Stat 212b: Topics in Deep Learning
Lecture 22

Joan Bruna
UC Berkeley

Review: Optimization in ML

- Online stochastic optimization adapts well to the needs of large-scale ML optimization.
  - Interplay of generalization, approximation and optimization error [Bottou & Bousquet]

- First order methods can be accelerated by incorporating momentum term.

From rate $O(1/t)$ to rate $O(1/t^2)$ for smooth, convex problems

From rate $O((1 - c\kappa^{-1})^t)$ to rate $O((1 - c\kappa^{-1/2})^t)$ for smooth, strongly convex problems.

Minimax optimal rates in the class of smooth, convex (resp. strongly convex) class for first order methods.

Generalization Error

- Recall

$$\Phi^* = \arg\min_{\Phi} F(\Phi) \ , \ \text{optimal model} \ ,$$

$$\Phi^*_{\mathcal{F}} = \arg\min_{\Phi \in \mathcal{F}} F(\Phi) \ , \ \text{optimal achievable model in } \mathcal{F} \ ,$$

$$\Phi_{\mathcal{F},n} = \arg\min_{\Phi \in \mathcal{F}} \hat{F}_n(\Phi) \ , \ \text{optimal empirical model in } \mathcal{F} \ ,$$

$$\widetilde{\Phi}_{\mathcal{F},n} = \ \text{solution of our optimization of } \min_{\Phi \in \mathcal{F}} \hat{F}_n(\Phi) \ ,$$

- With

$$F(\Phi) = \mathbb{E}_{z \sim \pi} f(z; \Phi) \ . \qquad \hat{F}_n(\Phi) = \frac{1}{n} \sum_{i \leq n} f(z_i; \Phi) \ .$$

- Q: How to modify our optimization in order to improve generalization error?

Review: Tikhonov Regularization

- Suppose we have the following inverse linear problem

$$\min_x \|y - Ax\|^2 \ , \ A \in \mathbb{R}^{n \times p} \ .$$

- When $p \leq \mathrm{rank}(A)$, the system has unique solution

$$A^T(y - Ax) = 0 \Rightarrow x^* = (A^T A)^{-1} A^T y = A^\dagger y \ .$$

$A^\dagger = (A^T A)^{-1} A^T$: Moore-Penrose pseudoinverse of A.

- When $p > \mathrm{rank}(A)$, under-determined system.
  Which solution to select?

  Tikhonov proposed selecting the solution x^* having smallest norm $\|\Gamma^{1/2} x\|$:

$$\min_{Ax=y} \langle x, \Gamma x \rangle \ , \ \Gamma : \text{Tikhonov psd kernel.}$$

Review: Tikhonov Regularization

- Limitations?
  - Minimizing the L2 norm tends to spread out the weights. Lack of sparsity in our predictions.
  - In image applications, this tends to produce blurred estimates.
  - We can regularize using different priors that favor sparsity (e.g. Lasso).

  - In machine learning, some models work better with L1 regularization (e.g. Logistic Regression, [Ng,'04]).

Review: Algorithmic Stability vs Generalization

[Bousquet, Eliseff], [Hardt, Recht, Singer]

- We can interpret generalization as a form of stability of our learning protocol.
- Expected Generalization error:

$$\epsilon_{gen} = \mathbb{E}_{S,A}[F_n(\Phi(A,S)) - F(\Phi(A,S))] \ ,$$

A : (randomized) algorithm
S : (random) sample

- Stability of a randomized algorithm:

A randomized algorithm A is ϵ-uniformly stable if for all datasets S,S' differing in at most one sample we have

$$\sup_z \mathbb{E}_A[f(\Phi(A(S)); z) - f(\Phi(A(S')); z)] \leq \epsilon \ .$$

Review: Dropout [Hinton'12]

- The ridge regression replaced the empirical data covariance $X^T X$ by $X^T X + \lambda I$.
  - This is equivalent as replacing data x_i by

$$\tilde{x}_{i,j} = x_i + \epsilon_{i,j} \ , \ \mathbb{E}\epsilon_{i,j} = 0, \mathrm{cov}(\epsilon_{i,j}) = \lambda I \ .$$

$$\text{as } j \to \infty.$$

Review: Dropout [Hinton'12]

- The ridge regression replaced the empirical data covariance $X^T X$ by $X^T X + \lambda I$.
  - This is equivalent as replacing data x_i by
    $$\tilde{x}_{i,j} = x_i + \epsilon_{i,j} \ , \ \mathbb{E}\epsilon_{i,j} = 0, \mathrm{cov}(\epsilon_{i,j}) = \lambda I \ .$$
    as $j \to \infty$.

- Indeed,

$$\frac{1}{N} \sum_{i \leq N} \frac{1}{J} \sum_{j \leq J} (y_i - \langle \tilde{x}_{i,j}, \beta \rangle)^2 \overset{J \to \infty}{\to} \frac{1}{N} \sum_{i \leq N} \mathbb{E}_{\epsilon} (y_i - \langle x_i, \beta \rangle - \langle \epsilon_{i,j}, \beta \rangle)^2$$

$$= \frac{1}{N} \sum_{i \leq N} (y_i - \langle x_i, \beta \rangle)^2 + \lambda \|\beta\|^2 = \|Y - X\beta\|^2 + \lambda \|\beta\|^2 \ .$$

Review: Dropout [Hinton'12]

- The ridge regression replaced the empirical data covariance $X^T X$ by $X^T X + \lambda I$.
  - This is equivalent as replacing data x_i by
    $$\tilde{x}_{i,j} = x_i + \epsilon_{i,j} \ , \ \mathbb{E}\epsilon_{i,j} = 0, \mathrm{cov}(\epsilon_{i,j}) = \lambda I \ .$$
    $$\text{as } j \to \infty.$$
- Indeed,

$$\frac{1}{N}\sum_{i\leq N}\frac{1}{J}\sum_{j\leq J}(y_i - \langle \tilde{x}_{i,j}, \beta\rangle)^2 \overset{J\to\infty}{\to} \frac{1}{N}\sum_{i\leq N}\mathbb{E}_{\epsilon}(y_i - \langle x_i, \beta\rangle - \langle \epsilon_{i,j}, \beta\rangle)^2$$
$$= \frac{1}{N}\sum_{i\leq N}(y_i - \langle x_i, \beta\rangle)^2 + \lambda\|\beta\|^2 = \|Y - X\beta\|^2 + \lambda\|\beta\|^2 \ .$$

- Q: to what extent one can regularize by adding noise to the input? what noise distributions are appropriate?

Review: Dropout [Hinton et al.'12]

- Given a deep model

$$\Phi(x;\Theta) = \phi_K(\phi_{K-1}(\dots\phi_1(X;\Theta_1);\Theta_2)\dots;\Theta_K)$$

- we consider the following noise distribution

$$\tilde{\Phi}(x;\Theta) = \phi_K(b_{K-1}\cdot\phi_{K-1}(\dots(b_1\cdot\phi_1(b_0\cdot X;\Theta_1);\Theta_2)\dots;\Theta_K)~,$$

$$b_0,\dots,b_{K-1}~\text{Bernoulli}~p~.$$

Review: Dropout [Hinton et al.'12]

- Given a deep model

$$\Phi(x;\Theta) = \phi_K(\phi_{K-1}(\dots \phi_1(X;\Theta_1);\Theta_2)\dots;\Theta_K)$$

- we consider the following noise distribution

$$\tilde{\Phi}(x;\Theta) = \phi_K(b_{K-1}\cdot\phi_{K-1}(\dots(b_1\cdot\phi_1(b_0\cdot X;\Theta_1);\Theta_2)\dots;\Theta_K)~,$$

$$b_0,\dots,b_{K-1} \text{ Bernoulli } p~.$$

- At test time, we approximate $\mathbb{E}_b\tilde{\Phi}(x;\Theta)$ with $\Phi(x;p\Theta)$.
- Typically, we choose $p=0.5$.
- Very robust, very efficient.
- Not clear why (yet).

Dropout and Ensemble Methods

- Dropout performs a form of exponential ensemble of tiny networks.
  - Let $M = \sum_{k=1}^{K} \dim(\Theta_k)$ be the total number of weights.
  - For each given training sample, on average we have pM active weights. Number of different configurations is $\sim \binom{M}{pM}$
  - At test time, we approximate the committee of these smaller networks.
  - Hinton argues that this fights feature "co-adaptation": relying on spurious, unreliable high-order dependencies within the data.

Dropout and Adaptive Regularization

- [Wager et al'13] performed the first rigorous analysis of Dropout in the context of Generalized Linear Models:

Dropout and Adaptive Regularization

- [Wager et al'13] performed the first rigorous analysis of Dropout in the context of Generalized Linear Models:

Suppose response y given input features $x \in \mathbb{R}^d$

$$p(y|x,\beta) = p_0(y)\exp\left(y\langle x,\beta\rangle - A(x,\beta)\right)\ ,\ \ell(\beta) = -\log p(y|x,\beta)\ .$$

Standard MLE $\hat{\beta}$: $$\hat{\beta} = \arg\min_{\beta} \sum_i \ell_{x_i,y_i}(\beta)\ .$$

Dropout and Adaptive Regularization

- [Wager et al'13] performed the first rigorous analysis of Dropout in the context of Generalized Linear Models:

Suppose response y given input features $x \in \mathbb{R}^d$

$$p(y|x,\beta) = p_0(y)\exp\left(y\langle x,\beta\rangle - A(x,\beta)\right)~,~\ell(\beta) = -\log p(y|x,\beta)~.$$

Standard MLE $\hat{\beta}$: $$\hat{\beta} = \arg\min_{\beta}\sum_i \ell_{x_i,y_i}(\beta)~.$$

Noisy features: $\tilde{x}_i = \nu(x_i,\xi_i)$.

Regularized MLE estimation:

$$\hat{\beta} = \arg\min_{\beta}\sum_i \mathbb{E}_{\xi}\ell_{\nu(x_i,\xi),y_i}(\beta)~.$$

Dropout and Adaptive Regularization

- [Wager et al'13] performed the first rigorous analysis of Dropout in the context of Generalized Linear Models:

Suppose response y given input features $x \in \mathbb{R}^d$

$$p(y|x,\beta) = p_0(y) \exp\left(y\langle x, \beta\rangle - A(x,\beta)\right) \ , \ \ell(\beta) = -\log p(y|x,\beta) \ .$$

Standard MLE $\hat{\beta}$: $$\hat{\beta} = \arg\min_{\beta} \sum_i \ell_{x_i,y_i}(\beta) \ .$$

Noisy features: $\tilde{x}_i = \nu(x_i, \xi_i)$.

Regularized MLE estimation:

$$\hat{\beta} = \arg\min_{\beta} \sum_i \mathbb{E}_{\xi} \ell_{\nu(x_i,\xi),y_i}(\beta) \ .$$

- The latter can be rewritten as

$$\sum_i \ell_{x_i,y_i}(\beta) + R(\beta) \ , \ \text{with } R(\beta) = \sum_i \mathbb{E}_{\xi} A(\tilde{x}_i, \beta) - A(x_i, \beta)$$

Dropout and Adaptive Regularization

- Taylor approximation of a non-linear moment:

$$\begin{aligned}\mathbb{E}f(X) &= f(\mathbb{E}X) + f'(\mathbb{E}X)\mathbb{E}(X-\mathbb{E}X) + \frac{1}{2}f''(\mathbb{E}X)\mathbb{E}(X-\mathbb{E}X)^2 + O(|f'''|_\infty)\mu_3(X) \\ &\approx f(\mathbb{E}X) + \frac{1}{2}f''(\mathbb{E}X)\mathrm{var}(X)\ .\end{aligned}$$

Dropout and Adaptive Regularization

- Taylor approximation of a non-linear moment:

$$\mathbb{E}f(X) = f(\mathbb{E}X) + f'(\mathbb{E}X)\mathbb{E}(X - \mathbb{E}X) + \frac{1}{2}f''(\mathbb{E}X)\mathbb{E}(X - \mathbb{E}X)^2 + O(|f'''|_\infty)\mu_3(X)$$
$$\approx f(\mathbb{E}X) + \frac{1}{2}f''(\mathbb{E}X)\mathrm{var}(X) \ .$$

- Applying it to R, the authors show that dropout noise performs *adaptive regularization:*

$$R(\beta) \approx \beta^T \mathrm{diag}(X^T V(\beta) X)\beta \ ,$$
$$X^T V(\beta) X \ : \text{ Fisher information} \quad V(\beta)_{i,i} = A''(\langle x_i, \beta \rangle) \ .$$

Dropout and Adaptive Regularization

- Taylor approximation of a non-linear moment:

$$\mathbb{E}f(X) = f(\mathbb{E}X) + f'(\mathbb{E}X)\mathbb{E}(X - \mathbb{E}X) + \frac{1}{2}f''(\mathbb{E}X)\mathbb{E}(X - \mathbb{E}X)^2 + O(|f'''|_\infty)\mu_3(X)$$

$$\approx f(\mathbb{E}X) + \frac{1}{2}f''(\mathbb{E}X)\mathrm{var}(X)\ .$$

- Applying it to R, the authors show that dropout noise performs *adaptive regularization:*

$$R(\beta) \approx \beta^T \mathrm{diag}(X^T V(\beta) X)\beta\ ,$$

$$X^T V(\beta) X\ :\ \text{Fisher information} \quad V(\beta)_{i,i} = A''(\langle x_i, \beta\rangle)\ .$$

- In logistic regression, this becomes

$$R(\beta) \approx \frac{\delta}{2(1-\delta)} \sum_{i,j} p_i(1-p_i)x_{i,j}^2\beta_j^2\ .$$

- In contrast to additive noise

$$R(\beta) \approx \frac{1}{2}\sigma^2\|\beta\|^2 \sum_i p_i(1-p_i)\ .$$

(some) Dropout open questions

- Analysis for deep networks
  - Statistical dependency not only on input distribution but also on parameters that we are learning

- Dropout vs Dropconnect vs Structured Dropconnect
  - Activate/desactivate weights rather than neurons. Why/ How?

- Relationship with Bootstrap
  - Can we use dropout to construct confidence intervals of network predictions?

Objectives

- Batch Normalization

- Tensor Methods in Deep Learning
  - [Cohen, Sharir, Shashua]
  - [Haeffele & Vidal]
  - [Janzamin, Sedghi, Anandkumar]

- …and beyond [next week, time permitting].
  - Spin glasses and deep networks: [Choromaska et al],[Chaudhari et al]
  - Alternative to gradient descent? [Zhang, Lee, Wainwright, Jordan]

Conditioning in Gradient Descent

- Suppose we want to learn a function $\Phi(x;\Theta)$ using gradient descent with respect to Θ, e.g.

$$f(\Theta) = \|Y - \Phi(X;\Theta)\|^2 = \|Y - X\Theta\|^2 \ .$$

- We saw that gradient descent is sensitive to the conditioning of the problem:

$$l\|x - y\| \leq \|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$$

Conditioning in Gradient Descent

- Suppose we want to learn a function $\Phi(x;\Theta)$ using gradient descent with respect to Θ, e.g.

$$f(\Theta) = \|Y - \Phi(X;\Theta)\|^2 = \|Y - X\Theta\|^2 \ .$$

- We saw that gradient descent is sensitive to the conditioning of the problem:

$$l\|x - y\| \leq \|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$$

- It results that

$$\nabla f(\Theta) - \nabla f(\Theta') = X^T X(\Theta - \Theta')$$

Conditioning in Gradient Descent

- Suppose we want to learn a function $\Phi(x;\Theta)$ using gradient descent with respect to Θ, e.g.

$$f(\Theta) = \|Y - \Phi(X;\Theta)\|^2 = \|Y - X\Theta\|^2 \ .$$

- We saw that gradient descent is sensitive to the conditioning of the problem:

$$l\|x-y\| \leq \|\nabla f(x) - \nabla f(y)\| \leq L\|x-y\|$$

- It results that

$$\nabla f(\Theta) - \nabla f(\Theta') = X^T X(\Theta - \Theta')$$

- Thus, we may improve the conditioning by *whitening*

$$\mu = \mathbb{E}(X) \ , \ \Sigma = \mathbb{E}\{(X-\mu)(X-\mu)^T\} \ ,$$

$$\tilde{X} = \Sigma^{-1/2}(X-\mu) \ .$$

Conditioning in Gradient Descent

- Suppose we want to learn a function $\Phi(x; \Theta)$ using gradient descent with respect to Θ, e.g.

$$f(\Theta) = \|Y - \Phi(X; \Theta)\|^2 = \|Y - X\Theta\|^2 \ .$$

- We saw that gradient descent is sensitive to the conditioning of the problem:

$$l\|x - y\| \leq \|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$$

- It results that

$$\nabla f(\Theta) - \nabla f(\Theta') = X^T X(\Theta - \Theta')$$

- More generally, gradient descent can be adaptively conditioned, e.g. using *Adagrad* [Duchi et al.]
  - Learning rates are adjusted per feature.

The non-stationary learning problem

- Suppose now a two-layer model

$$f(\Theta_1, \Theta_2) = \|Y - \Theta_2(\rho(\Theta_1 X))\|^2 \ .$$

The non-stationary learning problem

- Suppose now a two-layer model

$$f(\Theta_1, \Theta_2) = \|Y - \Theta_2(\rho(\Theta_1 X))\|^2 \ .$$

- By denoting $\tilde{X} = \rho(\Theta_1 X)$, good conditioning for Θ_2 now would require to whiten $\tilde{X}$.

The non-stationary learning problem

- Suppose now a two-layer model

$$f(\Theta_1, \Theta_2) = \|Y - \Theta_2(\rho(\Theta_1 X))\|^2 \ .$$

- By denoting $\tilde{X} = \rho(\Theta_1 X)$, good conditioning for Θ_2 now would require to whiten $\tilde{X}$.
- **Problem:** $\tilde{X}$ depends on Θ_1, thus its distribution is non-stationary as the learning of Θ_1 progresses.
  - Θ_1 Θ_2

The non-stationary learning problem

- Suppose now a two-layer model

$$f(\Theta_1, \Theta_2) = \|Y - \Theta_2(\rho(\Theta_1 X))\|^2 \ .$$

- By denoting $\tilde{X} = \rho(\Theta_1 X)$, good conditioning for Θ_2 now would require to whiten $\tilde{X}$.
- **Problem:** $\tilde{X}$ depends on Θ_1, thus its distribution is non-stationary as the learning of Θ_1 progresses.
- Note that the role of Θ_1 and Θ_2 is not symmetric in the learning (sequential learning).
  - Θ_1 affects $\nabla\Theta_2$ through $\mathrm{cov}(\Theta_1 X)$.
  - Θ_2 affects $\nabla\Theta_1$ through $\mathrm{cov}\{\Theta_2\rho(\Theta_1 X), \Theta_2 \mathrm{diag}(\rho'(\Theta_1 X))X\}$.

The non-stationary learning problem

- This is an instance of *covariate shift* [Shimodaira,'00].

Training set: $\{x_i, y_i\}$ with $x_i \sim q_0(x)$.
Model: $p(y|x,\theta)$ trained as

$$\theta^* = \arg\min_\theta \sum_{x_i \sim q_0(x)} -\log p(y_i|x_i,\theta) \ ,$$

But tested with

$$\mathbb{E}_{x \sim q_1} - \log p(y|x,\theta^*) \ , \ \text{with } q_1 \neq q_0 \ .$$

- So the training estimator is biased.

The non-stationary learning problem

- This is an instance of *covariate shift* [Shimodaira,'00].

Training set: $\{x_i, y_i\}$ with $x_i \sim q_0(x)$.
Model: $p(y|x,\theta)$ trained as

$$\theta^* = \arg\min_\theta \sum_{x_i \sim q_0(x)} -\log p(y_i|x_i,\theta) \ ,$$

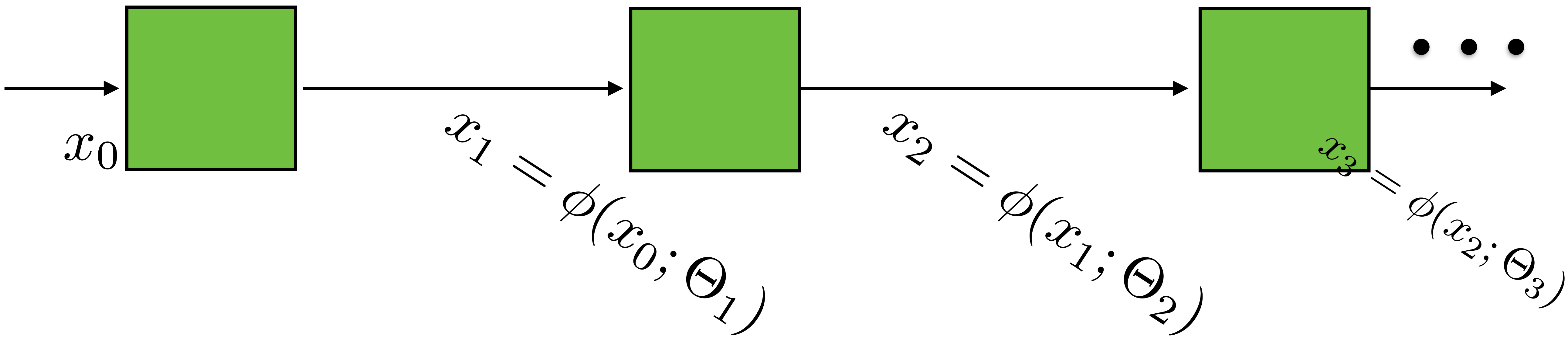
But tested with

$$\mathbb{E}_{x\sim q_1} - \log p(y|x,\theta^*) \ , \ \text{with } q_1 \neq q_0 \ .$$

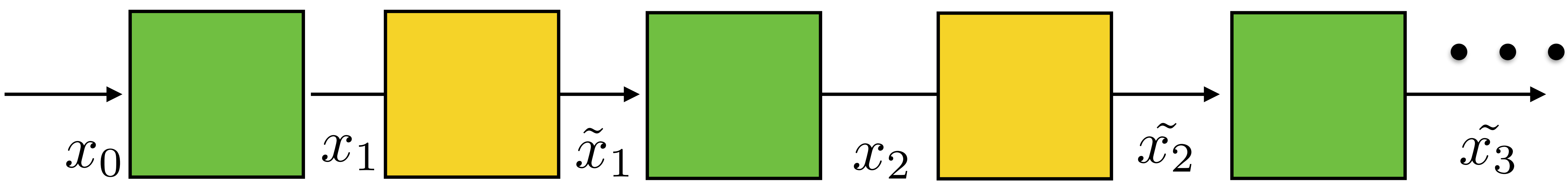
- So the training estimator is biased.

- Q: How to compensate for this effect?
- Q: How to apply it to the setting of deep networks? Coupling of parameters

Batch Normalization [Ioffe & Szegedy]



Batch Normalization [Ioffe & Szegedy]



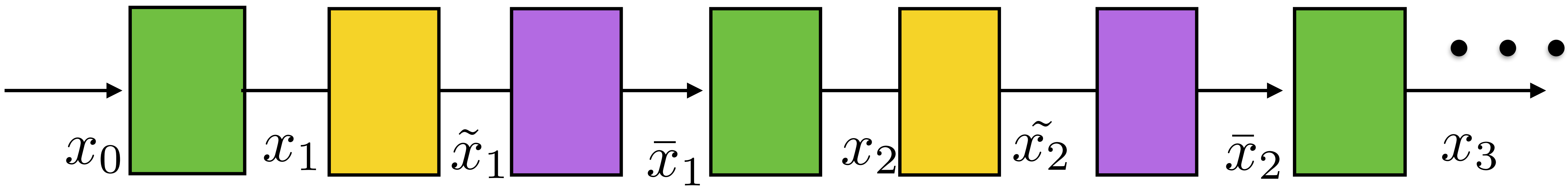
$$x_1 = \phi(x_0; \Theta_1)$$
$$\tilde{x}_1 = \hat{\sigma_1}^{-1} \odot (x_1 - \hat{\mu_1})$$

$$x_2 = \phi(\tilde{x}_1; \Theta_2)$$
$$\tilde{x}_2 = \hat{\sigma_2}^{-1} \odot (x_2 - \hat{\mu_2})$$

$$x_3 = \phi(\hat{x}_2; \Theta_3)$$

- Idea 1: Standardize the output of each layer to mitigate ill-conditioning.
- Idea 2: Do it *continuously* during training to avoid "internal covariate shift".

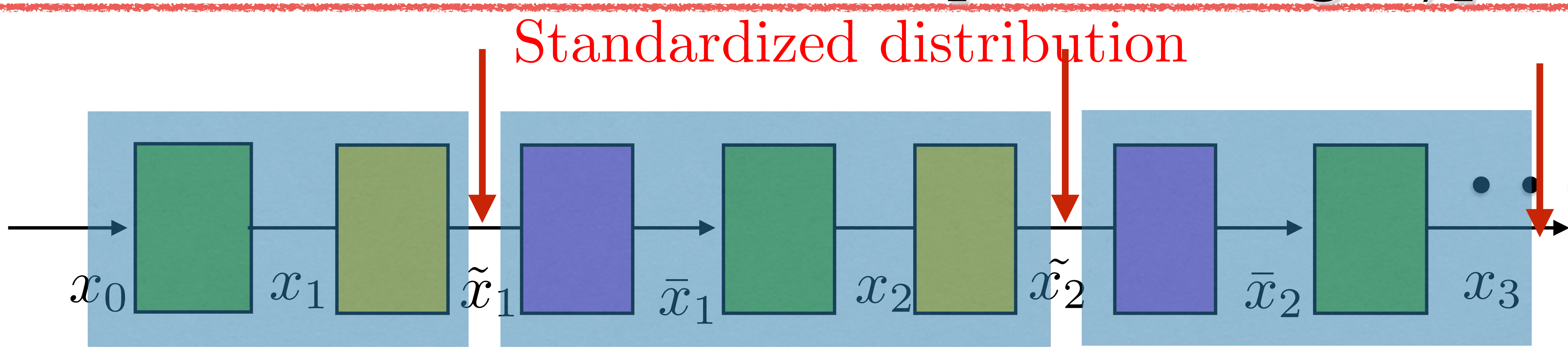
Batch Normalization [Ioffe & Szegedy]



$$
\begin{aligned}
x_1 &= \phi(x_0; \Theta_1) \\
\tilde{x}_1 &= \hat{\sigma_1}^{-1} \odot (x_1 - \hat{\mu_1}) \\
\bar{x}_1 &= \lambda_1 \odot (\tilde{x}_1 + \beta_1)
\end{aligned}
\qquad
\begin{aligned}
x_2 &= \phi(\bar{x}_1; \Theta_2) \\
\tilde{x}_2 &= \hat{\sigma_2}^{-1} \odot (x_2 - \hat{\mu_2}) \\
\bar{x}_2 &= \lambda_2 \odot (\tilde{x}_2 + \beta_2)
\end{aligned}
\qquad
x_3 = \phi(\bar{x}_2; \Theta_3)
$$

- Idea 1: Standardize the output of each layer to mitigate ill-conditioning.
- Idea 2: Do it *continuously* during training to avoid "internal covariate shift".
- Idea 3: Restore the first two moments with explicit linear layers.

Batch Normalization [Ioffe & Szegedy]



$$x_1 = \phi(x_0; \Theta_1) \qquad x_2 = \phi(\bar{x}_1; \Theta_2) \qquad x_3 = \phi(\bar{x}_2; \Theta_3)$$

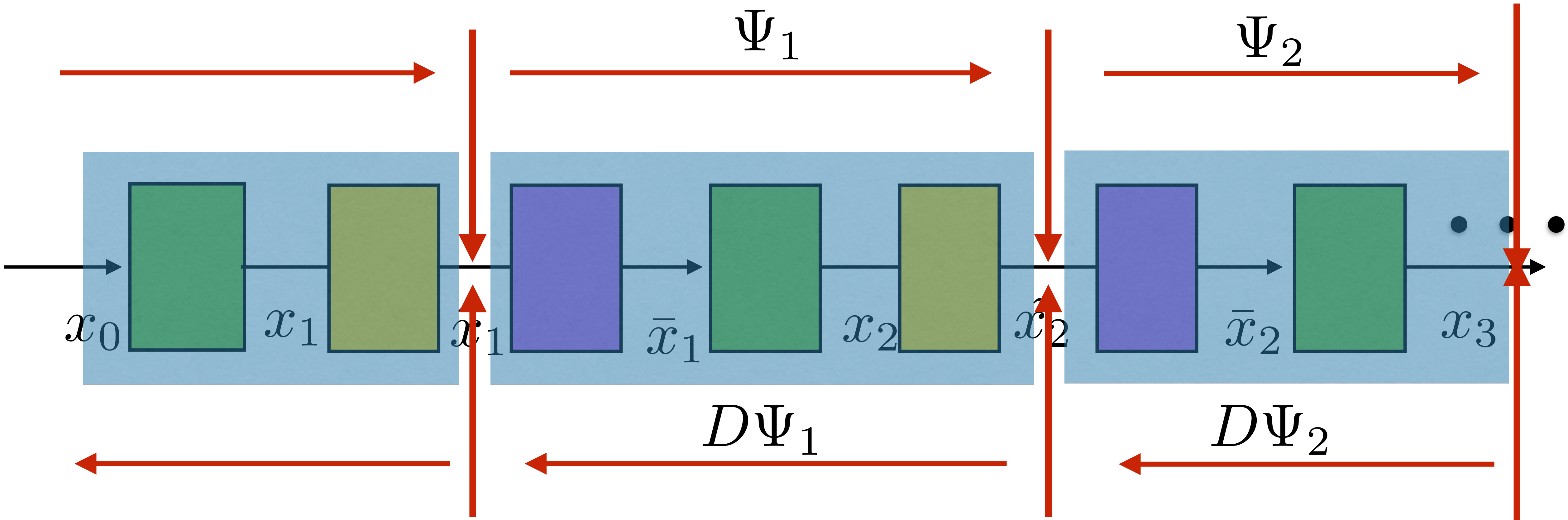
$$\tilde{x}_1 = \hat{\sigma}_1^{-1} \odot (x_1 - \hat{\mu}_1) \qquad \tilde{x}_2 = \hat{\sigma}_2^{-1} \odot (x_2 - \hat{\mu}_2)$$

$$\bar{x}_1 = \lambda_1 \odot (\tilde{x}_1 + \beta_1) \qquad \bar{x}_2 = \lambda_2 \odot (\tilde{x}_2 + \beta_2)$$

- <u>Idea 1</u>: Standardize the output of each layer to mitigate ill-conditioning.
- <u>Idea 2</u>: Do it *continuously* during training to avoid "internal covariate shift".
- <u>Idea 3</u>: Restore the first two moments with explicit linear layers.

Batch Normalization [Ioffe & Szegedy]

Standardized distribution



$$x_1 = \phi(x_0; \Theta_1)$$
$$\tilde{x}_1 = \hat{\sigma_1}^{-1} \odot (x_1 - \hat{\mu_1})$$
$$\bar{x}_1 = \lambda_1 \odot (\tilde{x}_1 + \beta_1)$$

$$x_2 = \phi(\bar{x}_1; \Theta_2)$$
$$\tilde{x}_2 = \hat{\sigma_2}^{-1} \odot (x_2 - \hat{\mu_2})$$
$$\bar{x}_2 = \lambda_2 \odot (\tilde{x}_2 + \beta_2)$$

$$x_3 = \phi(\bar{x}_2; \Theta_3)$$

- <u>Forward Pass:</u> Standardized by design.
- <u>Backward Pass:</u> Ψ_i maps standardized data.
  Jacobians $D\Psi_i$ might have better condition number (why?)

Batch Normalization

- Q: How to estimate mean and variance efficiently?

$$\hat{\mu} = \frac{1}{n}\sum_{i \leq n} x_{(i)} \ , \ \hat{\sigma^2} = \frac{1}{n-1}\sum_{i \leq n}(x_{(i)} - \hat{\mu})^2 \ .$$

Empirical average over whole training unfeasible

Instead, we consider estimations using *minibatches* of m examp

$$\hat{\mu}_b = \frac{1}{m}\sum_{i \leq m} x_{(b(i))} \ , \ \hat{\sigma^2}_b = \frac{1}{m-1}\sum_{i \leq m}(x_{(b(i))} - \hat{\mu}_b)^2 \ .$$

Batch Normalization

- Q: How to estimate mean and variance efficiently?

$$\hat{\mu} = \frac{1}{n}\sum_{i\leq n} x_{(i)} \ , \ \hat{\sigma^2} = \frac{1}{n-1}\sum_{i\leq n}(x_{(i)} - \hat{\mu})^2 \ .$$

Empirical average over whole training unfeasible

Instead, we consider estimations using *minibatches* of m examp

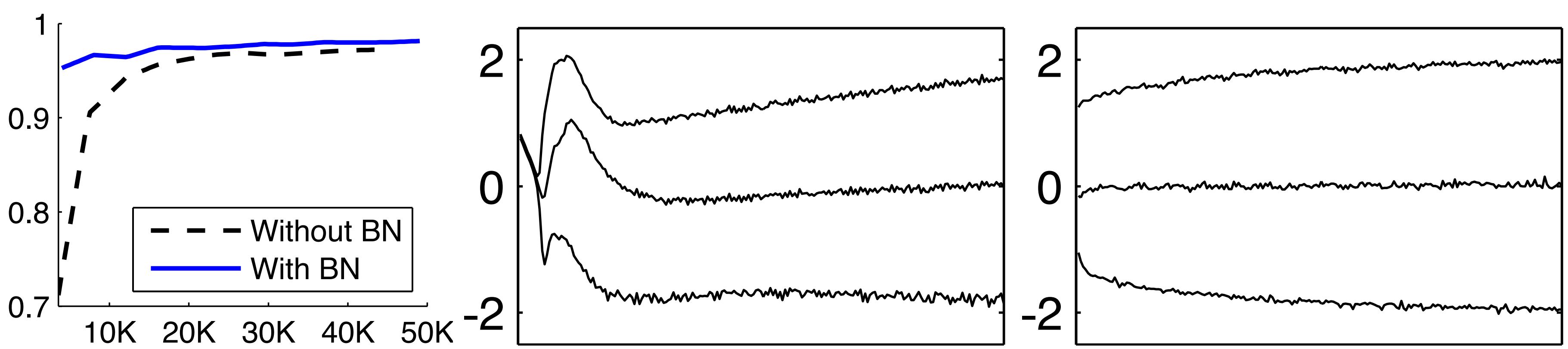
$$\hat{\mu}_b = \frac{1}{m}\sum_{i\leq m} x_{(b(i))} \ , \ \hat{\sigma^2}_b = \frac{1}{m-1}\sum_{i\leq m}(x_{(b(i))} - \hat{\mu}_b)^2 \ .$$

- Since the estimators are also function of the parameters, we must update the gradients:

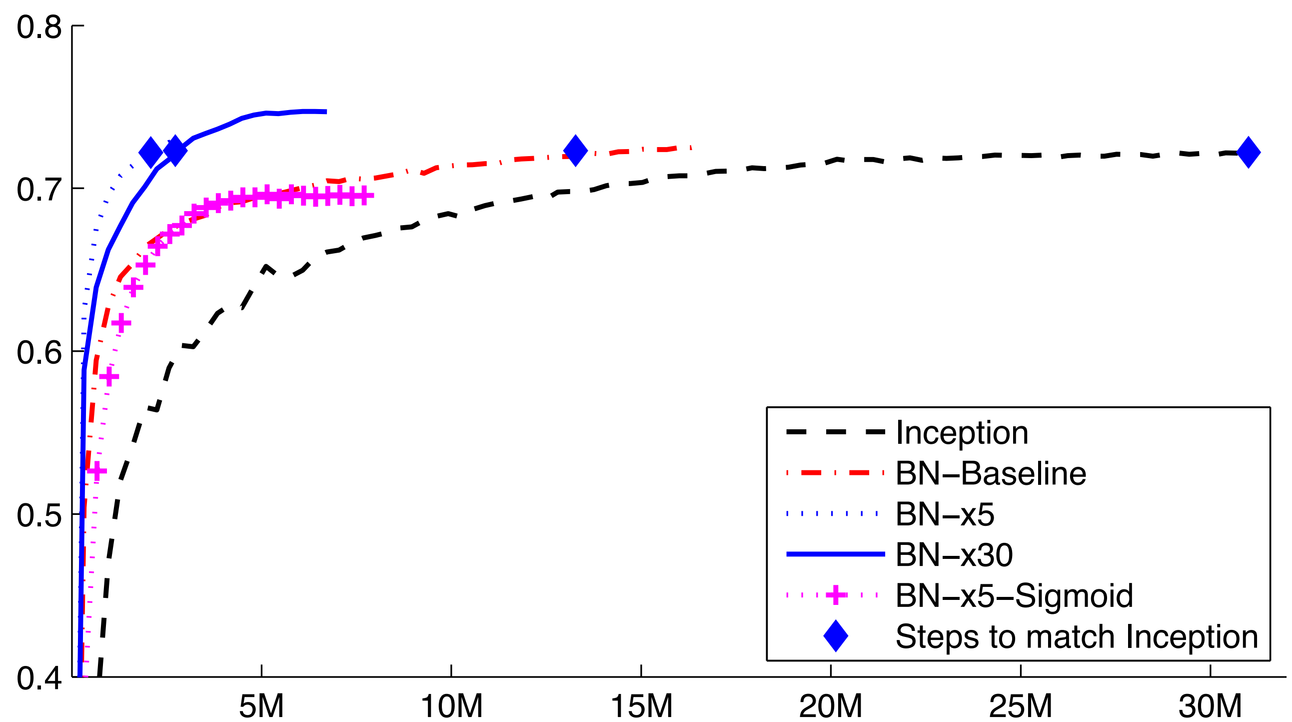
$$\begin{aligned} x_i &= \phi(\bar{x}_{i-1}; \Theta_i) \\ \tilde{x}_i &= \hat{\sigma}_i^{-1} \odot (x_i - \hat{\mu}_i) \end{aligned} \quad \Rightarrow \quad \tilde{x}_i = \tilde{\phi}(\{\bar{x}_{i-1,j}\}_{j\in\text{minibatch}}, \Theta_i) \ .$$

Batch Normalization

- Consequence 1: Much faster, more robust training
  - Less sensitive to initizaliation of the parameters
  - Simpler learning rate decay schemes.
  - Effectively larger learning rates.



(a) (b) Without BN (c) With BN



Batch Normalization

- Consequence 2: Better generalization

Model	Resolution	Crops	Models	Top-1 error	Top-5 error
GoogLeNet ensemble	224	144	7	-	6.67%
Deep Image low-res	256	-	1	-	7.96%
Deep Image high-res	512	-	1	24.88	7.42%
Deep Image ensemble	variable	-	-	-	5.98%
BN-Inception single crop	224	1	1	25.2%	7.82%
BN-Inception multicrop	224	144	1	21.99%	5.82%
BN-Inception ensemble	224	144	6	20.1%	**4.9%***

  – State-of-the-art in Imagenet classification (ResNet).
  – Key ingredient in recent GAN models.

Batch Normalization

- Some questions:

1. Combine forward normalization with backward normalization possible? useful? i.e. ensure input gradients to each layer are also normalized.
2. Particular to deep networks, or any "coupled" learning model, i.e. where Lipschitz constants of $g_{\Theta_2}(\Theta_1) = \nabla f(\Theta_1, \Theta_2)$ depend upon Θ_2 and viceversa?
3. Interplay with Dropout.
4. Better generalization explained by improved stability?

Tensor Methods in Deep Learning

- Optimizing the training error with a generic deep network is a non-convex problem.

$$\min_{\Theta} \frac{1}{n} \sum_{i \leq n} \ell(y_i, \Phi(x_i; \Theta)) + \mathcal{R}(\Theta) \ .$$

- Consider a network of depth d with ReLU nonlinearities. Seen as a function of its parameters Θ , $\Phi(x;\Theta)$ ressembles a homogeneous "piece-wise" polynomial:

$$\Phi(x;\Theta) = \sum_p \pi(x;\Theta) x_{p(1)} \prod_{j=1}^{d} \Theta^j_{p(j)} \ , \ \pi(x;\Theta) = \{0,1\} \ .$$

$$\Theta = \{\Theta^1, \dots, \Theta^d\} \ .$$

Tensor Methods in Deep Learning

- Optimizing the training error with a generic deep network is a non-convex problem.

$$\min_{\Theta} \frac{1}{n} \sum_{i \leq n} \ell(y_i, \Phi(x_i;\Theta)) + \mathcal{R}(\Theta) \ .$$

- Consider a network of depth d with ReLU nonlinearities. Seen as a function of its parameters Θ , $\Phi(x;\Theta)$ ressembles a homogeneous "piece-wise" polynomial:

$$\Phi(x;\Theta) = \sum_p \pi(x;\Theta) x_{p(1)} \prod_{j=1}^{d} \Theta^j_{p(j)} \ , \ \pi(x;\Theta) = \{0,1\} \ .$$

$$\Theta = \{\Theta^1, \dots, \Theta^d\} \ .$$

- The dependencies on Θ are partly captured by the d-order tensor $\Theta^1 \otimes \Theta^2 \dots \otimes \Theta^d$.

Tensor Methods

$$\min_{\Theta^1,\ldots,\Theta^d} F(Y, \Psi_X(\Theta^1,\ldots,\Theta^d)) + \mathcal{R}(\Theta^1,\ldots,\Theta^d) \ .$$

- Tensor factorizations are a broad class of non-convex optimization problems.

Tensor Methods

$$\min_{\Theta^1,\dots,\Theta^d} F(Y, \Psi_X(\Theta^1, \dots, \Theta^d)) + \mathcal{R}(\Theta^1, \dots, \Theta^d) \ .$$

- Tensor factorizations are a broad class of non-convex optimization problems.

- A particularly famous instance is the matrix factorization problem:

$$\min_{U,V} \ \ell(Y, UV^T) + \mathcal{R}(U, V) \ , \ Y \in \mathbb{R}^{n \times m}, U \in \mathbb{R}^{n \times d}, \ V \in \mathbb{R}^{m \times d} \ .$$

  - Low-rank factorizations (e.g. PCA)
  - Sparse factorizations (Dictionary Learning, NMF)

Motivation: Matrix factorization

- Example: low-rank factorization.

$$\min_{U,V} \ell(Y, UV^T) \ , \ \text{s.t.} \ \operatorname{rank}(UV^T) \leq r \ .$$

  - When $\ell(Y,X) = \|Y - X\|_{op}, \ \ell(Y,X) = \|Y - X\|_F$ OK
  - We can *lift* the problem and relax the constraint:

$$\min_{X} \ell(Y, X) + \lambda \|X\|_* \ , \qquad \|X\|_* = \text{Nuclear norm of } X.$$

  - Factorized and relaxed formulations are connected via a variational principle:

$$\|X\|_* = \min_{UV^T = X} \frac{1}{2}(\|U\|_F^2 + \|V\|_F^2) \ .$$

- Q: General case?

Tensor Norms [Bach, Haeffele&Vidal]

- A first generalization is the tensor norm

$$\|X\|_{u,v} = \inf_r \min_{UV^T = X} \frac{1}{2}\left(\sum_i \|U_i\|_u^2 + \|V_i\|_v^2\right) .$$

Theorem [H-V]: A local minimizer of the factorized problem $\min_{U,V} \ell(Y, UV^T) + \lambda \sum_{i \leq r} \|U_i\|_u \|V_i\|_v$ such that for some i $U_i = V_i = 0$ is a global minimizer of the convex problem $\min_X \ell(Y, X) + \lambda \|X\|_{u,v}$ as well as the factorized problem.

- This produces an *optimality certificate:* we use a surrogate convex problem to obtain a guarantee that a non-convex problem is solved optimally.

From Tensor Factorizations to Deep Nets

- We start by generalizing a multilinear mapping (tensor) to homogeneous maps $\phi(\Theta^1, \dots, \Theta^d)$:

$$\forall \, \Theta \, , \forall \, \alpha \geq 0 \; , \; \phi(\alpha \Theta^1, \dots, \alpha \Theta^d) = \alpha^s \phi(\Theta^1, \dots, \Theta^d) \; .$$

s: degree of homogeneity.

Ex: ReLU $\rho(x) = \max(0, x)$ is homogeneous of degree 1.

- We construct models by adding *r* copies of homogenous maps:

$$\Phi_r(\Theta^1, \dots, \Theta^d) = \sum_{i \leq r} \phi(\Theta_i^1, \dots, \Theta_i^d) \; .$$

- We consider

$$\min_{\Theta^1, \dots, \Theta^d} \ell(Y, \Phi_r(\Theta^1, \dots, \Theta^d)) + \lambda \mathcal{R}(\Theta^1, \dots, \Theta^d) \; ,$$

Key assumption: $\mathcal{R}$ is positively homogeneous of the same degree as Φ.

From Tensor Factorizations to Deep Nets

$$\Phi_r(\Theta^1, \ldots, \Theta^d) = \sum_{i=1}^{r} \phi(\Theta^1, \ldots, \Theta^d) \ .$$

Matrices:

$$\Phi(U, V) = UV^T = \sum_{i=1}^{r} U_i V_i^T \ \ (\phi(U_i, V_i) = U_i V_i^T) \ .$$

Higher-order Tensors:

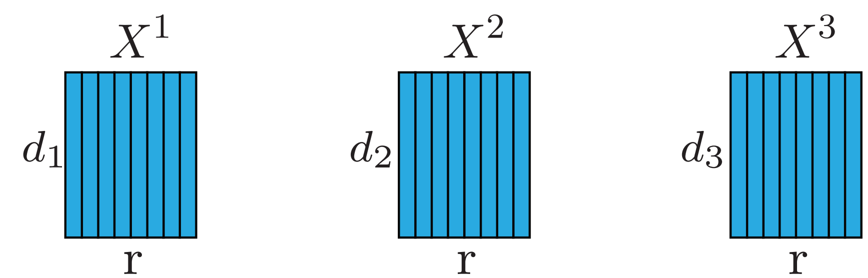
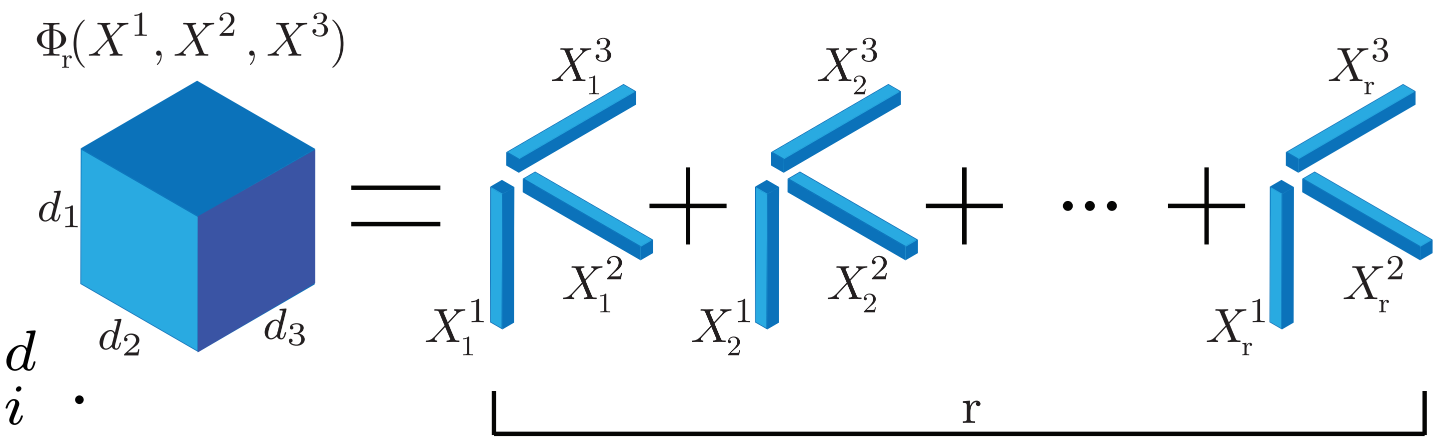


figure credit: R. Vidal



$$\phi(\Theta_i^1, \ldots, \Theta_i^d) = \Theta_i^1 \otimes \cdots \otimes \Theta_i^d \ .$$

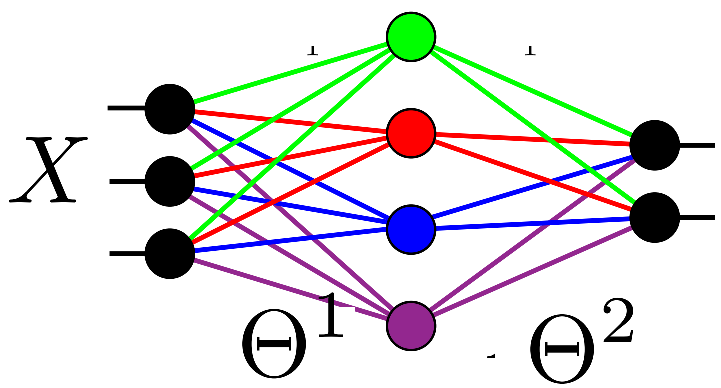
Candecomp/Parafac (CP) Tensor decomposition.

Adaptation to Deep Models

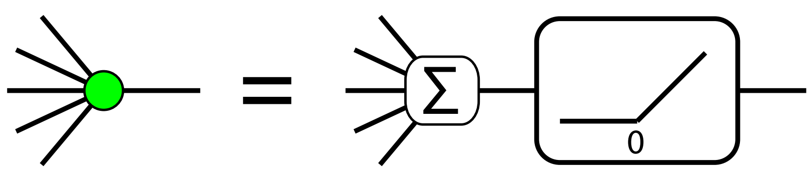
figure credit: R. Vidal

- ReLU Network:

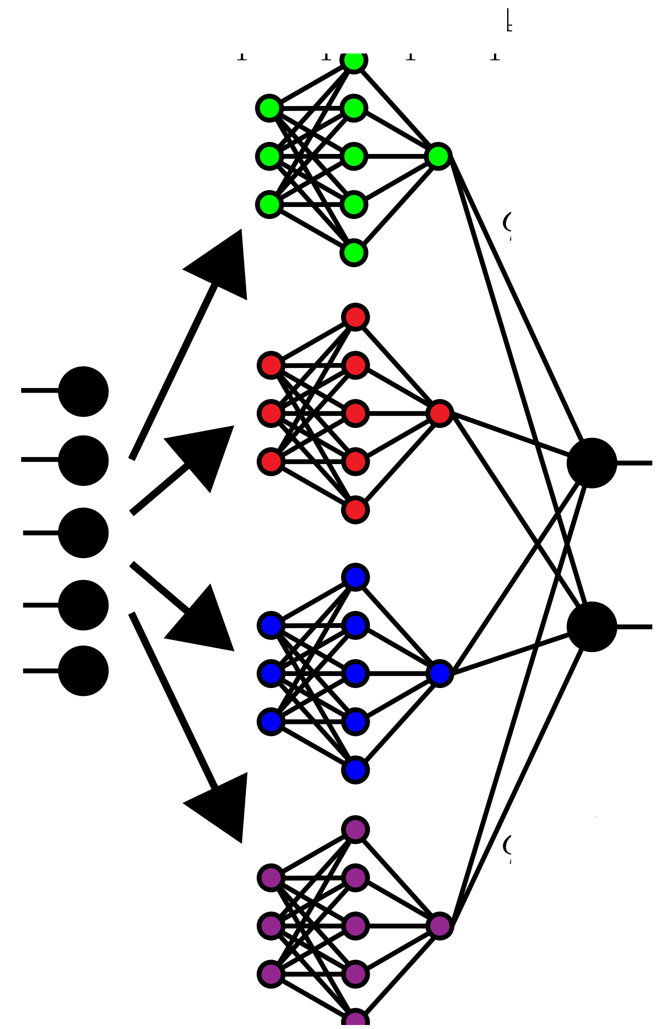
ReLU Network with One Hidden Layer



Rectified Linear Unit (ReLU)



$$\phi(\Theta^1, \Theta^2)$$



Multilayer ReLU Parallel Network

$$\Phi(\Theta^1, \ldots, \Theta^d) = \sum_i \phi(\Theta^1_i, \ldots, \Theta^d_i) \; .$$

Adaptation to Deep Models

- In the matrix case, the variational principle was

$$\|X\|_{u,v} = \min_{UV^T=X} \sum_{i\leq r} \|U_i\|_u \|V_i\|_v \ .$$

- This is generalized to

$$\mathcal{R}(\Theta) = \min_{\Theta^1,\dots,\Theta^d} \sum_{i\leq r} g(\Theta_i^1,\dots,\Theta_i^d) \ , \ s.t. \ \Phi_r(\Theta^1,\dots,\Theta^d) = \Theta \ .$$

- **Proposition [H-V]:** $\mathcal{R}$ is convex.
  Also, if g is positively homogeneous of degree s, so is $\mathcal{R}$.

Adaptation to Deep Models

Theorem [H-V]: A local minimizer of the factorized problem

$$\min_{\Theta^k} \ell(Y, \sum_{i\leq r} \phi_r(\Theta_i^k)) + \lambda \sum_{i\leq r} g(\Theta_i^k)$$

such that for some i and all k $\Theta_i^k = 0$ is a global minimizer for both factorized problem and the convex formulation

$$\min_{\Theta} \ell(Y, \Theta) + \lambda \mathcal{R}(\Theta).$$

- Global optimality certificate for a broad class of non-convex optimization problems, including some form of deep learning architectures.
- Q: How to use this certificate in practice?

Adaptation to Deep Models

- Pros
  - Global optimality certificate, easy to check
  - Inclues nonlinear models as long as they are homogeneous.
  - Provides a possible meta-algorithm: increase the lifting value r progressively is local optimum does not very condition.

- Cons
  - How much do we need to increase r in practice?
  - How stringent is the homogenous regularization condition?