Machine Learning with a 2020 Sight

People:

- Prof. Tali Tishby
- Prof. Amir Globerson
- Dr. Gal Elidan
- Dr. Ami Wiesel
- Dr. Ohad Shamir
- Prof. Nati Srebro
- Prof. Elad Hazan
- Dr. Koby Crammer
- Prof. Ran El-Yaniv
- Prof. Shie Manor
- Prof. Shai Shalev-Shwartz

Machine Learning with a 2020 Sight

Speakers:

- **Shai Shalev-Shwartz:** Deep Learning: Why and How ?
- **Ohad Shamir:** Silence is Golden: Distributed Learning with Minimal Communication
- **Ami Wiesel**: Distributed learning in networks
- **Koby Crammer:** Learning many tasks with a single teacher
- **Amir Globerson:** Holistic NLP Parsing via Sampling
- **Naftali Tishby:** The intriguing relations between power, time, and information in planning and learning - a lesson for computer architecture?
- **Shai Fine:** Intel Machine Learning View

Deep Learning: Why and How

Shai Shalev-Shwartz

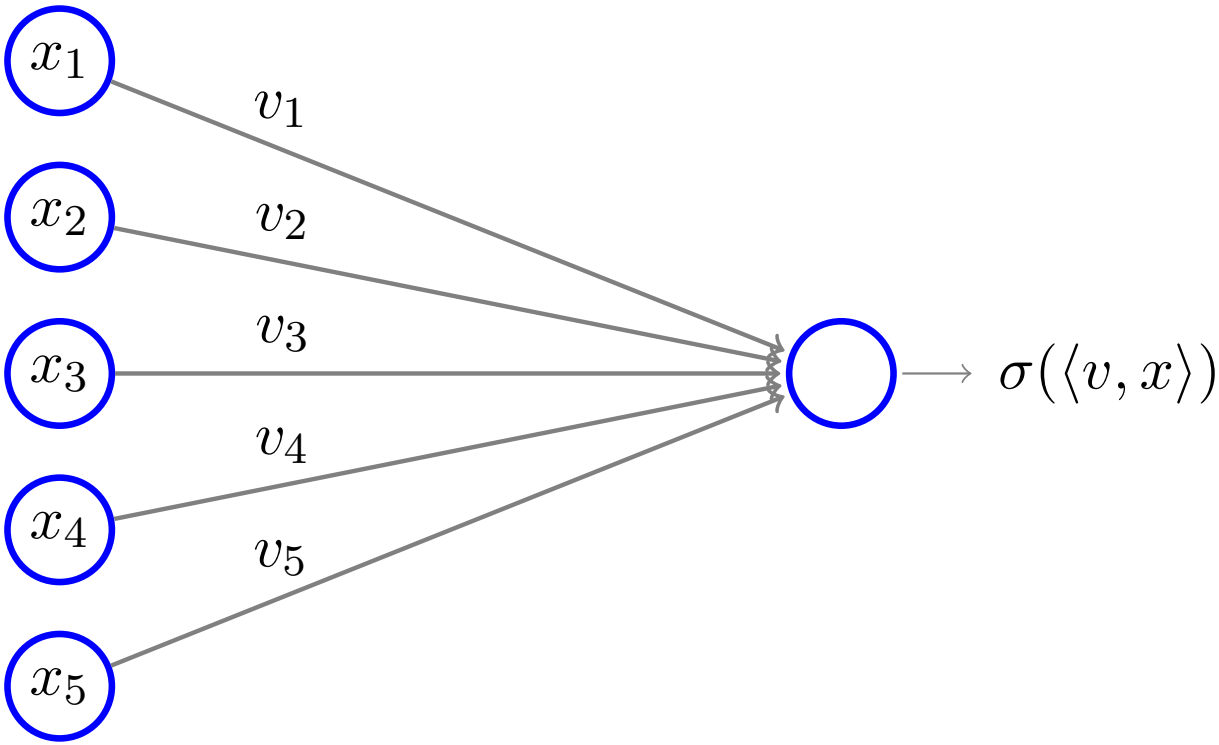
School of CS and Engineering,
The Hebrew University of Jerusalem

ICRI-CI retreat, May 2014

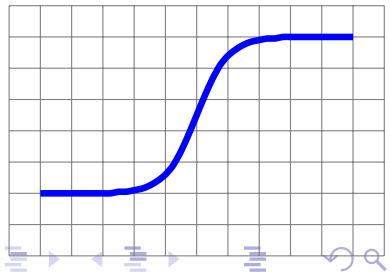
Joint work with **Roi Livni and Ohad Shamir**

Neural Networks

- A single neuron with activation function $\sigma : \mathbb{R} \to \mathbb{R}$

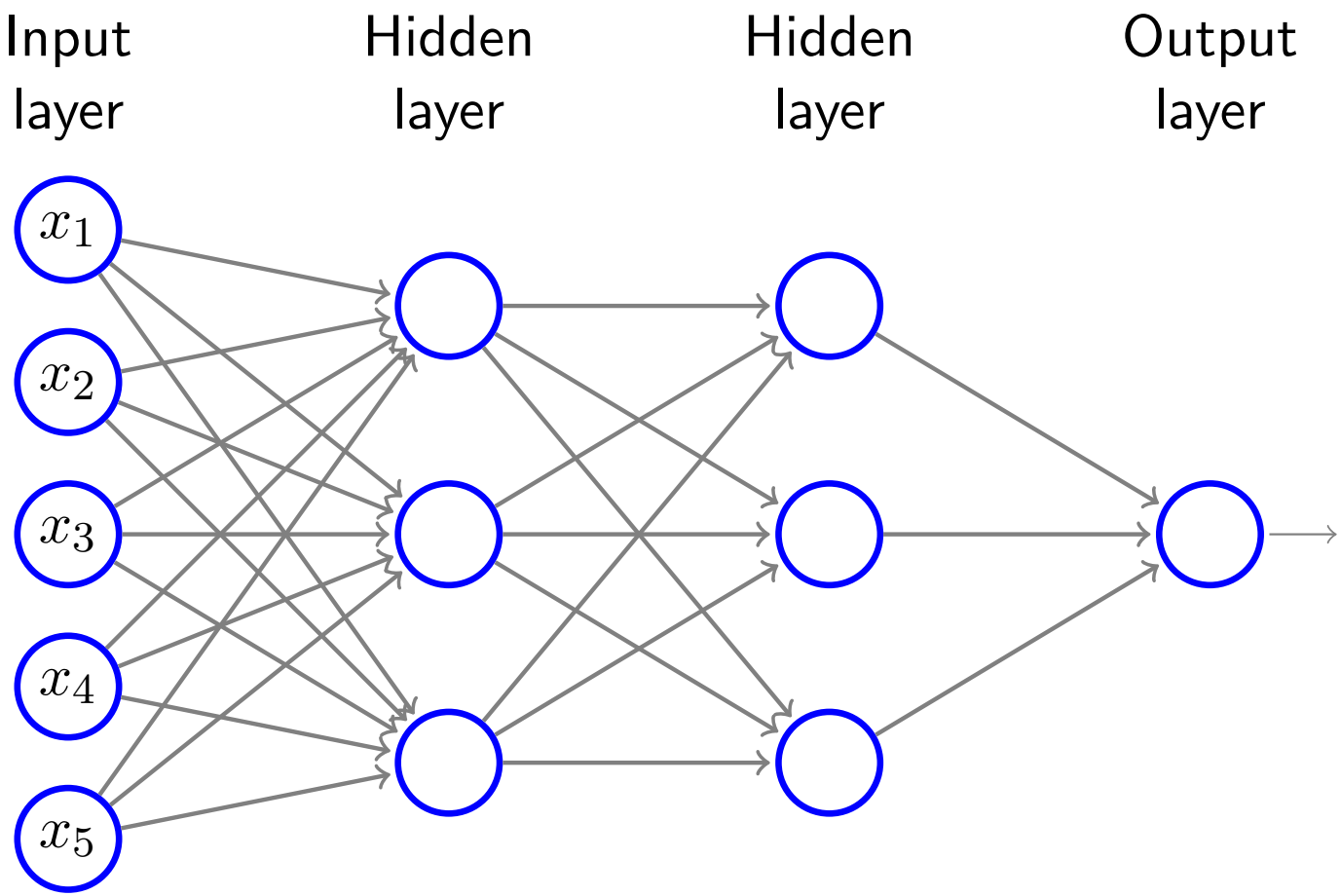


- E.g., σ is a sigmoidal function



Neural Networks

- A multilayer neural network of depth 3 and size 6



Brief history

- Neural networks are not new
- They were popular in the 70's and 80's
- Then, suppressed by SVM and Adaboost on the 90's
- In 2006, several deep architectures with unsupervised pre-training have been proposed
- In 2012, Krizhevsky, Sutskever, and Hinton significantly improved state-of-the-art without unsupervised pre-training

Why and How ?

Two immediate questions:

- Why should I prefer neural networks over other machine learning algorithms ? What is so special about them ?
- How should I train a neural network ?

Why Deep Neural Networks are Great?

- Because "A" used it to do "B".

Why Deep Neural Networks are Great?

- Because "A" used it to do "B". E.g.
  - {Krizhevsky, Sutskever, Hinton} won the imagenet challenge
  - {Taigman, Yang, Ranzato, Wolf} closed the gap to human-level performance in face recognition

Why Deep Neural Networks are Great?

- Because "A" used it to do "B". E.g.
  - {Krizhevsky, Sutskever, Hinton} won the imagenet challenge
  - {Taigman, Yang, Ranzato, Wolf} closed the gap to human-level performance in face recognition
- Classic explanation: Neural Networks are *universal approximators* — every Lipschitz function $f : [-1, 1]^d \to [-1, 1]$ can be approximated by a neural network

Why Deep Neural Networks are Great?

- Because "A" used it to do "B". E.g.
  - {Krizhevsky, Sutskever, Hinton} won the imagenet challenge
  - {Taigman, Yang, Ranzato, Wolf} closed the gap to human-level performance in face recognition
- Classic explanation: Neural Networks are *universal approximators* — every Lipschitz function $f : [-1, 1]^d \to [-1, 1]$ can be approximated by a neural network
- Not convincing because
  - It can be shown that the size of the network must be exponential in d, so why should we care about such large networks ?
  - Many other universal approximators exist (nearest neighbor, boosting with decision stumps, SVM with RBF kernels), so why should we prefer neural networks?

Why Deep Neural Networks are Great?
A Statistical Learning Perspective

- Goal: Learn a predictor $h : \mathcal{X} \to \mathcal{Y}$ (E.g., $\mathcal{X}$ is images, $\mathcal{Y}$ is 1000 possible categories) based on training examples

Why Deep Neural Networks are Great?
A Statistical Learning Perspective

- Goal: Learn a predictor $h : \mathcal{X} \to \mathcal{Y}$ (E.g., $\mathcal{X}$ is images, $\mathcal{Y}$ is 1000 possible categories) based on training examples
- No-Free-Lunch Theorem: For any learner A, exists "world" (formally defined by distribution and target function) s.t. A fails

Why Deep Neural Networks are Great? A Statistical Learning Perspective

- Goal: Learn a predictor $h : \mathcal{X} \to \mathcal{Y}$ (E.g., $\mathcal{X}$ is images, $\mathcal{Y}$ is 1000 possible categories) based on training examples
- No-Free-Lunch Theorem: For any learner A, exists "world" (formally defined by distribution and target function) s.t. A fails
- Prior knowledge: We must bias the learner toward "reasonable" functions — hypothesis class

Why Deep Neural Networks are Great? A Statistical Learning Perspective

- Goal: Learn a predictor $h : \mathcal{X} \to \mathcal{Y}$ (E.g., $\mathcal{X}$ is images, $\mathcal{Y}$ is 1000 possible categories) based on training examples
- No-Free-Lunch Theorem: For any learner A, exists "world" (formally defined by distribution and target function) s.t. A fails
- Prior knowledge: We must bias the learner toward "reasonable" functions — hypothesis class
- What should be the hypothesis class ?

Why Deep Neural Networks are Great? A Statistical Learning Perspective

- Goal: Learn a predictor $h : \mathcal{X} \to \mathcal{Y}$ (E.g., $\mathcal{X}$ is images, $\mathcal{Y}$ is 1000 possible categories) based on training examples
- No-Free-Lunch Theorem: For any learner A, exists "world" (formally defined by distribution and target function) s.t. A fails
- Prior knowledge: We must bias the learner toward "reasonable" functions — hypothesis class
- What should be the hypothesis class ?
- In SVM and Boosting, it is a composition of a linear predictor on top of features, and most of the practical work is on finding good features

Why Deep Neural Networks are Great?
A Statistical Learning Perspective

- But, consider a much weaker prior knowledge:
  We only care about predictors that can be implemented efficiently

Why Deep Neural Networks are Great? A Statistical Learning Perspective

- But, consider a much weaker prior knowledge:
  We only care about predictors that can be implemented efficiently

Theorem

The class of neural networks of depth $O(T(d))$ and size $O(T(d)^2)$ contains all functions that can be executed in time at most $T(d)$

Why Deep Neural Networks are Great?
A Statistical Learning Perspective

- But, consider a much weaker prior knowledge:
  We only care about predictors that can be implemented efficiently

Theorem

The class of neural networks of depth $O(T(d))$ and size $O(T(d)^2)$ contains all functions that can be executed in time at most $T(d)$

- A great hypothesis class:
  - With sufficiently large network depth and size, we can express all functions we would ever want to learn
  - Sample complexity behaves nicely and is well understood (see Anthony & Bartlett 1999)
- End of story ?

Neural Networks — The "How" Question

- It is NP hard to fit a neural network to data, even if it is produced exactly by a depth 2 network with $k \geq 3$ hidden neurons whose activation function is sigmoidal or sign (Blum and Rivest 1992, Bartlett and Ben-David 2002)
- Current approaches: Stochastic Gradient Descent (while using Back-propagation to calculate gradients), possibly with unsupervised pre-training, and other bells and whistles (initialization, momentum, dropout, ...)
- No theoretical guarantees:
  - How much time is required to converge ?
  - Local minima ?

How to circumvent hardness?

1. Over-specification

2. Change the activation function (polynomial networks)

Circumventing Hardness using Over-specification

- Yann LeCun:
  - Fix a network architecture and generate data according to it
  - Backpropagation fails to recover parameters
  - However, if we enlarge the network size, backpropagation works just fine

Circumventing Hardness using Over-specification

- Yann LeCun:
  - Fix a network architecture and generate data according to it
  - Backpropagation fails to recover parameters
  - However, if we enlarge the network size, backpropagation works just fine
  - Maybe we can efficiently learn neural network using over-specification?

Is over-specification enough ?

- Theorem (Livni, Shamir, S.): If number of neurons is larger than training set size then neural networks have no local (non-global) minima

Is over-specification enough ?

- Theorem (Livni, Shamir, S.): If number of neurons is larger than training set size then neural networks have no local (non-global) minima
- But, such large networks will lead to overfitting
- Maybe there's a clever trick that circumvent overfitting (regularization, dropout, ...) ?

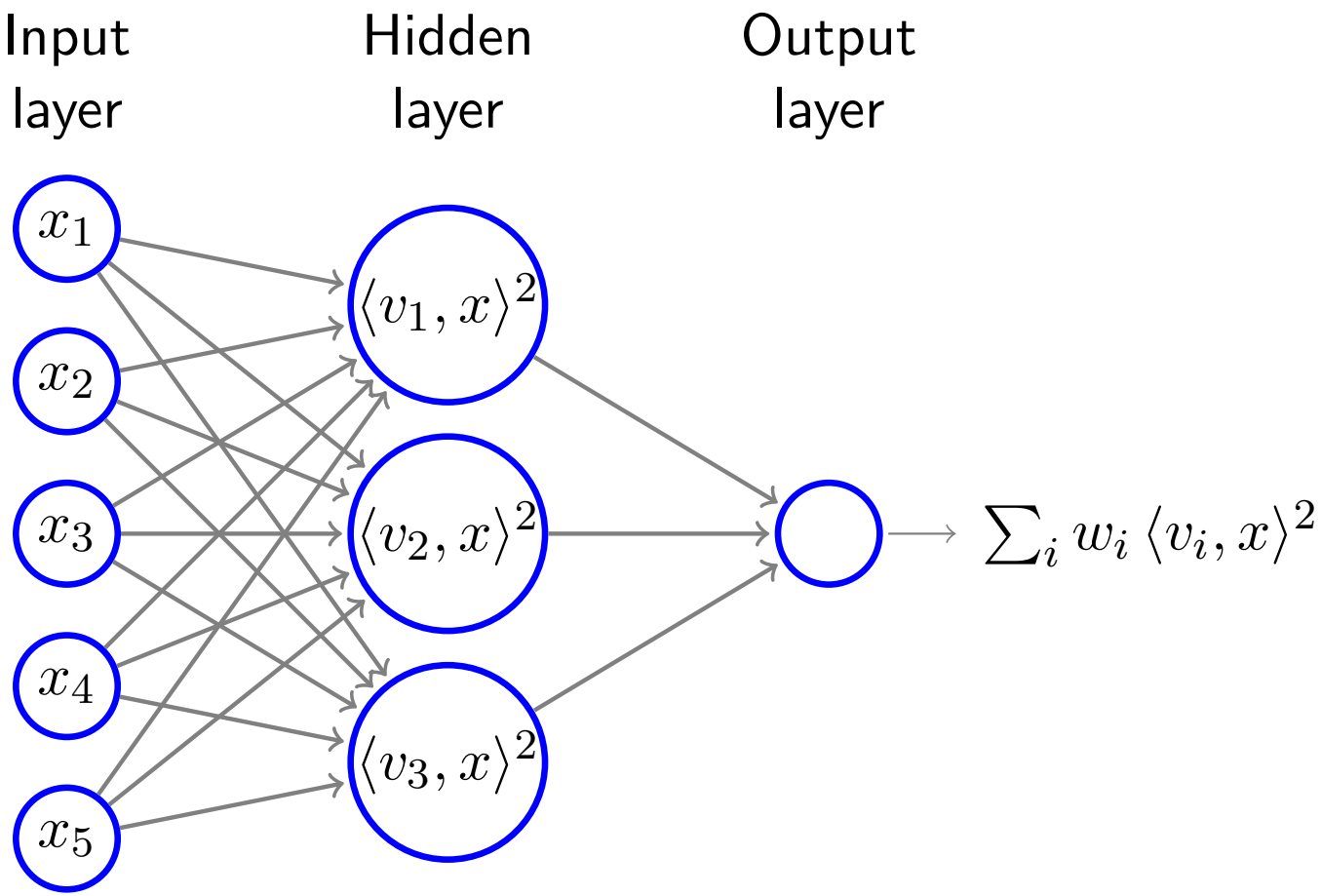
Is over-specification enough ?

- Theorem (Livni, Shamir, S.): If number of neurons is larger than training set size then neural networks have no local (non-global) minima
- But, such large networks will lead to overfitting
- Maybe there's a clever trick that circumvent overfitting (regularization, dropout, ...) ?
- Theorem (Daniely, Linial, S.) Even if the data is perfectly generated by a neural network of depth 2 and with only few neurons in the hidden layer, there is no algorithm that can achieve small test error
- Corollary: over-specification alone is not enough for efficient learnability

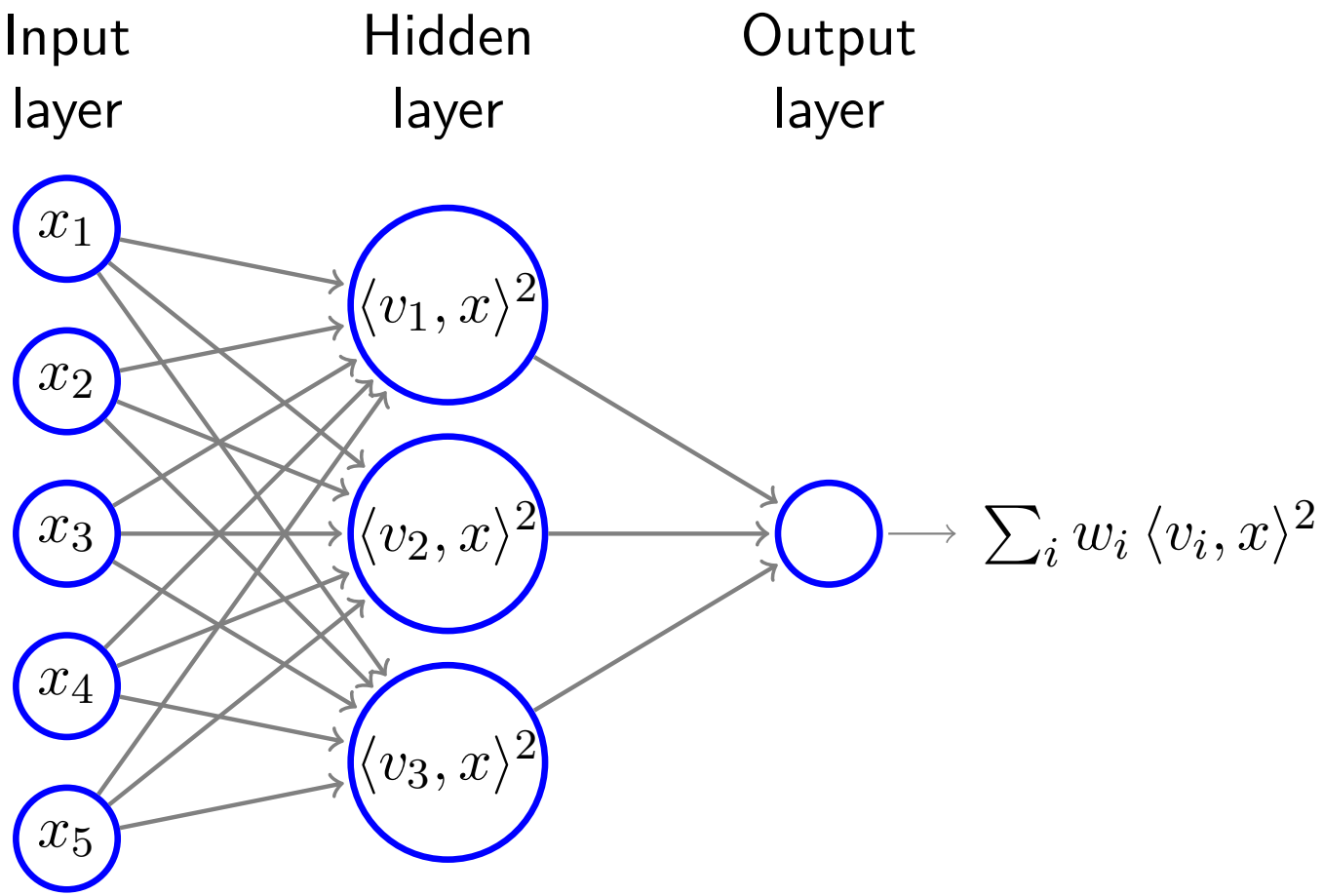
Circumventing hardness — polynomial networks

- Simpler non-linearity — replace sigmoidal activation function by the square function $\sigma(a) = a^2$
- Network implements polynomials, where the depth corresponds to degree
- The size of the network (number of neurons) determines generalization properties and evaluation time
- Can we efficiently learn the class of polynomial networks of small size?

Depth 2 polynomial network



Depth 2 polynomial network



- Finding the best fit to data is still NP hard
- But, here, a slight over-specification works !

Forward Greedy Selection for Polynomial Networks

- Let $\mathcal{S}$ be the Euclidean sphere of $\mathbb{R}^d$
- We show a correspondence between two layer polynomial networks and all mappings from $\mathcal{S}$ to $\mathbb{R}$ with sparse support
- Apply forward greedy selection for learning the sparse mapping
- Main caveat: at each greedy iteration we need to find v that approximately solve

$$\operatorname*{argmax}_{v \in \mathcal{S}} |\nabla_v R(w)|$$

- Luckily, this is an eigenvalue problem

$$\nabla_v R(w) = v^\top \left(\mathop{\mathbb{E}}_{(x,y)} \ell' \left(\sum_{u \in \mathrm{supp}(w)} w_u \langle u, x \rangle^2, y \right) x x^\top \right) v$$

The resulting algorithm

Greedy Efficient Component Optimization (GECO):

- Initialize $V = [\,]$, $w = []$
- For $t = 1, 2, \ldots, T$
  - Let $M = \mathbb{E}_{(x,y)} \ell'(\sum_i w_i (\langle v_i, x \rangle)^2, y) x x^\top$
  - $V = [V \; v]$ where v is an approximate leading eigenvector of M
  - Let $B = \operatorname{argmin}_{B \in \mathbb{R}^{t,t}} \mathbb{E}_{(x,y)} \ell((x^\top V) B (V^\top x), y)$
  - Update $w = \text{eigenvalues}(B)$ and $V = V \text{eigenvectors}(B)$

The resulting algorithm

Greedy Efficient Component Optimization (GECO):

- Initialize $V = [\,]$, $w = []$
- For $t = 1, 2, \ldots, T$
  - Let $M = \mathbb{E}_{(x,y)}\, \ell'(\sum_i w_i (\langle v_i, x \rangle)^2, y) x x^\top$
  - $V = [V\ v]$ where v is an approximate leading eigenvector of M
  - Let $B = \operatorname{argmin}_{B \in \mathbb{R}^{t,t}} \mathbb{E}_{(x,y)}\, \ell((x^\top V) B (V^\top x), y)$
  - Update $w = \text{eigenvalues}(B)$ and $V = V \text{eigenvectors}(B)$

Remarks:

- Finding an approximate leading eigenvector takes linear time
- Overall complexity depends linearly on the size of the data

Deeper polynomial networks ?

- Learning depth 2 sigmoidal networks is hard even if we allow over-specification
- Learning depth 2 polynomial networks is tractable if we allow slight over-specification
- Using a more sophisticated algorithm employing tensor factorization we can also learn depth 3 polynomial networks with over-specification
- On the other hand, we show that learning very deep polynomial networks is in-tractable

Extension to Convolutional Networks

- Can efficiently learn one convolutional-pooling layer using similar ideas
- Preliminary experimental results are quite promising

Summary

- Why deep networks: Deep networks are the ultimate hypothesis class from the statistical perspective
- Why not: Deep networks are a horrible class from the computational point of view
- This work: Networks of small depth and "squared" activation can be trained efficiently using greedy selection and over-specification
- Very deep polynomial networks cannot be trained efficiently

Main open problems

- Why Stochastic Gradient Descent works so good ???
- Find a combination of network architecture and distributional assumptions that are useful in practice and lead to efficient algorithms

Shameless Advertisement