
- Voxels, point clouds, meshes, implicit representations
- Prediction of 3D models even from a single image
- Geometry, materials, light, motion



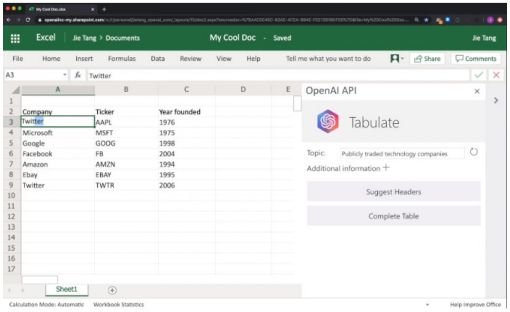


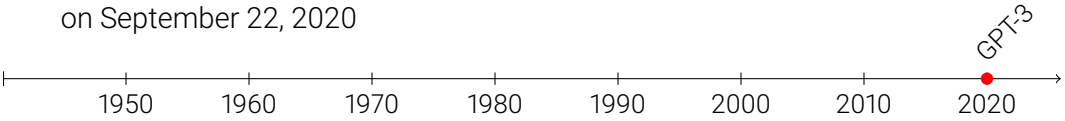
Niemeyer, Mescheder, Oechsle, Geiger: Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision. CVPR, 2020.

A Brief History of Deep Learning

2020: GPT-3

- Language model by OpenAI
- 175 Billion parameters
- Text-in / text-out interface
- Many use cases: coding, poetry, blogging, news articles, chatbots
- Controversial discussions
- Licensed exclusively to Microsoft on September 22, 2020





Brown et al.: Language Models are Few-Shot Learners. Arxiv, 2020.

A Brief History of Deep Learning

Current Challenges

- Un-/Self-Supervised Learning
- Interactive learning
- Accuracy (e.g., self-driving)
- Robustness and generalization
- Inductive biases
- Understanding and mathematics
- Memory and compute
- Ethics and legal questions
- Does “Moore’s Law of AI” continue?

1.3
Machine Learning Basics

Goodfellow et al.: Deep Learning, Chapter 5

http://www.deeplearningbook.org/contents/ml.html

Learning Problems

- **Supervised learning**
  - Learn model parameters using dataset of data-label pairs $\{(x_i, y_i)\}_{i=1}^{N}$
  - Examples: Classification, regression, structured prediction
- **Unsupervised learning**
  - Learn model parameters using dataset without labels $\{x_i\}_{i=1}^{N}$
  - Examples: Clustering, dimensionality reduction, generative models
- **Self-supervised learning**
  - Learn model parameters using dataset of data-data pairs $\{(x_i, x_i')\}_{i=1}^{N}$
  - Examples: Self-supervised stereo/flow, contrastive learning
- **Reinforcement learning**
  - Learn model parameters using active exploration from sparse rewards
  - Examples: deep q learning, gradient policy, actor critique

Supervised Learning

Classification, Regression, Structured Prediction

Classification / Regression:

$$f : \mathcal{X} \to \mathbb{N} \qquad or \qquad f : \mathcal{X} \to \mathbb{R}$$

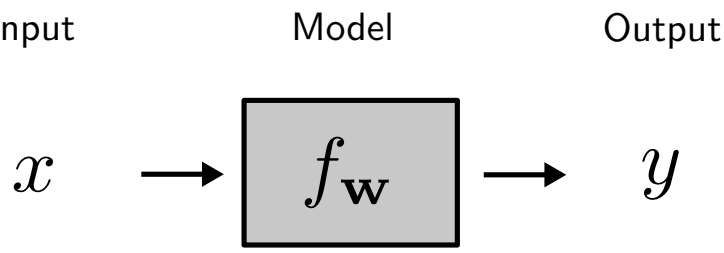
- Inputs $x \in \mathcal{X}$ can be any kind of objects
  - images, text, audio, sequence of amino acids, ...
- Output $y \in \mathbb{N} / y \in \mathbb{R}$ is a discrete or real number
  - classification, regression, density estimation, ...

Structured Output Learning:

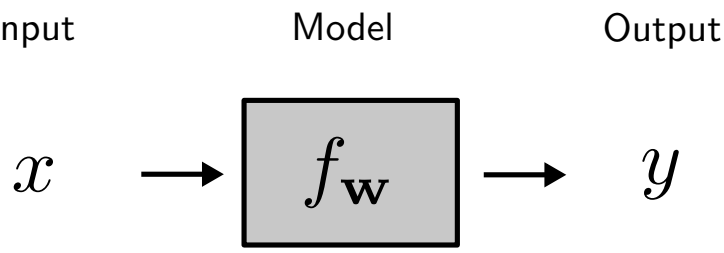
$$f : \mathcal{X} \to \mathcal{Y}$$

- Inputs $x \in \mathcal{X}$ can be any kind of objects
- Outputs $y \in \mathcal{Y}$ are complex (structured) objects
  - images, text, parse trees, folds of a protein, computer programs, ...

Supervised Learning

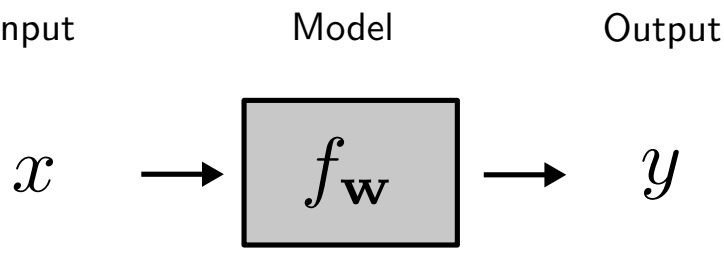


Supervised Learning



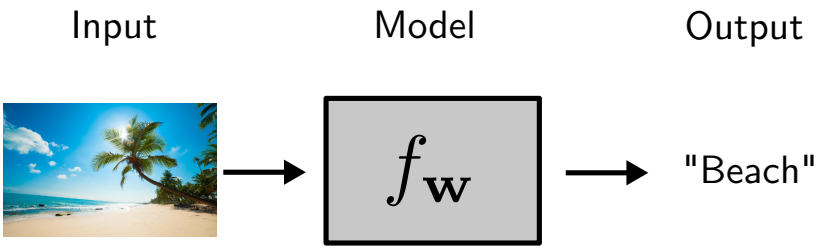
- **Learning:** Estimate parameters $\mathbf{w}$ from training data $\{(x_i, y_i)\}_{i=1}^{N}$

Supervised Learning



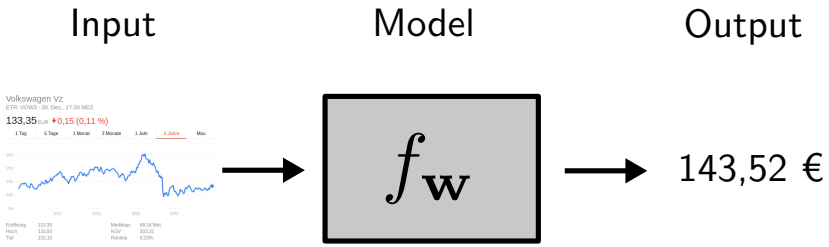
- **Learning:** Estimate parameters $\mathbf{w}$ from training data $\{(x_i, y_i)\}_{i=1}^{N}$
- **Inference:** Make novel predictions: $y = f_{\mathbf{w}}(x)$

Classification



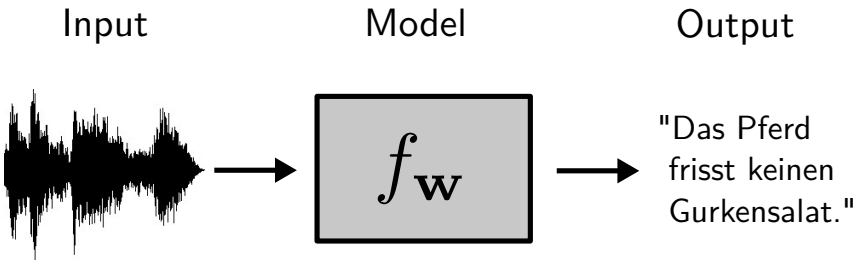
- **Mapping:** $f_{\mathbf{w}} : \mathbb{R}^{W \times H} \rightarrow \{\text{“Beach”}, \text{“No Beach”}\}$

Regression



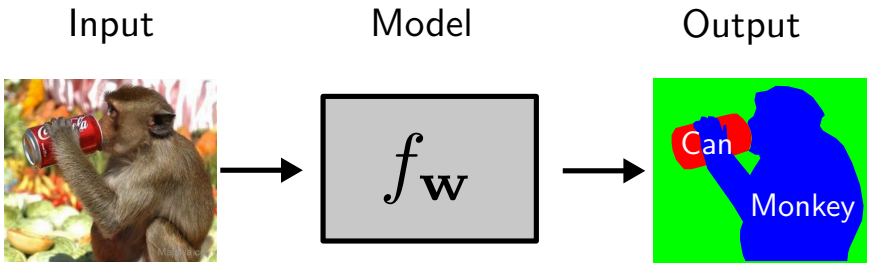
- **Mapping:** $f_{\mathbf{w}} : \mathbb{R}^N \to \mathbb{R}$

Structured Prediction



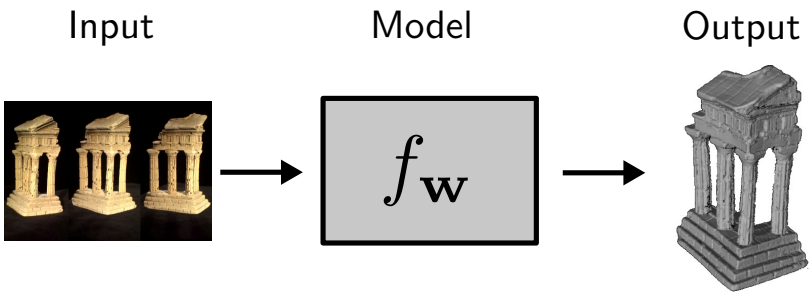
- **Mapping:** $f_{\mathbf{w}} : \mathbb{R}^N \rightarrow \{1, \ldots, C\}^M$

Structured Prediction



- **Mapping:** $f_{\mathbf{w}} : \mathbb{R}^{W \times H} \rightarrow \{1, \ldots, C\}^{W \times H}$

Structured Prediction



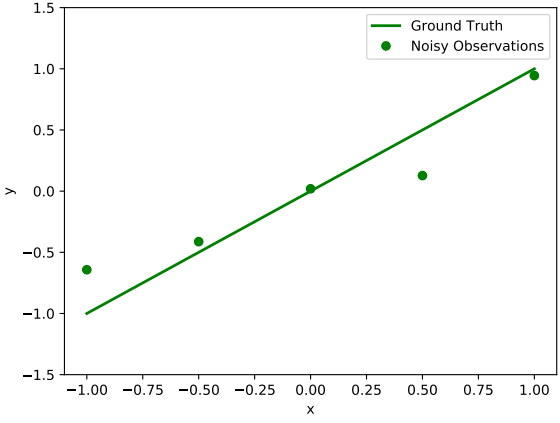
- **Mapping:** $f_{\mathbf{w}} : \mathbb{R}^{W \times H \times N} \rightarrow \{0,1\}^{M^3}$
- **Suppose:** 32^3 voxels, binary variable per voxel (occupied/free)
- **Question:** How many different reconstructions? $2^{32^3} = 2^{32768}$
- **Comparison:** Number of atoms in the universe? $\sim 2^{273}$

Linear Regression

Linear Regression

Let $\mathcal{X}$ denote a dataset of size N and let $(\mathbf{x}_i, y_i) \in \mathcal{X}$ denote its elements ($y_i \in \mathbb{R}$).

Goal: Predict y for a previously unseen input $\mathbf{x}$. The input $\mathbf{x}$ may be multidimensional.

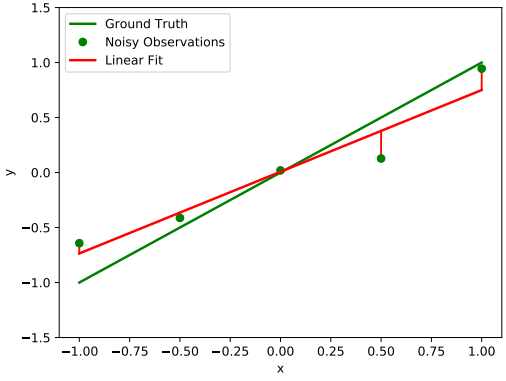


Linear Regression

The **error function** $E(\mathbf{w})$ measures the displacement along the y dimension between the data points (green) and the model $f(\mathbf{x}, \mathbf{w})$ (red) specified by the parameters $\mathbf{w}$.

$$f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^\top \mathbf{x}$$

$$\begin{aligned} E(\mathbf{w}) &= \sum_{i=1}^{N} (f(\mathbf{x}_i, \mathbf{w}) - y_i)^2 \\ &= \sum_{i=1}^{N} \left(\mathbf{x}_i^\top \mathbf{w} - y_i\right)^2 \\ &= \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 \end{aligned}$$



Here: $\mathbf{x} = [1, x]^\top \Rightarrow f(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x$

Linear Regression

The **gradient of the error function** with respect to the parameters $\mathbf{w}$ is given by:

$$\begin{aligned}\nabla_{\mathbf{w}} E(\mathbf{w}) &= \nabla_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 \\ &= \nabla_{\mathbf{w}} (\mathbf{X}\mathbf{w} - \mathbf{y})^\top (\mathbf{X}\mathbf{w} - \mathbf{y}) \\ &= \nabla_{\mathbf{w}} \Big(\mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w} - 2\mathbf{w}^\top \mathbf{X}^\top \mathbf{y} + \mathbf{y}^\top \mathbf{y} \Big) \\ &= 2\mathbf{X}^\top \mathbf{X}\mathbf{w} - 2\mathbf{X}^\top \mathbf{y}\end{aligned}$$

As $E(\mathbf{w})$ is quadratic and convex in $\mathbf{w}$, its minimizer (wrt. $\mathbf{w}$) is given in closed form:

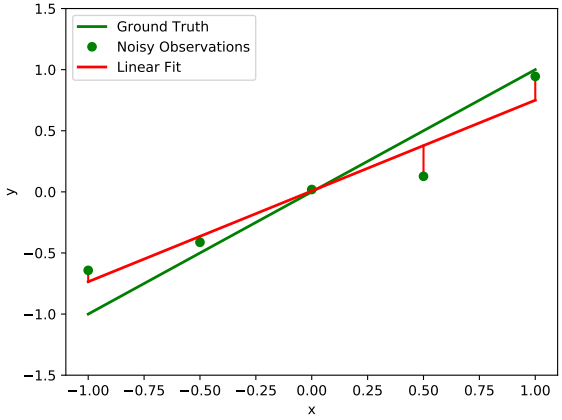
$$\nabla_{\mathbf{w}} E(\mathbf{w}) = 0$$

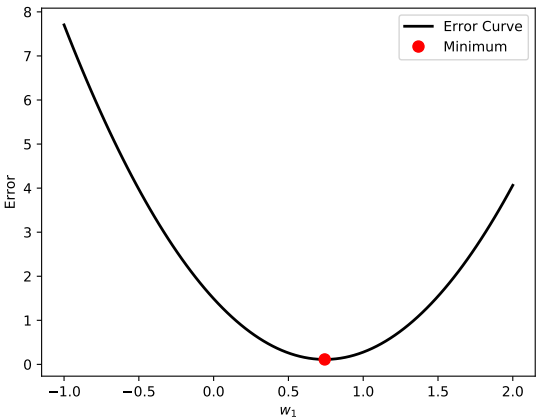
$$\Rightarrow \mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

The matrix $(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$ is also called **Moore-Penrose inverse** or pseudoinverse.

Example: Line Fitting

Line Fitting





Linear least squares fit of model $f(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x$ (red) to data points (green).
Errors are also shown in red. Right: Error function $E(\mathbf{w})$ wrt. parameter w_1.

Example: Polynomial Curve Fitting

Polynomial Curve Fitting

Let us choose a **polynomial of order M** to model dataset $\mathcal{X}$:

$$f(x, \mathbf{w}) = \sum_{j=0}^{M} w_j x^j = \mathbf{w}^\top \mathbf{x} \qquad \text{with features} \qquad \mathbf{x} = (1, x^1, x^2, \ldots, x^M)^\top$$

Tasks:

- **Training:** Estimate $\mathbf{w}$ from dataset $\mathcal{X}$
- **Inference:** Predict y for novel x given estimated $\mathbf{w}$

Note:

- Features can be anything, including multi-dimensional inputs (e.g., images, audio), radial basis functions, sine/cosine functions, etc. In this example: monomials.

Polynomial Curve Fitting

Let us choose a **polynomial of order M** to model the dataset $\mathcal{X}$:

$$f(x, \mathbf{w}) = \sum_{j=0}^{M} w_j x^j = \mathbf{w}^\top \mathbf{x} \qquad \text{with features} \qquad \mathbf{x} = (1, x^1, x^2, \ldots, x^M)^\top$$

How can we estimate $\mathbf{w}$ from $\mathcal{X}$?

Polynomial Curve Fitting

The **error function** from above is quadratic in $\mathbf{w}$ but not in x:

$$E(\mathbf{w}) = \sum_{i=1}^{N} \left(f(x_i, \mathbf{w}) - y_i\right)^2 = \sum_{i=1}^{N} \left(\mathbf{w}^\top \mathbf{x}_i - y_i\right)^2 = \sum_{i=1}^{N} \left(\sum_{j=0}^{M} w_j x_i^j - y_i\right)^2$$

It can be rewritten in the **matrix-vector notation** (i.e., as linear regression problem)

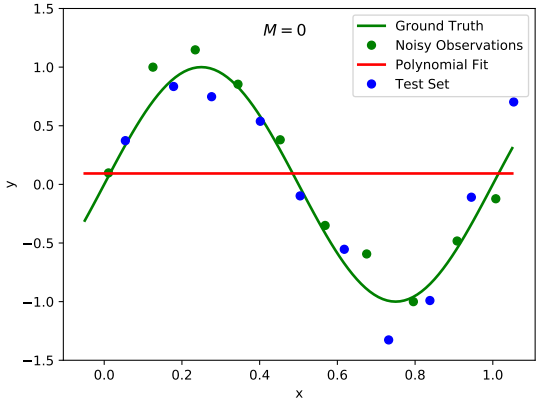
$$E(\mathbf{w}) = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$$

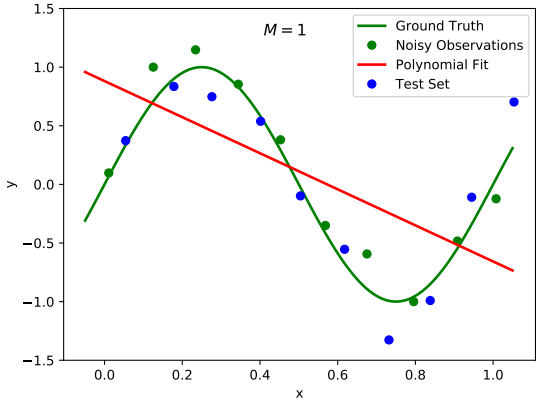
with feature matrix $\mathbf{X}$, observation vector $\mathbf{y}$ and weight vector $\mathbf{w}$:

$$\mathbf{X} = \begin{pmatrix} \vdots & \vdots & \vdots & & \vdots \\ 1 & x_i & x_i^2 & \dots & x_i^M \\ \vdots & \vdots & \vdots & & \vdots \end{pmatrix} \qquad \mathbf{y} = \begin{pmatrix} \vdots \\ y_i \\ \vdots \end{pmatrix} \qquad \mathbf{w} = \begin{pmatrix} w_0 \\ \vdots \\ w_M \end{pmatrix}$$

Polynomial Curve Fitting Results

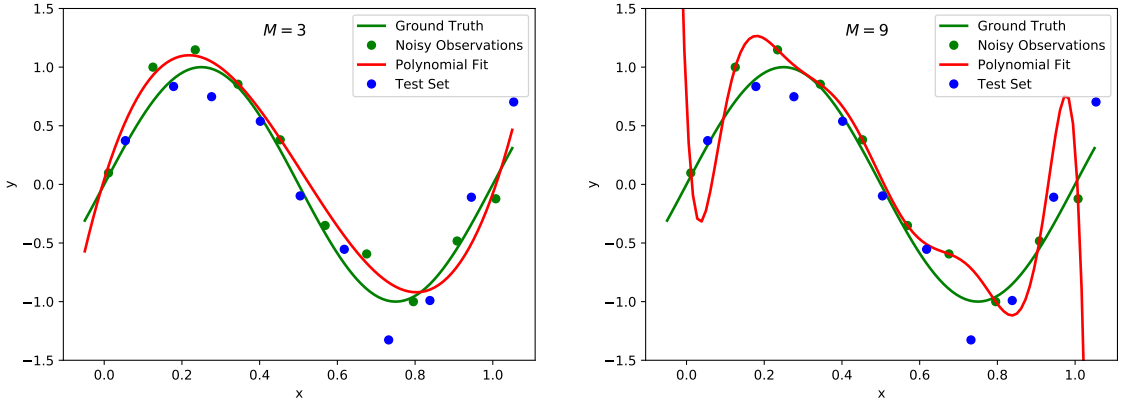
Polynomial Curve Fitting





Plots of polynomials of various degrees M (red) fitted to the data (green). We observe underfitting ($M = 0/1$) and overfitting ($M = 9$). This is a model selection problem.

Polynomial Curve Fitting



Plots of polynomials of various degrees M (red) fitted to the data (green). We observe underfitting ($M = 0/1$) and overfitting ($M = 9$). This is a model selection problem.

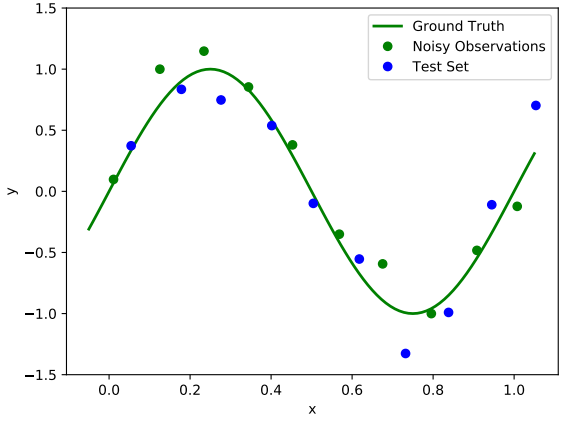
Capacity, Overfitting and Underfitting

Goal:

- Perform well on new, previously unseen inputs (test set, blue), not only on the training set (green)
- This is called **generalization** and separates ML from optimization
- Assumption: training and test data independent and identically (i.i.d.) drawn from distribution $p_{data}(x, y)$
- Here:

$$p_{data}(x) = \mathcal{U}(0, 1)$$

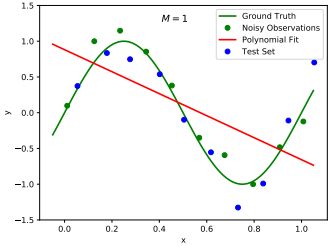
$$p_{data}(y|x) = \mathcal{N}(\sin(2\pi x), \sigma)$$



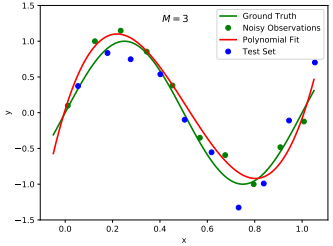
Capacity, Overfitting and Underfitting

Terminology:

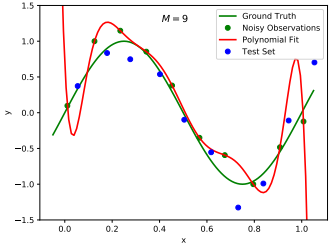
- **Capacity:** Complexity of functions which can be represented by model f
- **Underfitting:** Model too simple, does not achieve low error on training set
- **Overfitting:** Training error small, but test error (= generalization error) large



Capacity too low



Capacity about right

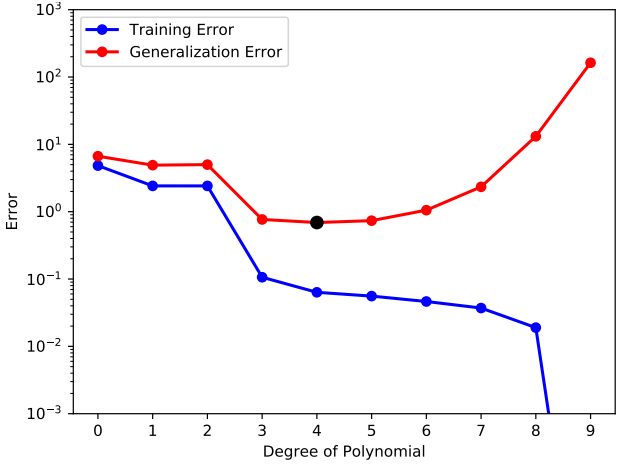


Capacity too high

Capacity, Overfitting and Underfitting

Example: Generalization error for various polynomial degrees M

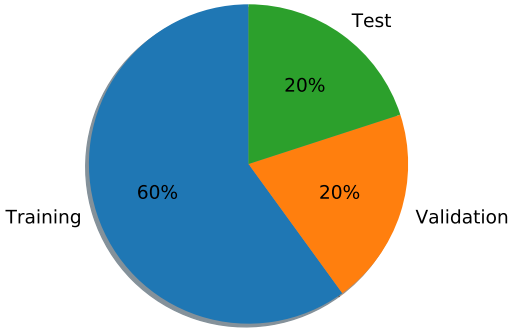
- Model selection: Select model with the smallest generalization error



Capacity, Overfitting and Underfitting

General Approach: Split dataset into training, validation and test set

- Choose hyperparameters (e.g., degree of polynomial, learning rate in neural net, ..) using validation set. Important: Evaluate once on test set (typically not available).



- When dataset is small, use (k-fold) cross validation instead of fixed split.

Ridge Regression

Ridge Regression

Polynomial Curve Model:

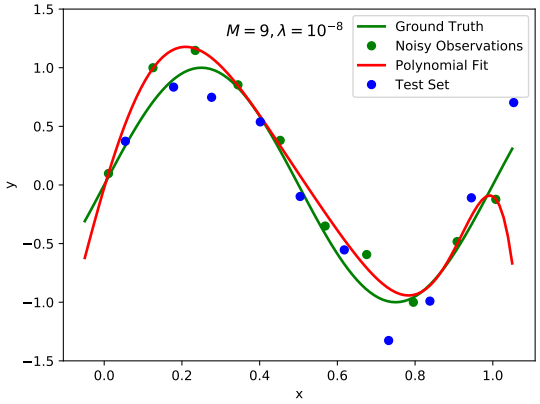
$$f(x, \mathbf{w}) = \sum_{j=0}^{M} w_j x^j = \mathbf{w}^\top \mathbf{x} \qquad \text{with features} \qquad \mathbf{x} = (1, x^1, x^2, \ldots, x^M)^\top$$

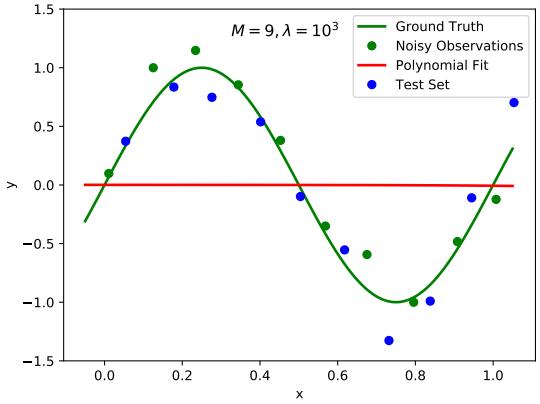
Ridge Regression:

$$\begin{aligned} E(\mathbf{w}) &= \sum_{i=1}^{N} (f(x_i, \mathbf{w}) - y_i)^2 + \lambda \sum_{j=0}^{M} w^2 \\ &= \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_2^2 \end{aligned}$$

- Idea: Discourage large parameters by adding a regularization term with strength λ
- Closed form solution: $\mathbf{w} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$

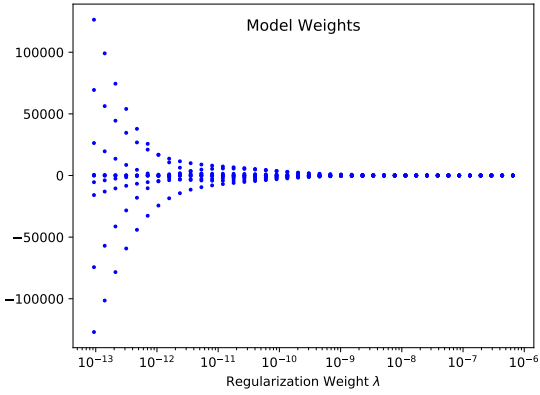
Ridge Regression

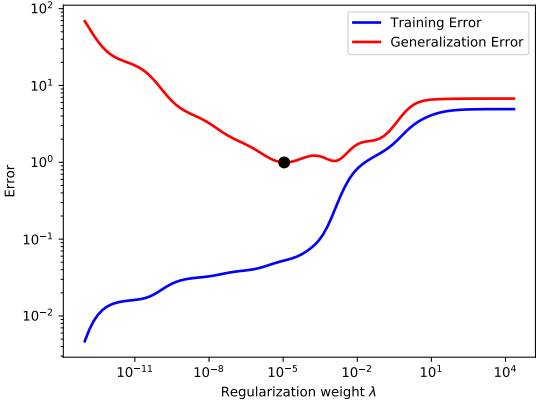




Plots of polynomial with degree $M = 9$ fitted to 10 data points using ridge regression. Left: weak regularization ($\lambda = 10^{-8}$). Right: strong regularization (right, $\lambda = 10^{3}$).

Ridge Regression





Left: With low regularization, parameters can become very large (ill-conditioning).

Right: Select model with the smallest generalization error on the validation set.

Estimators, Bias and Variance

Estimators, Bias and Variance

Point Estimator:

- A point estimator $g(\cdot)$ is function that maps a dataset $\mathcal{X}$ to model parameters $\hat{\mathbf{w}}$:

$$\hat{\mathbf{w}} = g(\mathcal{X})$$

- Example: Estimator of ridge regression model: $\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$
- We use the hat notation to denote that $\hat{\mathbf{w}}$ is an estimate
- A good estimator is a function that returns a parameter set close to the true one
- The data $\mathcal{X} = \{(x_i, y_i)\}$ is drawn from a random process $(x_i, y_i) \sim p_{data}(\cdot)$
- Thus, any function of the data is random and $\hat{\mathbf{w}}$ is a random variable.

Estimators, Bias and Variance

Properties of Point Estimators:

Bias:

$$\text{Bias}(\hat{\mathbf{w}}) = \mathbb{E}(\hat{\mathbf{w}}) - \mathbf{w}$$

- Expectation over datasets $\mathcal{X}$
- $\hat{\mathbf{w}}$ is unbiased $\Leftrightarrow \text{Bias}(\hat{\mathbf{w}}) = 0$
- A good estimator has little bias

Variance:

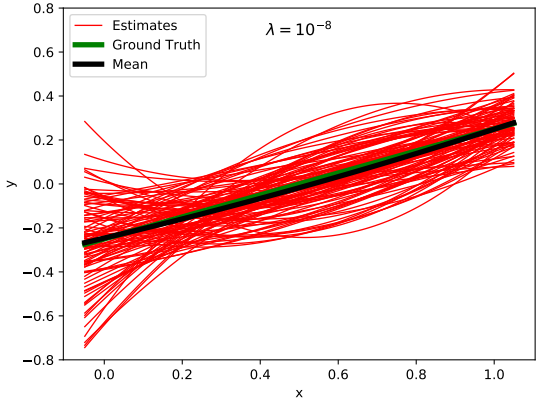
$$\text{Var}(\hat{\mathbf{w}}) = \mathbb{E}(\hat{\mathbf{w}}^2) - \mathbb{E}(\hat{\mathbf{w}})^2$$

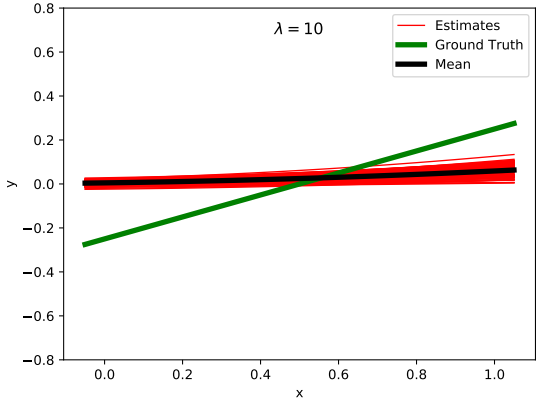
- Variance over datasets $\mathcal{X}$
- $\sqrt{\text{Var}(\hat{\mathbf{w}})}$ is called "standard error"
- A good estimator has low variance

Bias-Variance Dilemma:

- Statistical learning theory tells us that we can't have both $\Rightarrow$ there is a trade-off

Estimators, Bias and Variance

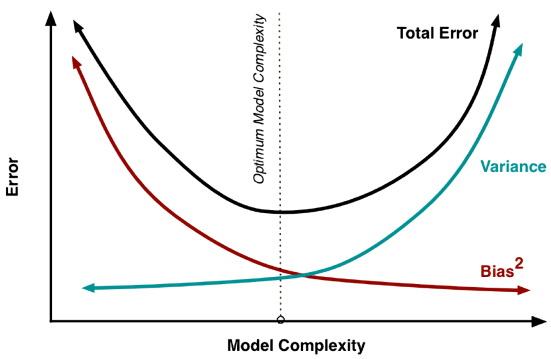




Ridge regression with weak ($\lambda = 10^{-8}$) and strong ($\lambda = 10$) regularization.

Green: True model. Black: Plot of model with mean parameters $\bar{\mathbf{w}} = \mathbb{E}(\mathbf{w})$.

Estimators, Bias and Variance



- There is a bias-variance tradeoff: $\mathbb{E}[(\hat{\mathbf{w}} - \mathbf{w})^2] = \mathrm{Bias}(\hat{\mathbf{w}})^2 + \mathrm{Var}(\hat{\mathbf{w}})$
- Or not? In deep neural networks the test error decreases with network width!
  `https://www.bradyneal.com/bias-variance-tradeoff-textbooks-update`

Neal et al.: A Modern Take on the Bias-Variance Tradeoff in Neural Networks. ICML Workshops, 2019.

Maximum Likelihood Estimation

Maximum Likelihood Estimation

- We now reinterpret our results by taking a **probabilistic viewpoint**
- Let $\mathcal{X} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ be a dataset with samples drawn i.i.d. from p_{data}
- Let the model $p_{model}(y|\mathbf{x}, \mathbf{w})$ be a parametric family of probability distributions
- The conditional **maximum likelihood estimator** for $\mathbf{w}$ is given by

$$\begin{aligned}\hat{\mathbf{w}}_{ML} &= \underset{\mathbf{w}}{\operatorname{argmax}}\ p_{model}(\mathbf{y}|\mathbf{X}, \mathbf{w}) \\ &\overset{\text{iid}}{=} \underset{\mathbf{w}}{\operatorname{argmax}}\ \prod_{i=1}^{N} p_{model}(y_i|\mathbf{x}_i, \mathbf{w}) \\ &= \underset{\mathbf{w}}{\operatorname{argmax}}\ \underbrace{\sum_{i=1}^{N} \log p_{model}(y_i|\mathbf{x}_i, \mathbf{w})}_{\text{Log-Likelihood}}\end{aligned}$$

Note: In this lecture we consider log to be the natural logarithm.

Maximum Likelihood Estimation

Example: Assuming $p_{model}(y|\mathbf{x}, \mathbf{w}) = \mathcal{N}(y|\mathbf{w}^\top \mathbf{x}, \sigma)$, we obtain

$$
\begin{aligned}
\hat{\mathbf{w}}_{ML} &= \underset{\mathbf{w}}{\operatorname{argmax}} \sum_{i=1}^{N} \log p_{model}(y_i|\mathbf{x}_i, \mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \sum_{i=1}^{N} \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}\left(\mathbf{w}^\top \mathbf{x}_i - y_i\right)^2} \right] \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} -\sum_{i=1}^{N} \frac{1}{2} \log(2\pi\sigma^2) - \sum_{i=1}^{N} \frac{1}{2\sigma^2} \left(\mathbf{w}^\top \mathbf{x}_i - y_i\right)^2 \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} -\sum_{i=1}^{N} \left(\mathbf{w}^\top \mathbf{x}_i - y_i\right)^2 \\
&= \underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2
\end{aligned}
$$

Maximum Likelihood Estimation

We see that choosing $p_{model}(y|\mathbf{x}, \mathbf{w})$ to be Gaussian causes maximum likelihood to yield exactly the same least squares estimator derived before:

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \ \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$$

Variations:

- If we were choosing $p_{model}(y|\mathbf{x}, \mathbf{w})$ as a Laplace distribution, we would obtain an estimator that minimizes the ℓ_1 norm: $\hat{\mathbf{w}} = \operatorname{argmin}_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_1$
- Assuming a Gaussian distribution over the parameters $\mathbf{w}$ and performing a maximum a-posteriori (MAP) estimation yields ridge regression:

$$\underset{\mathbf{w}}{\operatorname{argmax}} \ p(\mathbf{w}|y, \mathbf{x}) = \underset{\mathbf{w}}{\operatorname{argmax}} \ p(y|\mathbf{x}, \mathbf{w})p(\mathbf{w})$$

Maximum Likelihood Estimation

We see that choosing $p_{model}(y|\mathbf{x}, \mathbf{w})$ to be Gaussian causes maximum likelihood to yield exactly the same least squares estimator derived before:

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \ \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$$

Remarks:

- **Consistency:** As the number of training samples approaches infinity $N \to \infty$, the maximum likelihood (ML) estimate converges to the true parameters
- **Efficiency:** The ML estimate converges most quickly as N increases
- These theoretical considerations make ML estimators appealing