Gradient Descent and the Structure of Neural Network Cost Functions

presentation by Ian Goodfellow

adapted for www.deeplearningbook.org
from a presentation to the
CIFAR Deep Learning summer school on August 9, 2015

Given neural network parameters $\boldsymbol{\theta}$, find the value of θ that minimizes cost function

$$J(\theta).$$

-Exhaustive search

-Random search (genetic algorithms)

-Analytical solution

-Model-based search (e.g. Bayesian optimization)

-Neural nets usually use **gradient-based search**

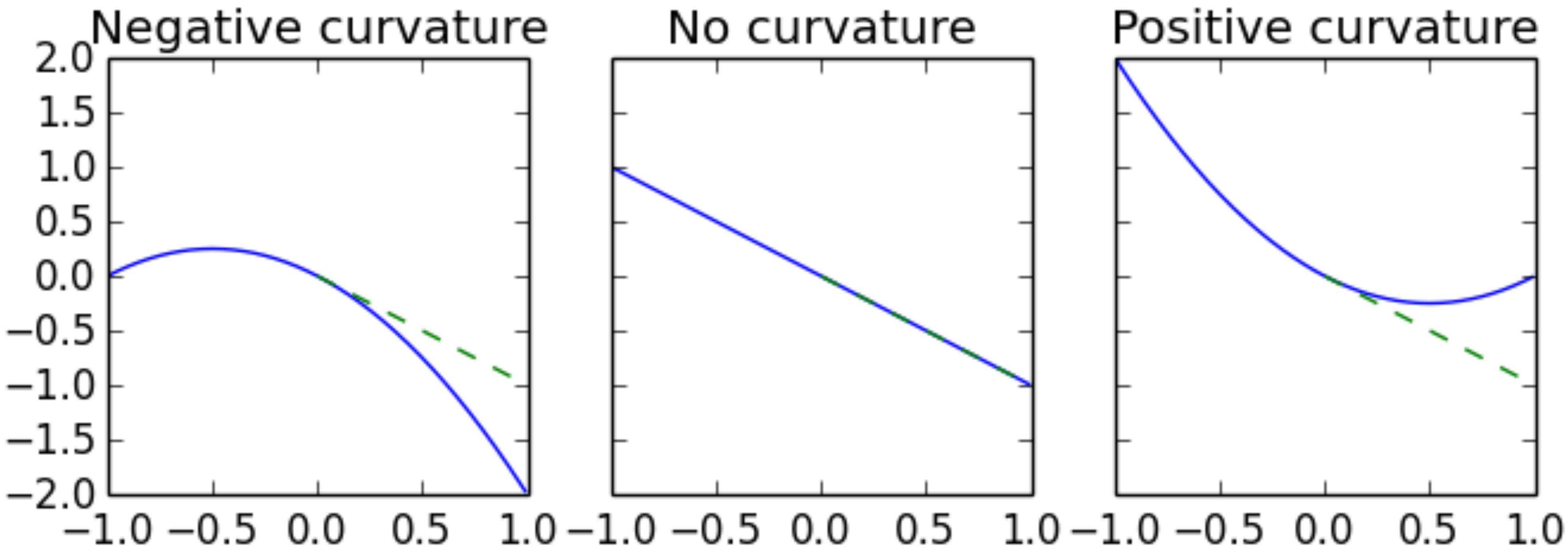
(Goodfellow 2015)

In this presentation....

- "Exact Solutions to the Nonlinear Dynamics of Learning in Deep Linear Neural Networks." Saxe et al, ICLR 2014
- "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization." Dauphin et al, NIPS 2014
- "The Loss Surfaces of Multilayer Networks." Choromanska et al, AISTATS 2015
- "Qualitatively characterizing neural network optimization problems." Goodfellow et al, ICLR 2015

Derivatives and Second Derivatives

Cost function	Gradient	Hessian
$J(\boldsymbol{\theta})$	$\boldsymbol{g} = \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$	$\boldsymbol{H}$
	$g_i = \frac{\partial}{\partial \theta_i} J(\boldsymbol{\theta})$	$H_{i,j} = \frac{\partial}{\partial \theta_j} g_i$



(Goodfellow 2015)

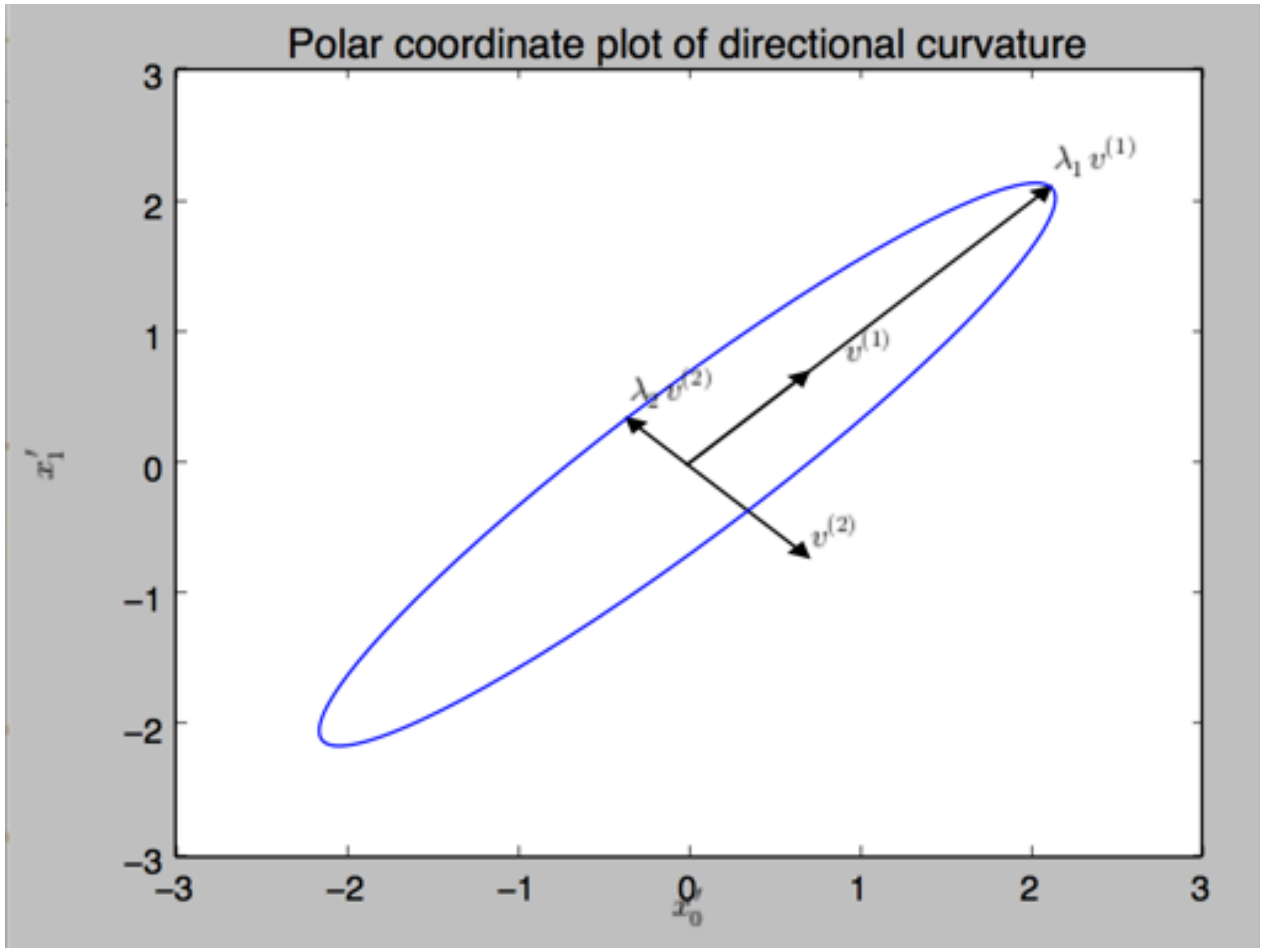
Directional Curvature

$$\boldsymbol{Q} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n]$$

$$\boldsymbol{H} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top$$

Second derivative in direction d:

$$\boldsymbol{d}^\top \boldsymbol{H} \boldsymbol{d} = \sum_i \lambda_i \cos^2 \angle(\boldsymbol{v}_i, \boldsymbol{d})$$



(Goodfellow 2015)

Taylor series approximation

$$f(x) = f(x_0) + (x - x_0) f'(x) + \frac{1}{2}(x - x_0)^2 f''(x) + \ldots$$

$$J(\boldsymbol{\theta}) = J(\boldsymbol{\theta}_0) + (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \boldsymbol{g} + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \boldsymbol{H} (\boldsymbol{\theta} - \boldsymbol{\theta}_0) + \ldots$$

Baseline

Linear change due to gradient

Correction from directional curvature

(Goodfellow 2015)

How much does a gradient step reduce the cost?

2nd-order Taylor series prediction:

$$J(\boldsymbol{\theta} - \epsilon \boldsymbol{g}) \approx J(\boldsymbol{\theta}) - \epsilon \boldsymbol{g}^\top \boldsymbol{g} + \frac{1}{2}\epsilon^2 \boldsymbol{g}^\top \boldsymbol{H} \boldsymbol{g}$$

The improvement in the worst case when $\boldsymbol{g}$ aligns with λ_{max}:

$$(\epsilon - \frac{1}{2}\epsilon^2 \lambda_{\text{max}}) \boldsymbol{g}^\top \boldsymbol{g}$$

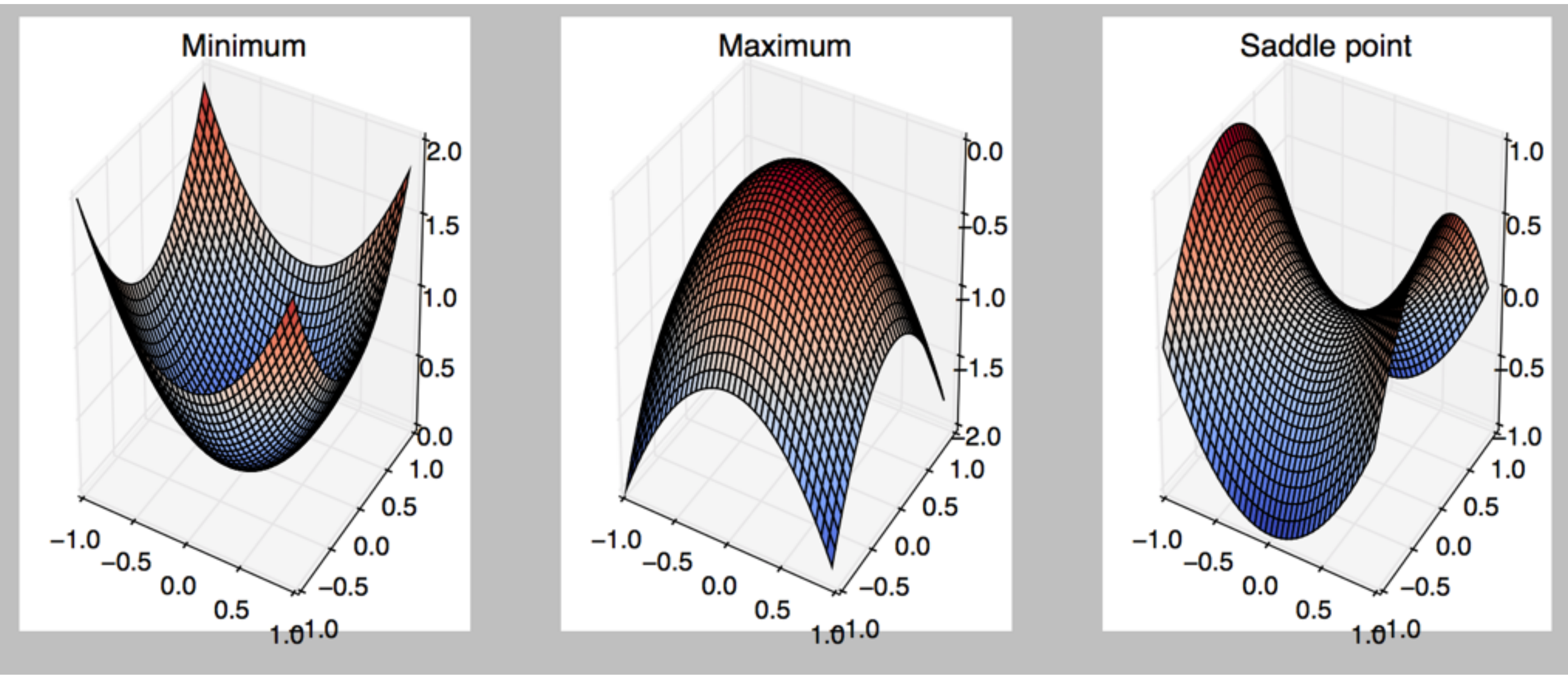
When $\boldsymbol{g}^\top \boldsymbol{H} \boldsymbol{g} \leq 0$, Taylor series predicts that all step sizes improve. Otherwise, optimal step size is

$$\epsilon^* = \frac{\boldsymbol{g}^\top \boldsymbol{g}}{\boldsymbol{g}^\top \boldsymbol{H} \boldsymbol{g}}$$

(Goodfellow 2015)

Critical points

Zero gradient, and Hessian with...



All positive eigenvalues | All negative eigenvalues | Some positive and some negative

(Goodfellow 2015)

Newton's method

Assume $\lambda_{\text{min}} > 0$.

Then all critical points are minima.

Solve $\boldsymbol{g}(\boldsymbol{\theta}) = 0$ for $\boldsymbol{\theta}$.

Too hard?

Use 1st-order Taylor approximation of $\boldsymbol{g}$:

Solve

$$\boldsymbol{g} + \boldsymbol{H}(\boldsymbol{\theta} - \boldsymbol{\theta}_0) = 0$$

$$\Rightarrow \boldsymbol{\theta} = \boldsymbol{\theta}_0 - \boldsymbol{H}^{-1}\boldsymbol{g}.$$

(Goodfellow 2015)

Newton's method's failure mode

When both signs of eigenvalues occur,
solving $\boldsymbol{g}(\boldsymbol{\theta}) = 0$ for $\boldsymbol{\theta}$
can yield...
a minimum...
a maximum...
or a saddle point.

The old view of SGD as difficult

- SGD usually moves downhill
- SGD eventually encounters a critical point
- Usually this is a minimum
- However, it is a *local minimum*
- J has a high value at this critical point
- Some *global minimum* is the real target, and has a much lower value of J

(Goodfellow 2015)

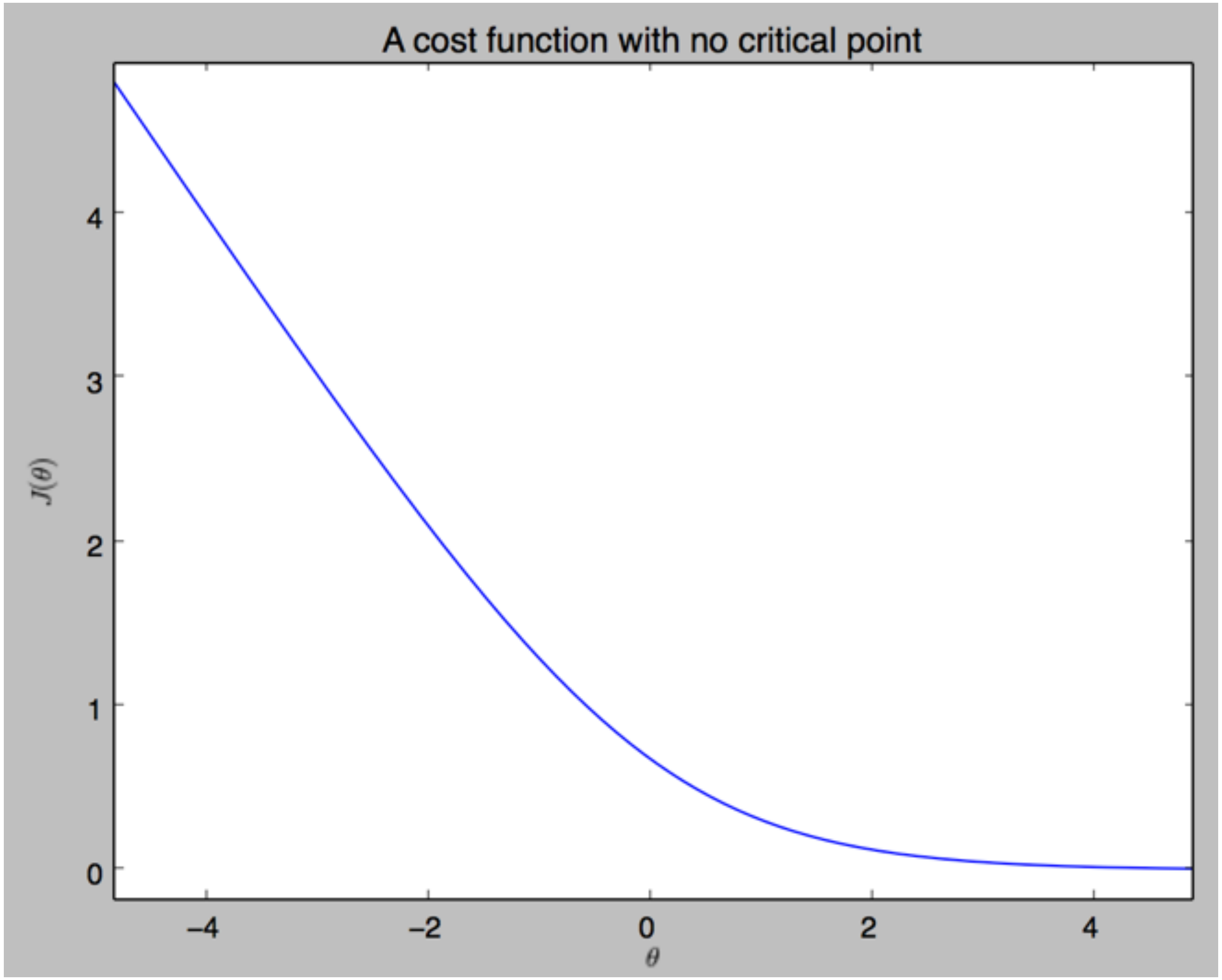
The new view: does SGD get stuck on saddle points?

- SGD usually moves downhill
- SGD eventually encounters a critical point
- Usually this is a saddle point
- SGD is stuck, and the main reason it is stuck is that it fails to exploit negative curvature

(as we will see, this happens to Newton's method, but not very much to SGD)

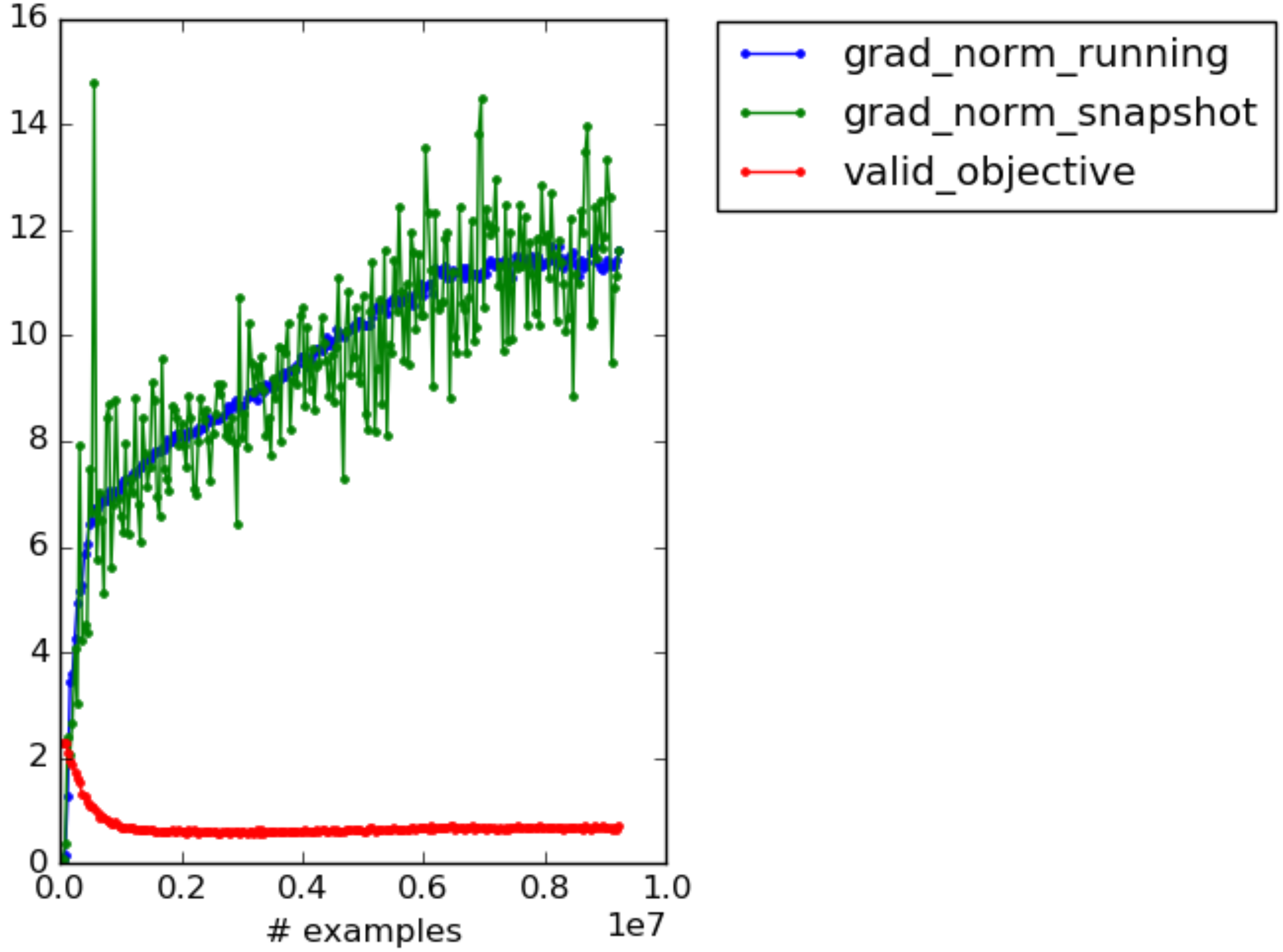
(Goodfellow 2015)

Some functions lack critical points



(Goodfellow 2015)

SGD may not encounter critical points



(Goodfellow 2015)

Gradient descent flees saddle points

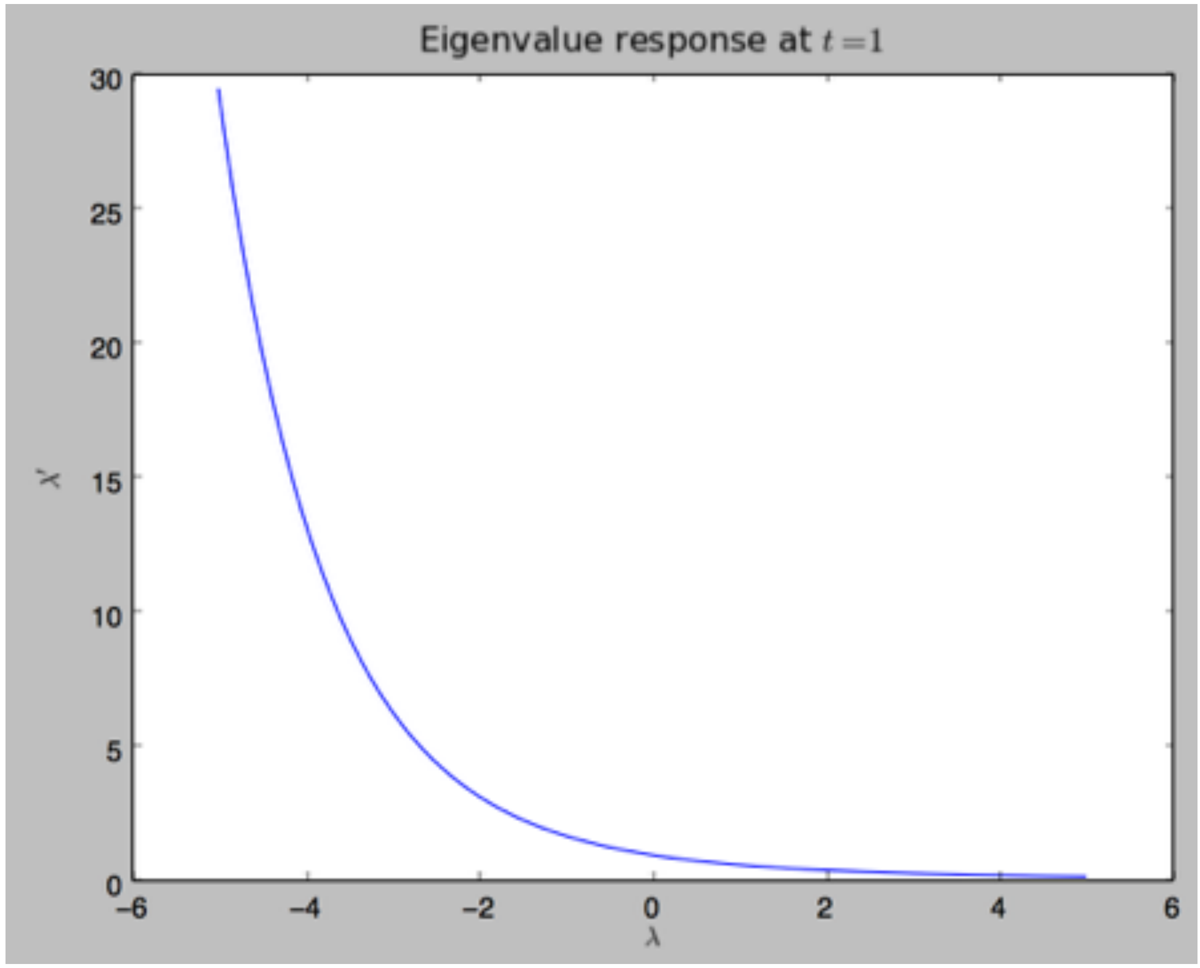
$$\frac{d}{dt}\boldsymbol{\theta}(t) = -\boldsymbol{g} - \boldsymbol{H}\left(\boldsymbol{\theta}(t) - \boldsymbol{\theta}(0)\right)$$

$$\Rightarrow$$

$$\boldsymbol{\theta}(t) = \boldsymbol{\theta}(0) - \boldsymbol{Q}\Lambda'(t)\boldsymbol{Q}^T\boldsymbol{g}$$

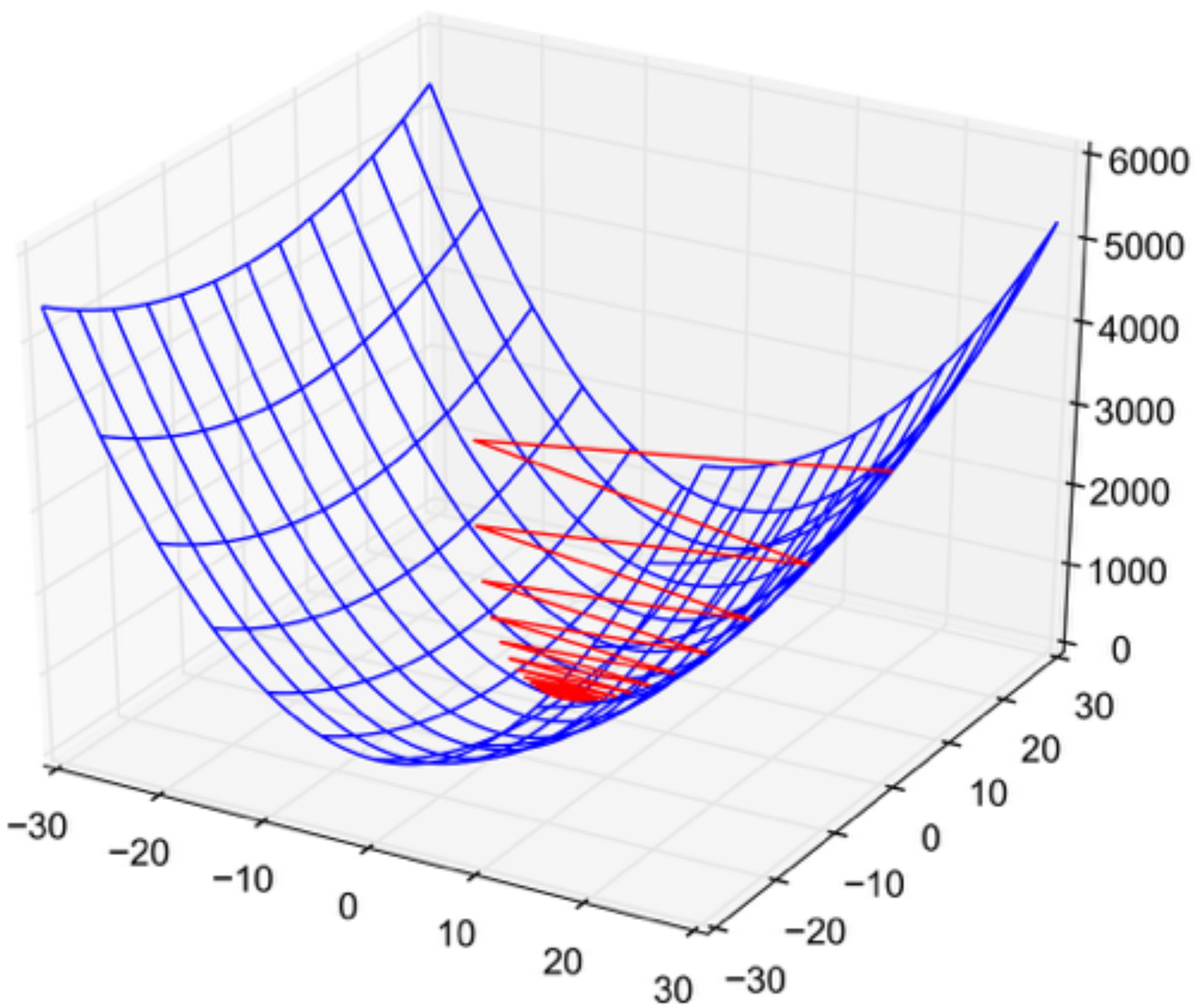
where

$$\lambda'(t) = \frac{1 - \exp(-\lambda t)}{\lambda}.$$



(Goodfellow 2015)

Poor conditioning



(Goodfellow 2015)

Poor conditioning

Optimal step size is:

$$\epsilon^* = \frac{\boldsymbol{g}^\top \boldsymbol{g}}{\boldsymbol{g}^\top \boldsymbol{H} \boldsymbol{g}}$$

If $\boldsymbol{g}$ aligns with a large eigenvalue, ϵ^* is very small.
If ϵ is chosen without computing $\boldsymbol{g}^\top \boldsymbol{H} \boldsymbol{g}$, the step might go uphill.
Usually we choose ϵ from a schedule fixed in advance, and $\boldsymbol{g}^\top \boldsymbol{H} \boldsymbol{g}$ can fluctuate rapidly, especially if $\boldsymbol{H}$ has many distinct eigenvalues.

(Goodfellow 2015)

Why convergence may not happen

- Never stop if function doesn't have a local minimum
- Get "stuck," possibly still moving but not improving
  - Too bad of conditioning
  - Too much gradient noise
  - Overfitting
  - Other?
- Usually we get "stuck" before finding a critical point
- Only Newton's method and related techniques are attracted to saddle points

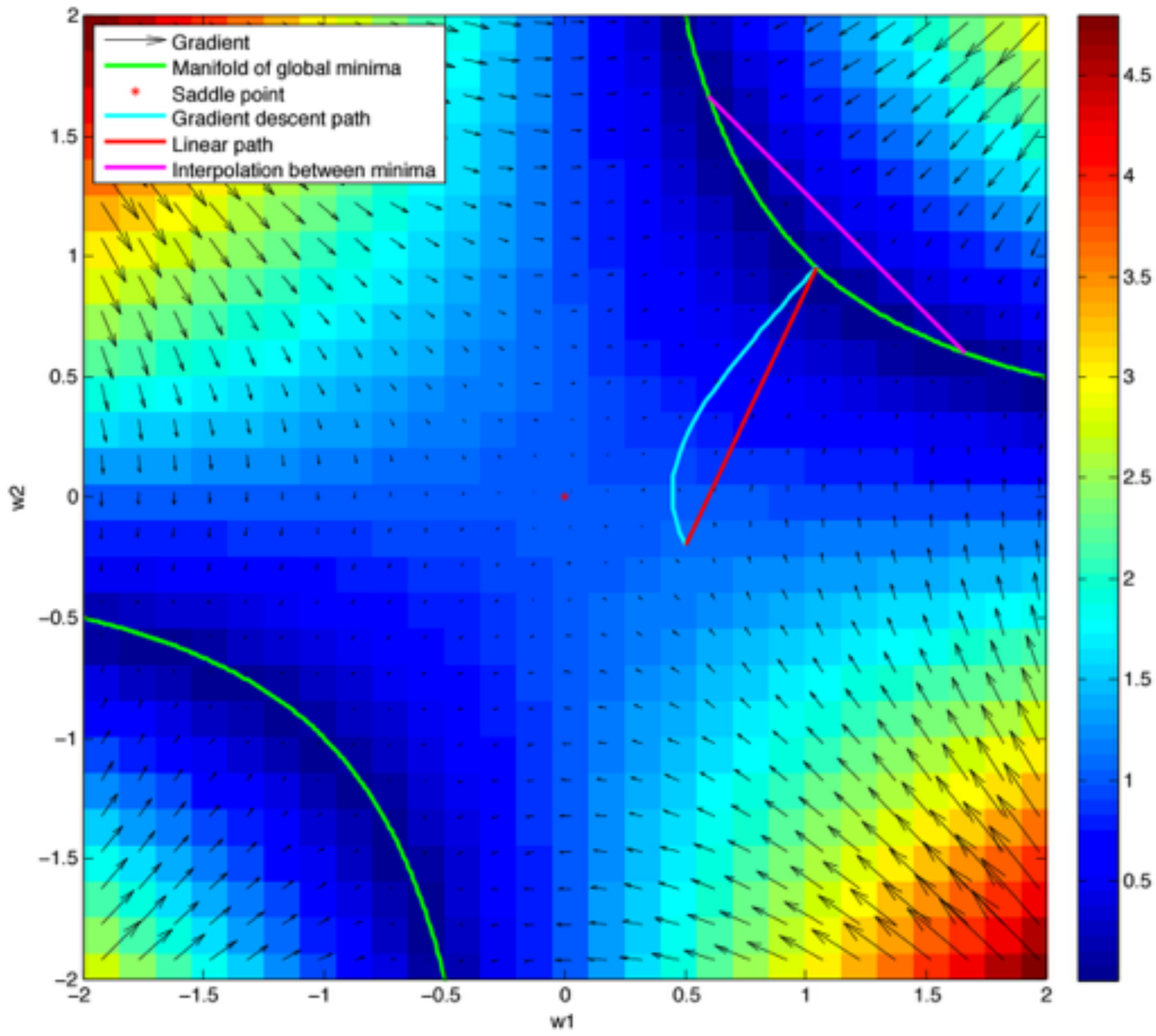
Are saddle points or local minima more common?

- Imagine for each eigenvalue, you flip a coin
- If heads, the eigenvalue is positive, if tails, negative
- Need to get all heads to have a minimum
- Higher dimensions -> exponentially less likely to get all heads
- Random matrix theory:
- The coin is weighted; the lower J is, the more likely to be heads
- So most local minima have low J!
- Most critical points with high J are saddle points!

(Goodfellow 2015)

Do neural nets have saddle points?

- Saxe et al, 2013:
- neural nets without non-linearities have many saddle points
- all the minima are global
- all the minima form a connected manifold



(Goodfellow 2015)

Do neural nets have saddle points?

- Dauphin et al 2014: Experiments show neural nets do have as many saddle points as random matrix theory predicts
- Choromanska et al 2015: Theoretical argument for why this should happen
- Major implication: **most minima are good, and this is more true for big models.**
- Minor implication: the reason that *Newton's method* works poorly for neural nets is its attraction to the ubiquitous saddle points.

The state of modern optimization

- We can optimize most classifiers, autoencoders, or recurrent nets if they are based on linear layers
- Especially true of LSTM, ReLU, maxout
- It may be *much slower than we want*
- Even depth does not prevent success, Sussillo 14 reached 1,000 layers
- We may not be able to optimize more exotic models
- Optimization benchmarks are usually not done on the exotic models

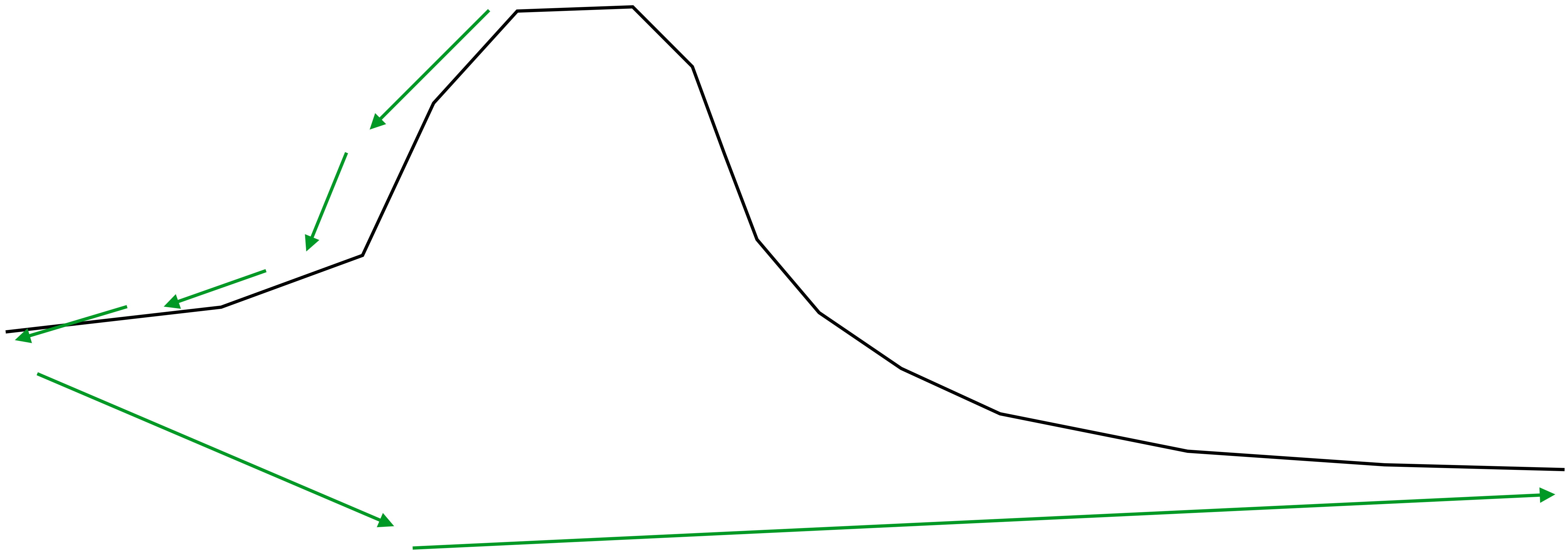
Why is optimization so slow?

We can fail to compute good local updates (get "stuck").

Or local information can disagree with global information,

even when there are not any non-global minima,

even when there are not any minima of any kind



(Goodfellow 2015)

Questions for visualization

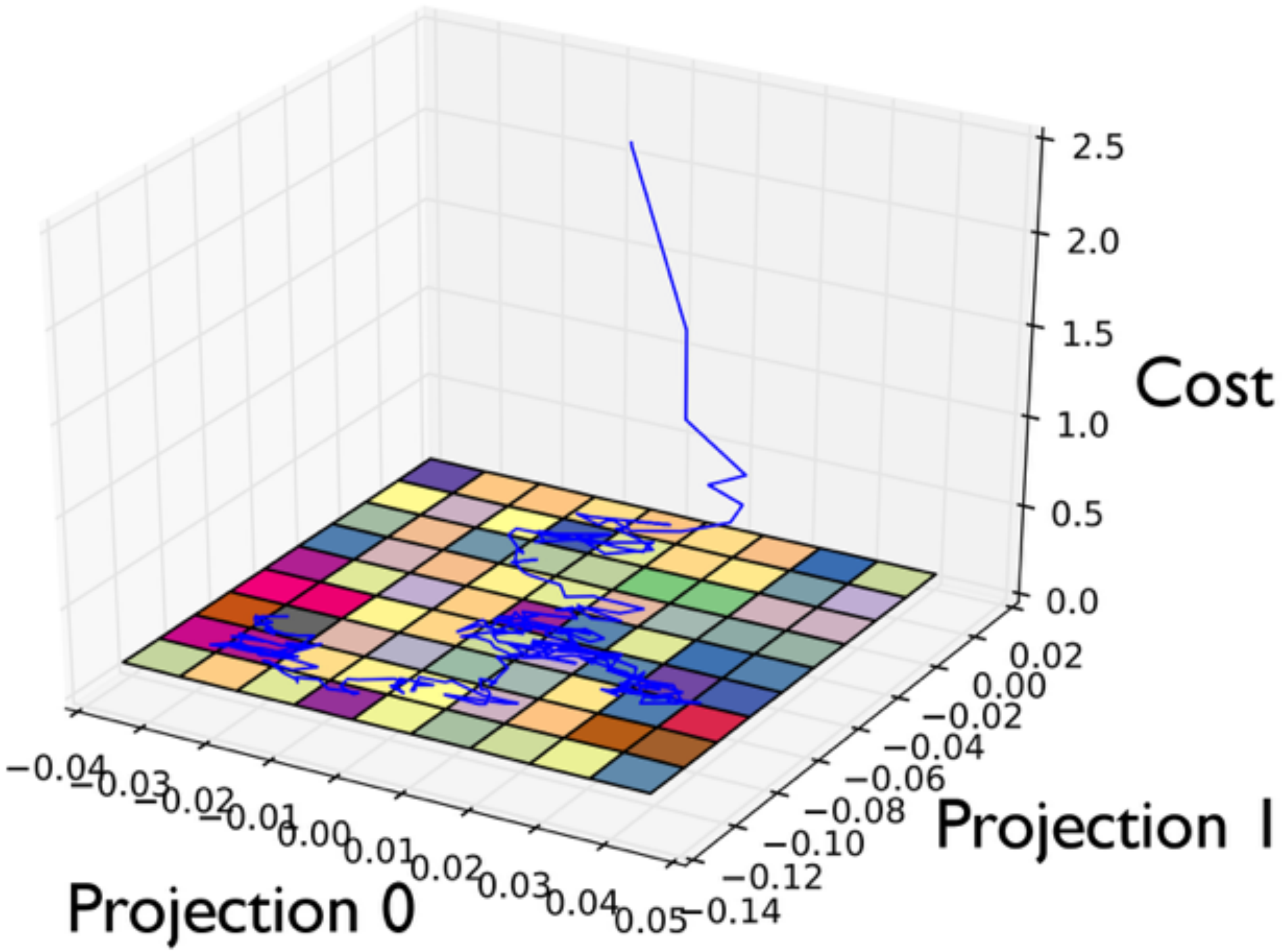
- Does SGD get stuck in local minima?
- Does SGD get stuck on saddle points?
- Does SGD waste time navigating around global obstacles despite properly exploiting local information?
- Does SGD wind between multiple local bumpy obstacles?
- Does SGD thread a twisting canyon?

(Goodfellow 2015)

History written by the winners

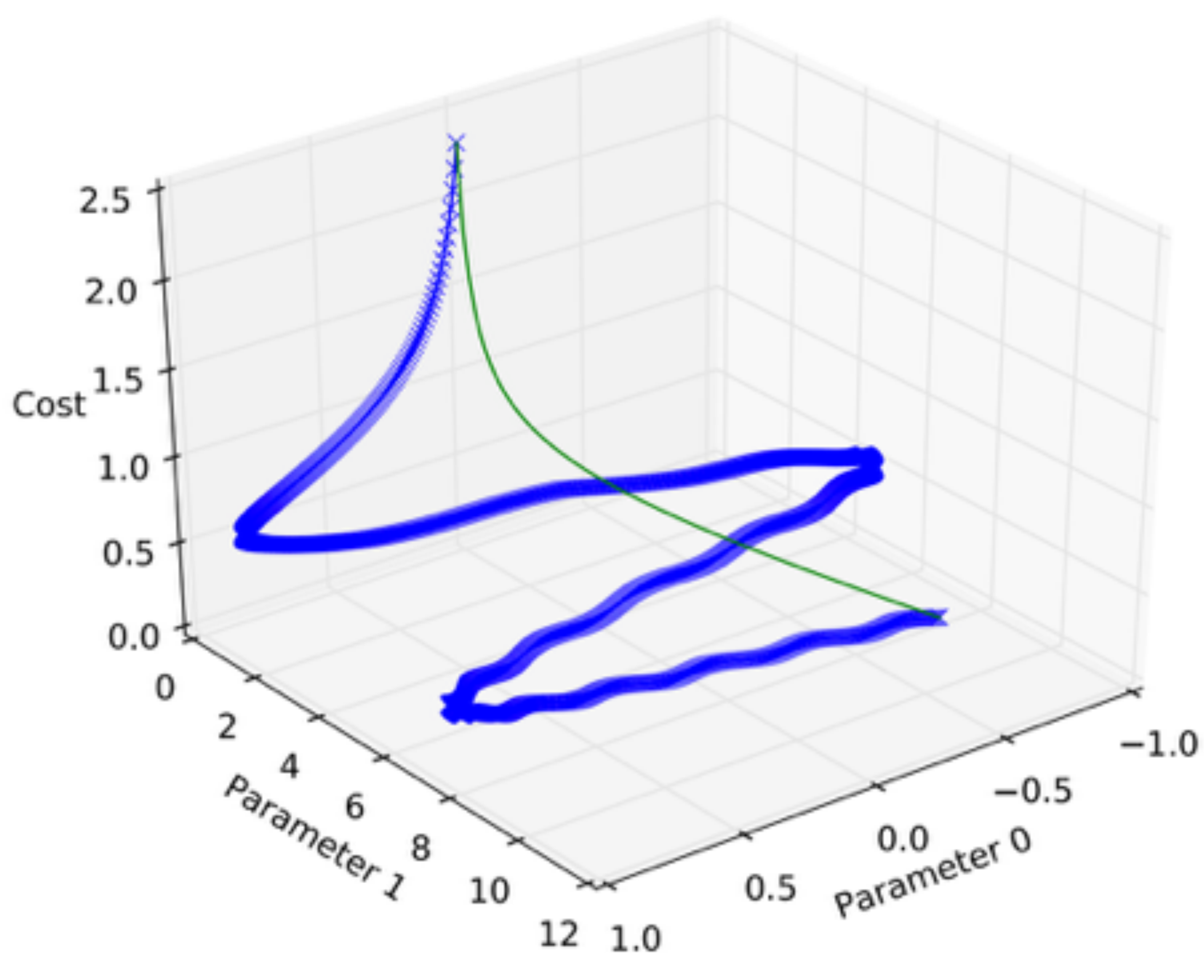
- Visualize trajectories of (near) SOTA results
- Selection bias: looking at success
- Failure is interesting, but hard to attribute to optimization
- Careful with interpretation: SGD never encounters X, or SGD fails if it encounters X?

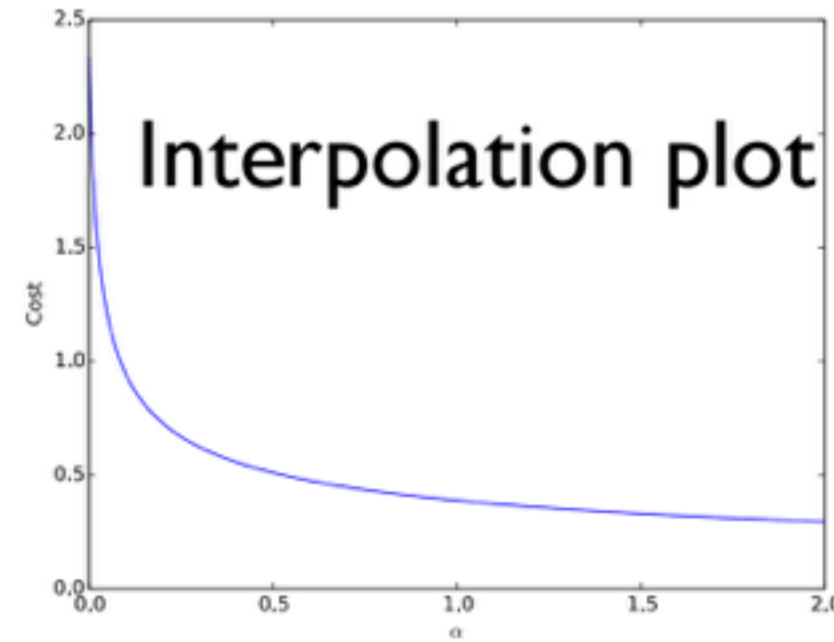
2D Subspace Visualization

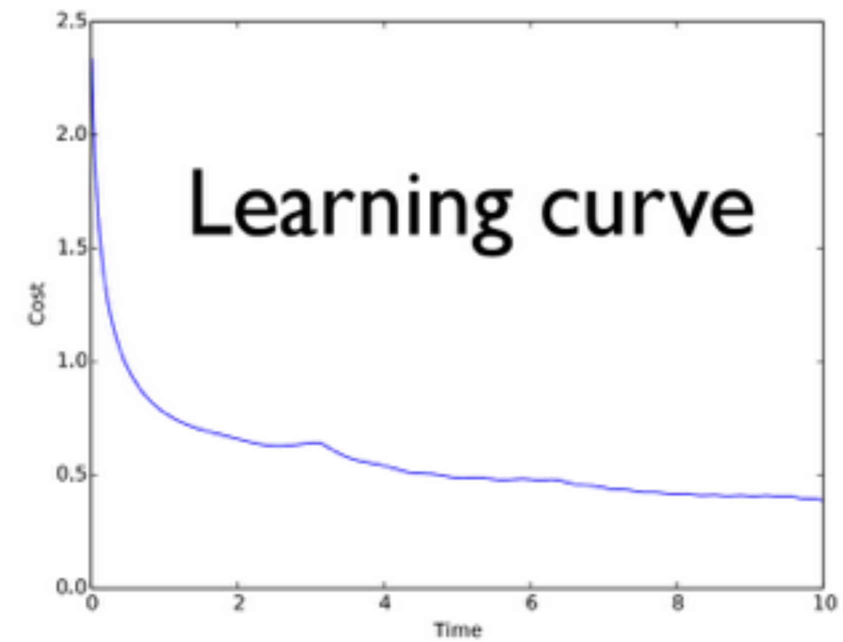


(Goodfellow 2015)

A Special 1-D Subspace

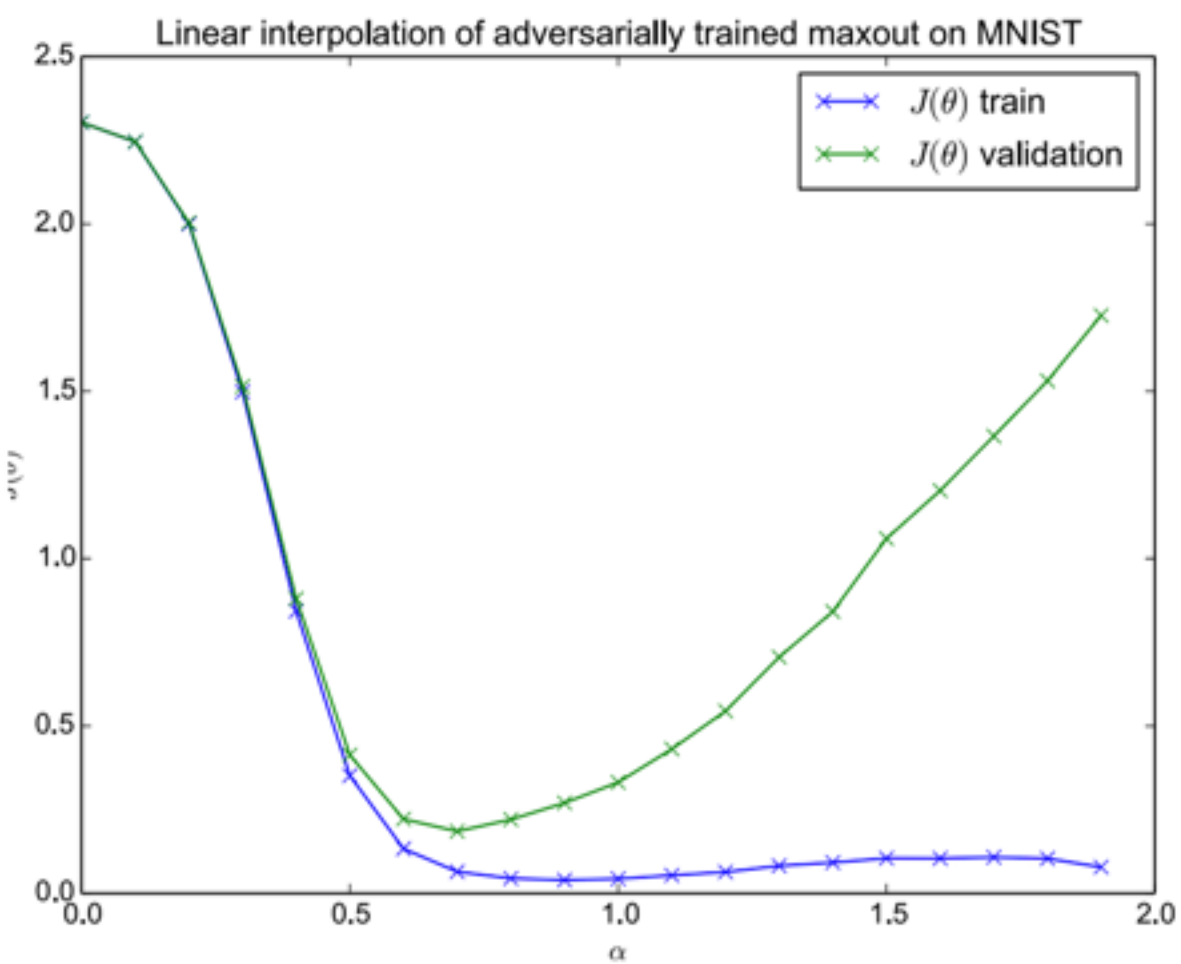


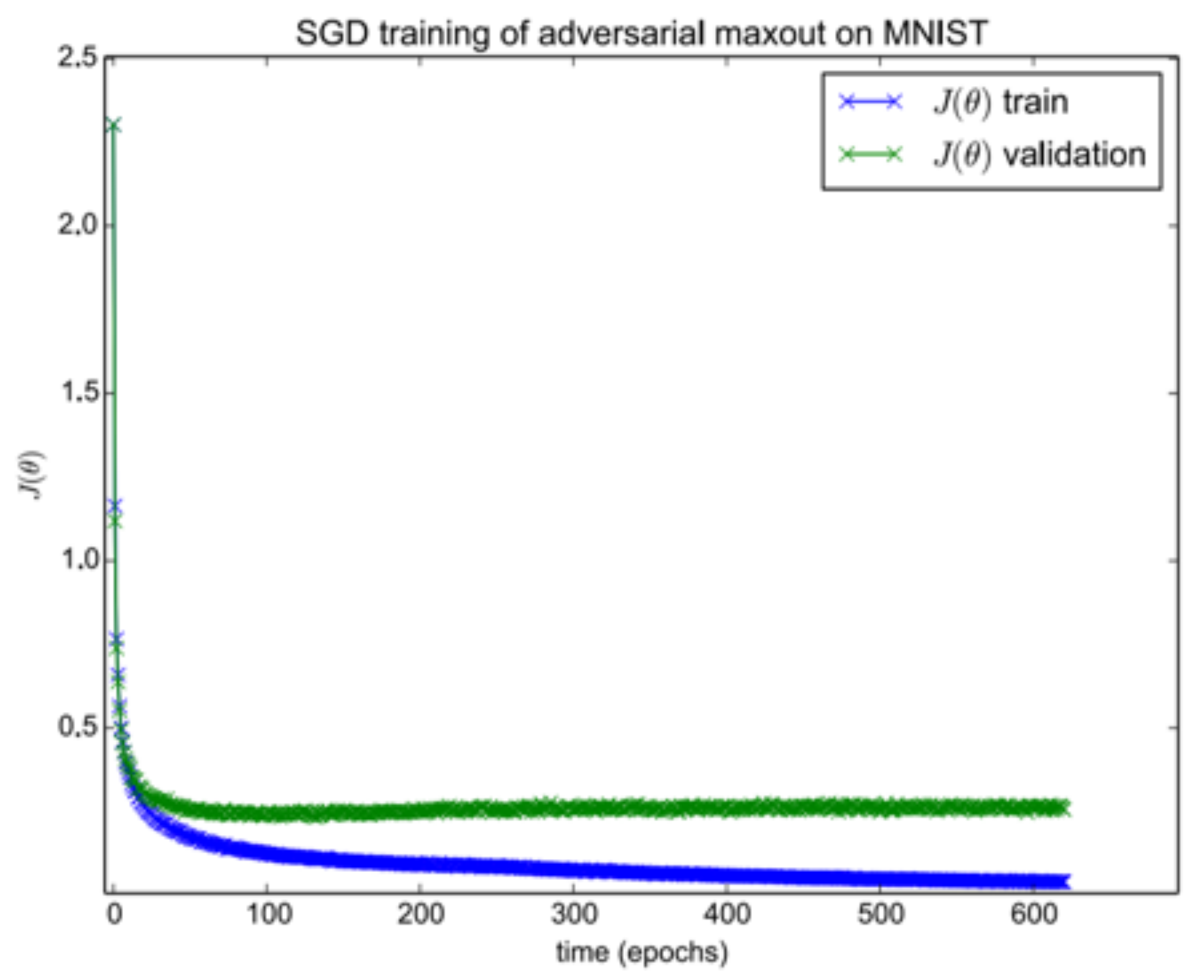




(Goodfellow 2015)

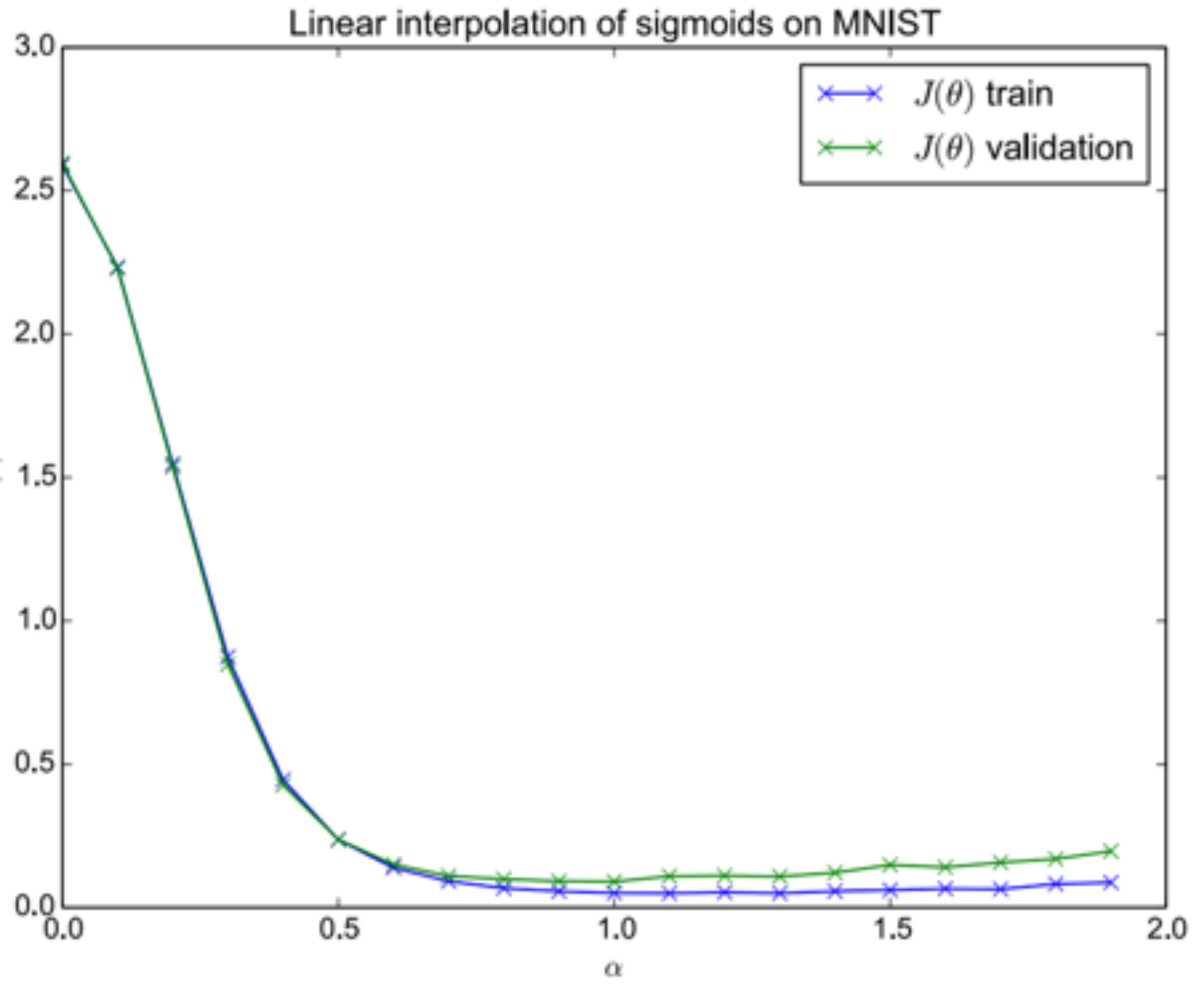
Maxout / MNIST experiment

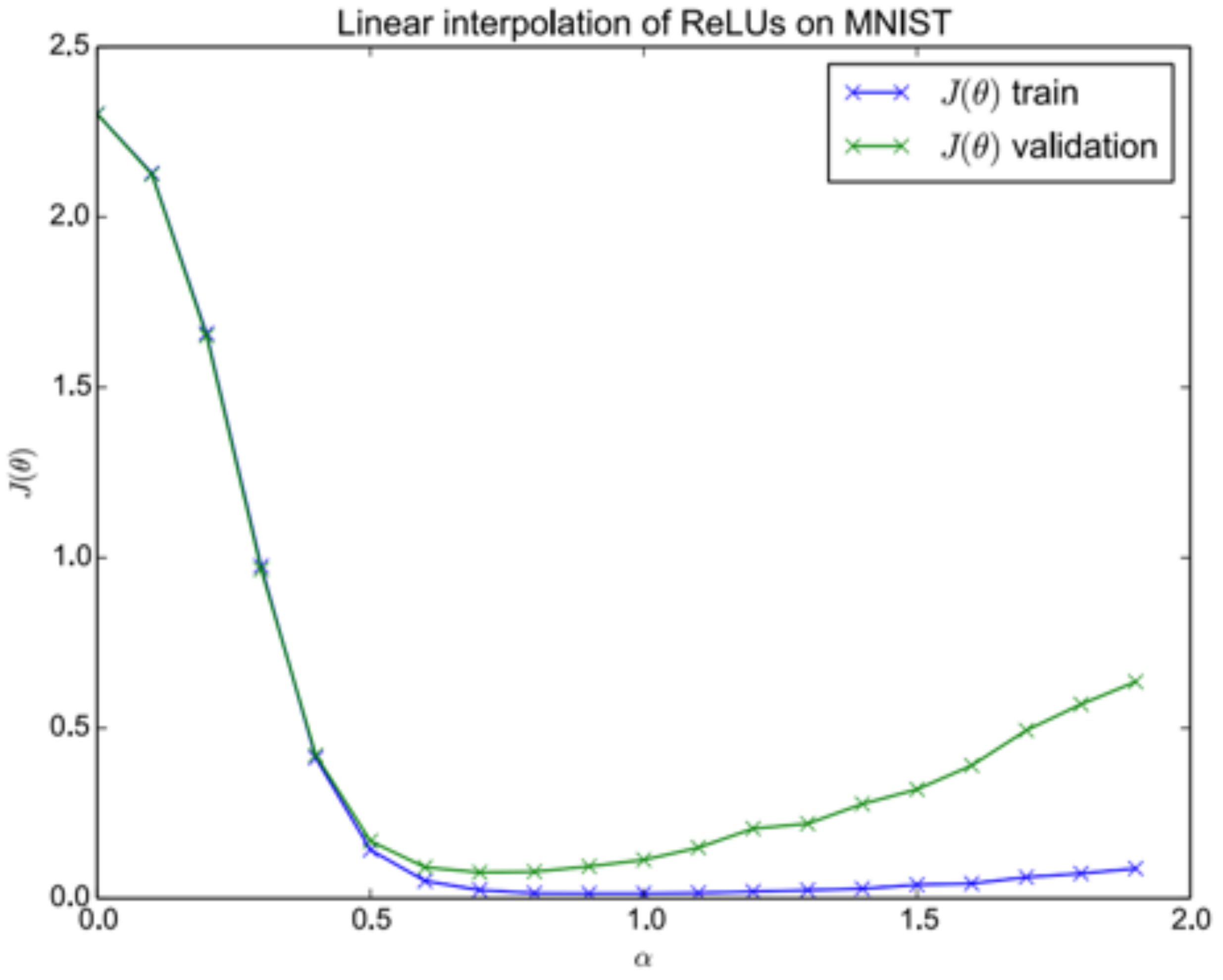




(Goodfellow 2015)

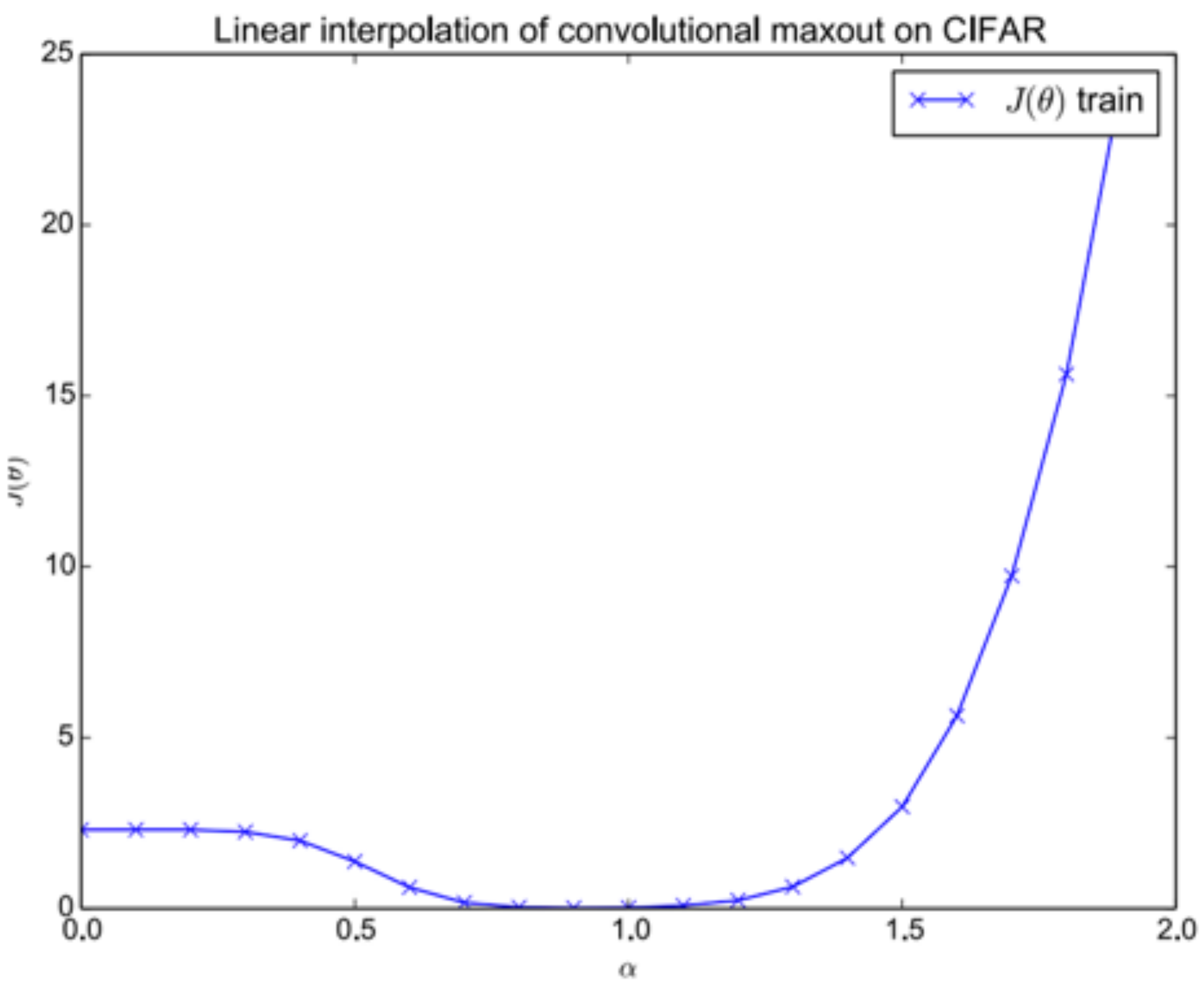
Other activation functions

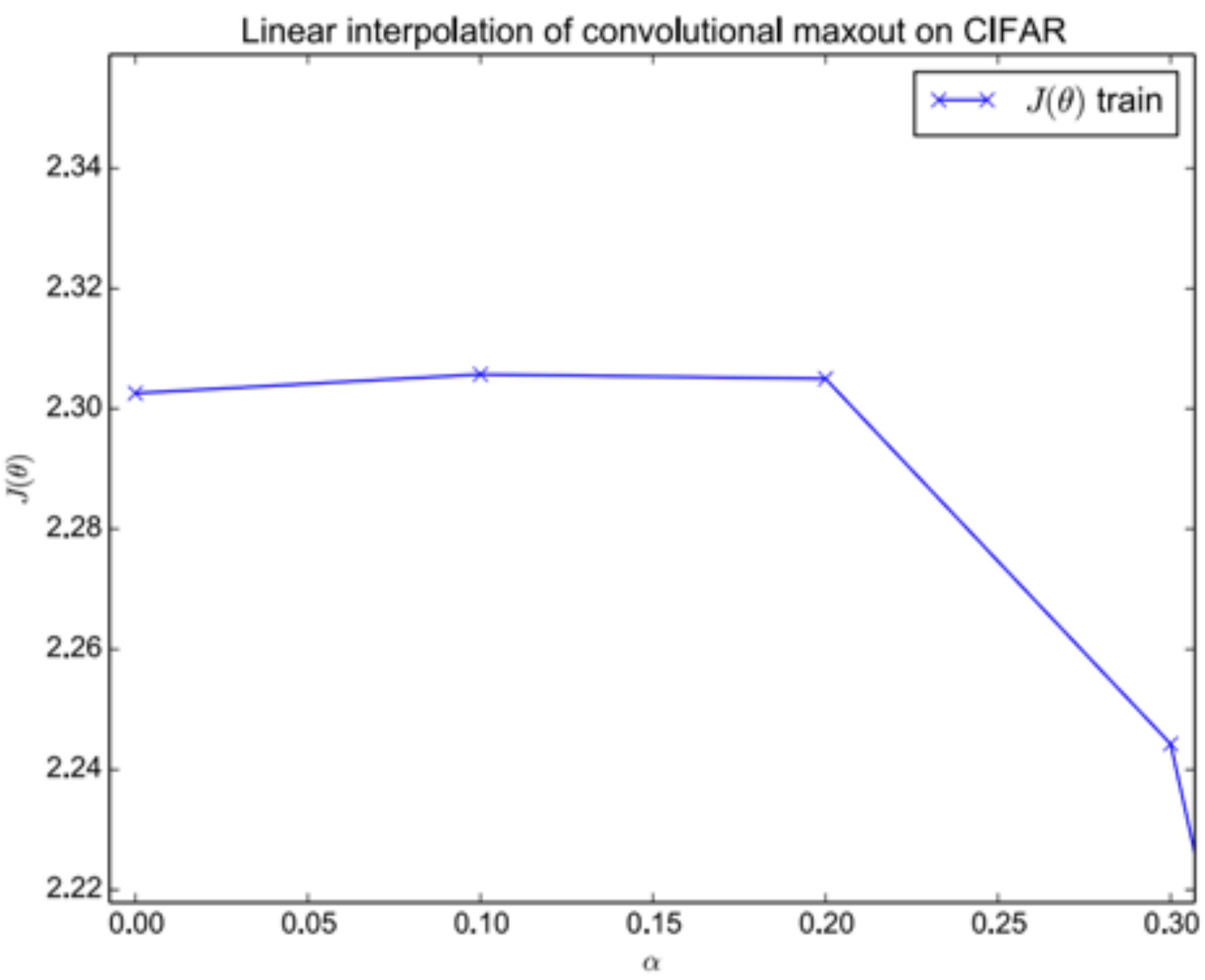




(Goodfellow 2015)

Convolutional network

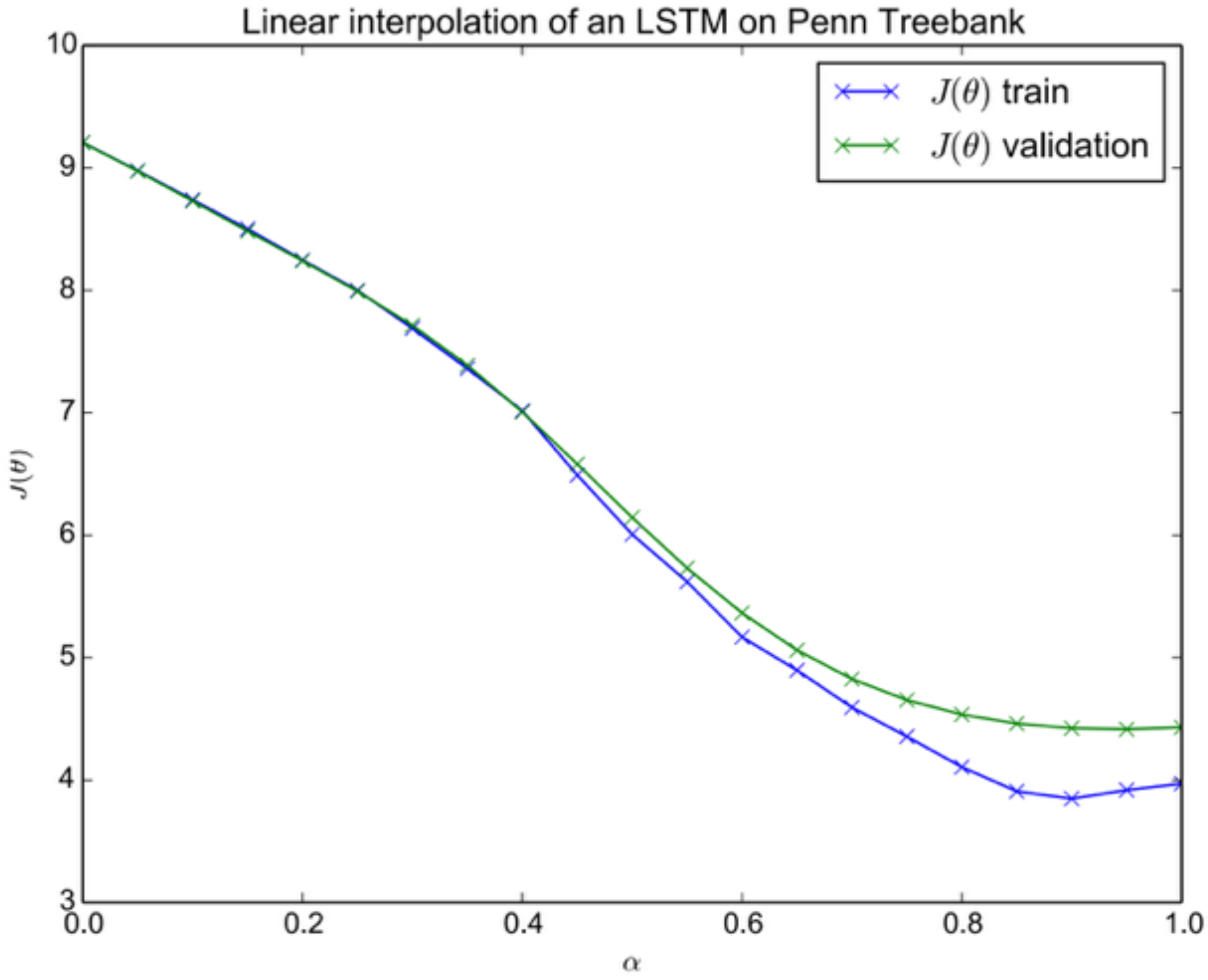




The "wrong side of the mountain" effect

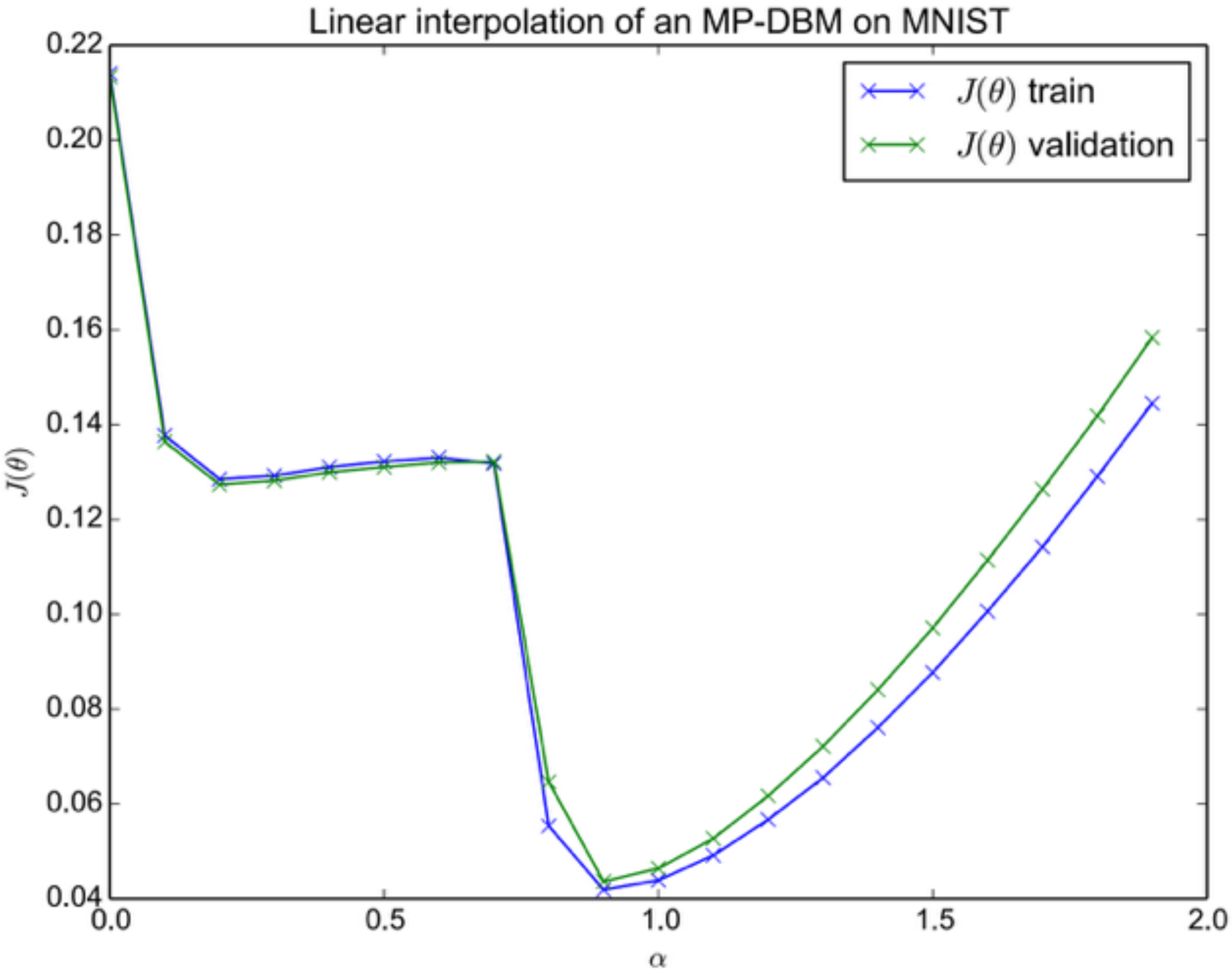
(Goodfellow 2015)

Sequence model (LSTM)



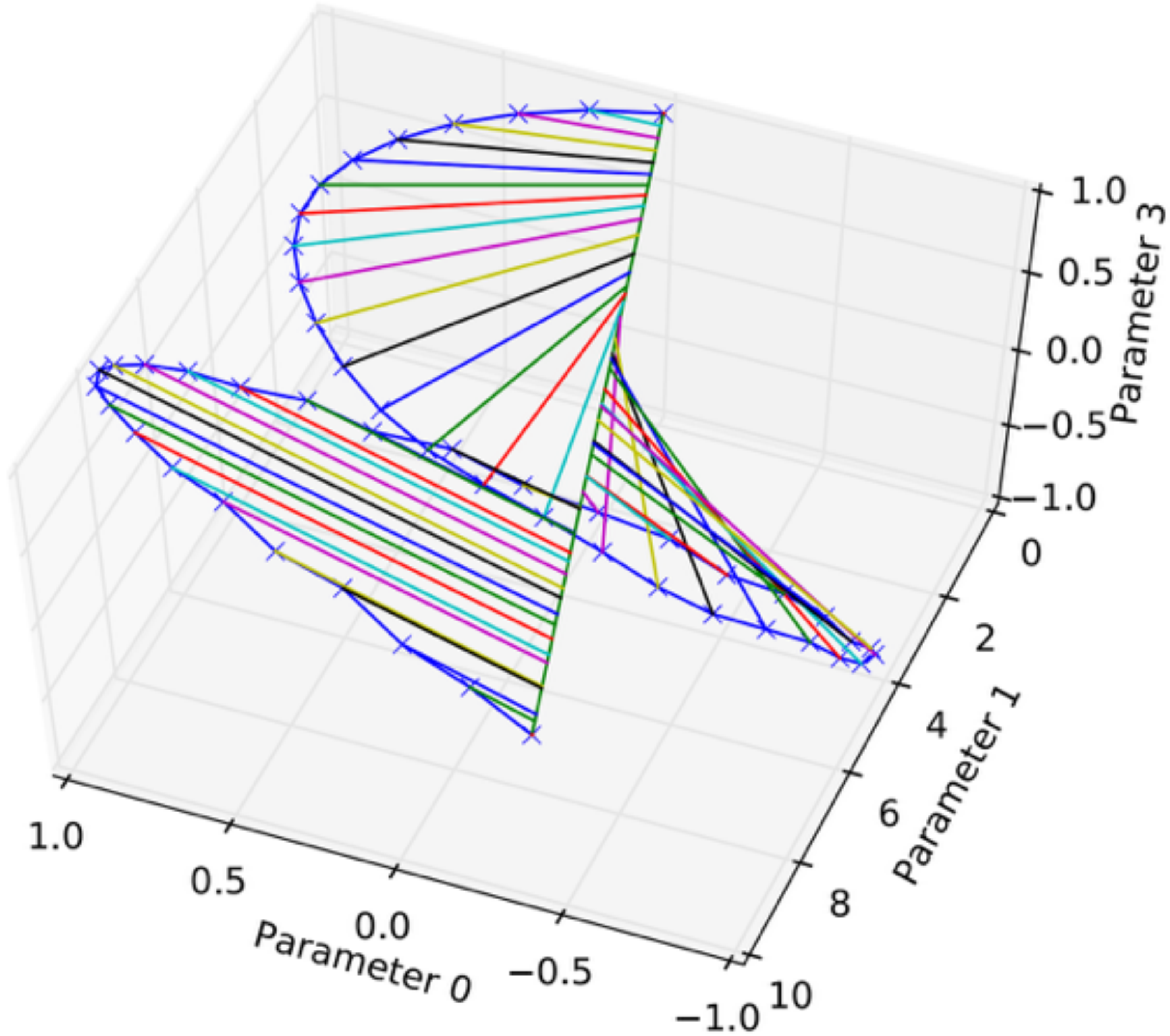
(Goodfellow 2015)

Generative model (MP-DBM)



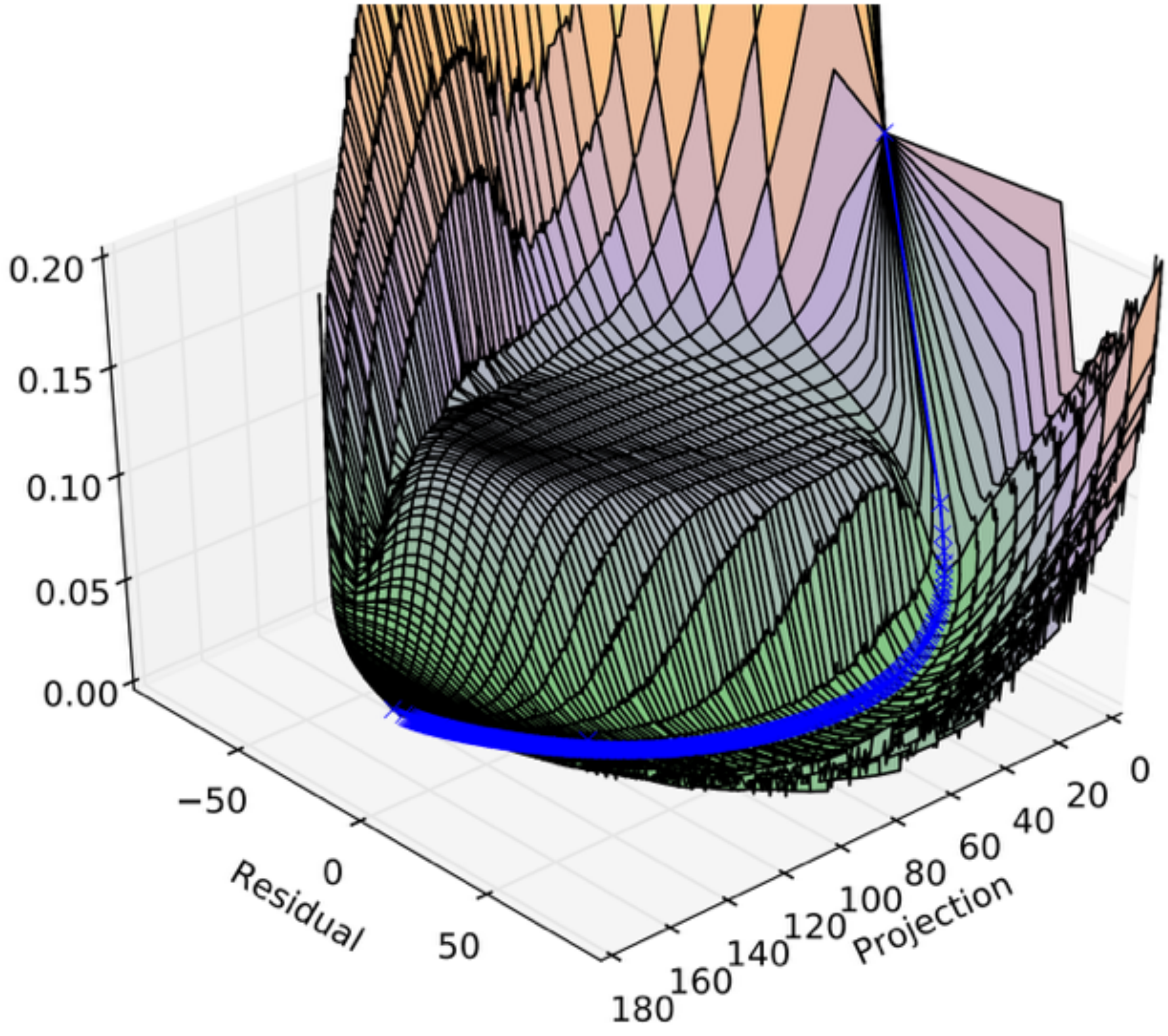
(Goodfellow 2015)

3-D Visualization

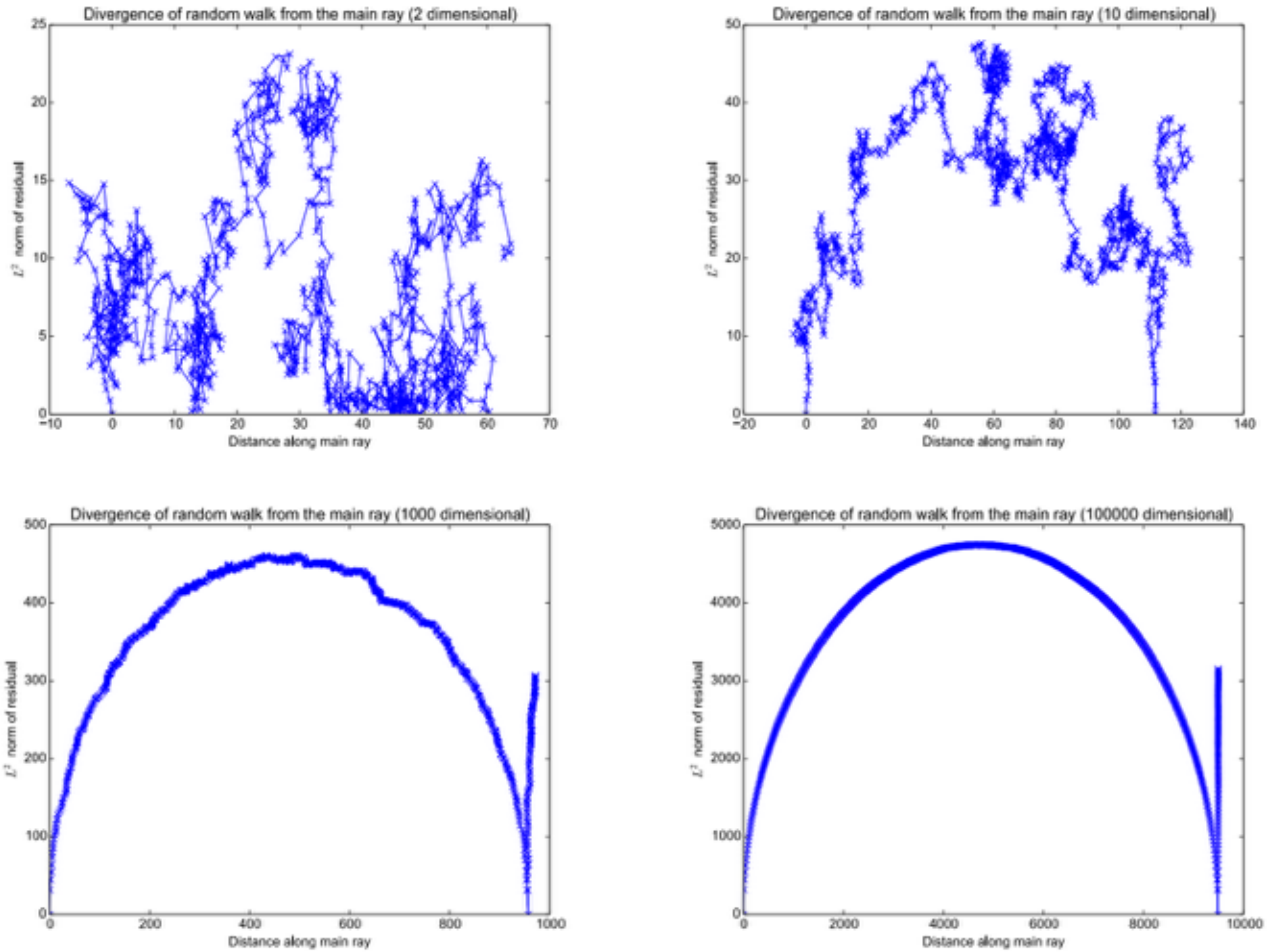


(Goodfellow 2015)

3-D Visualization of MP-DBM

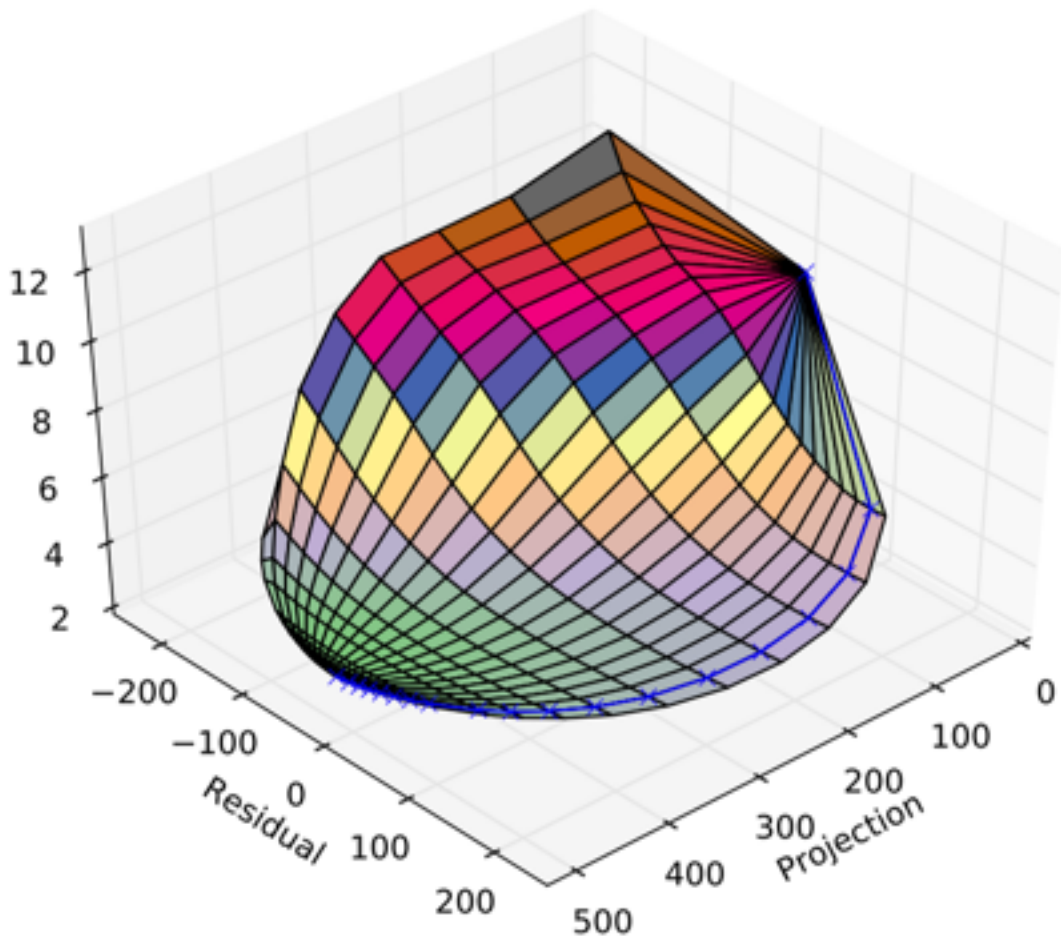


Random walk control experiment

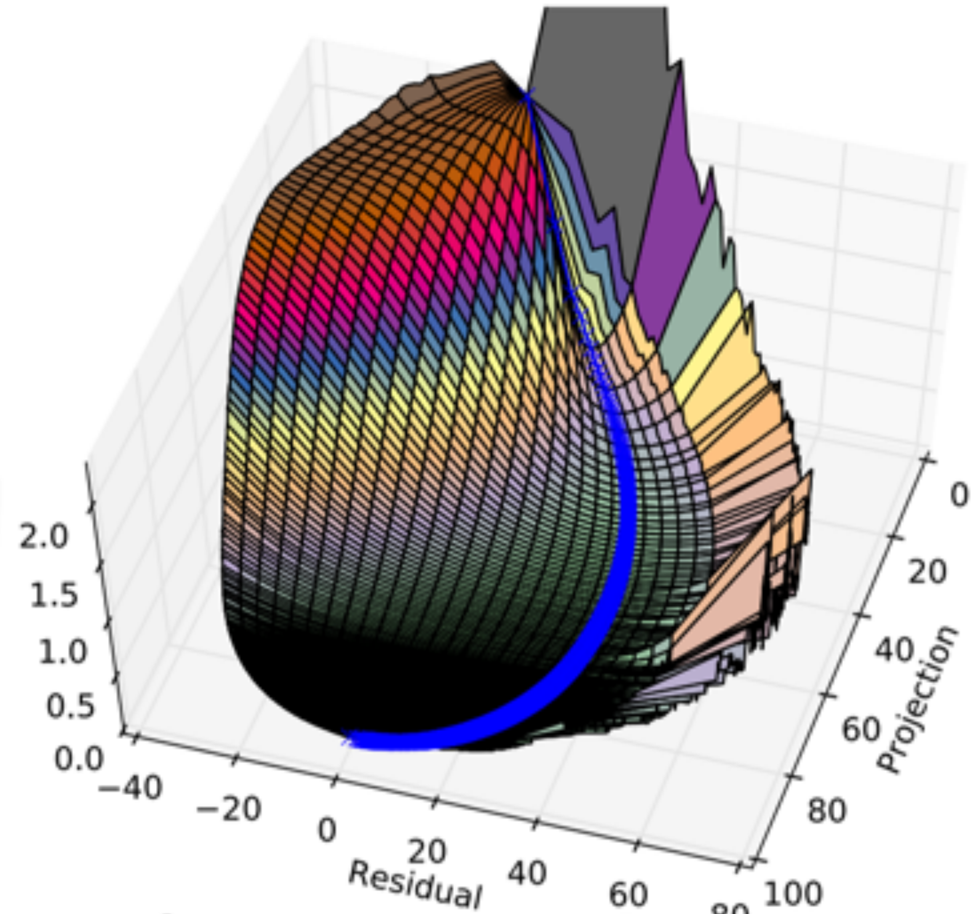


(Goodfellow 2015)

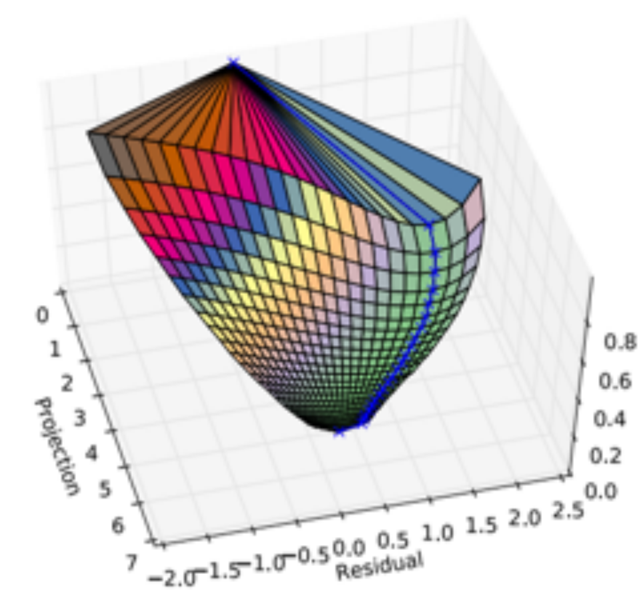
3-D plots without obstacles



LSTM



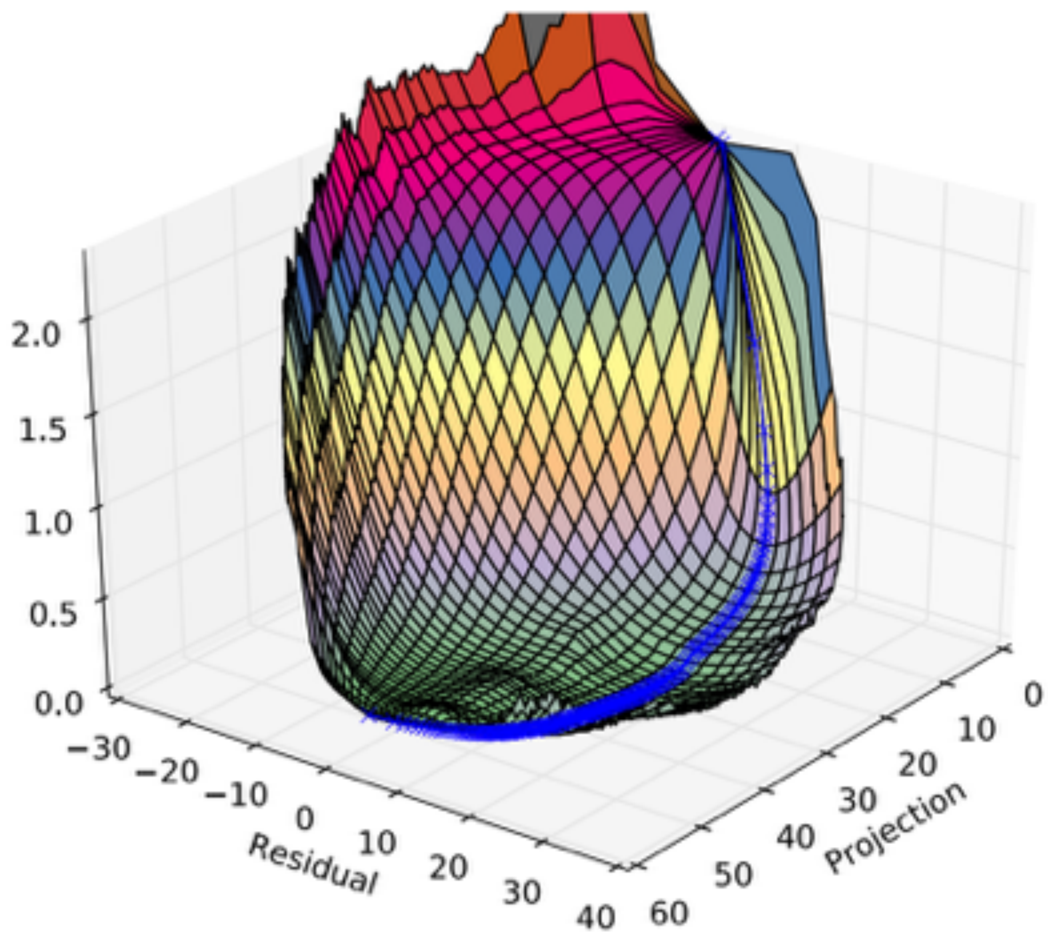
Adversarial
ReLUs



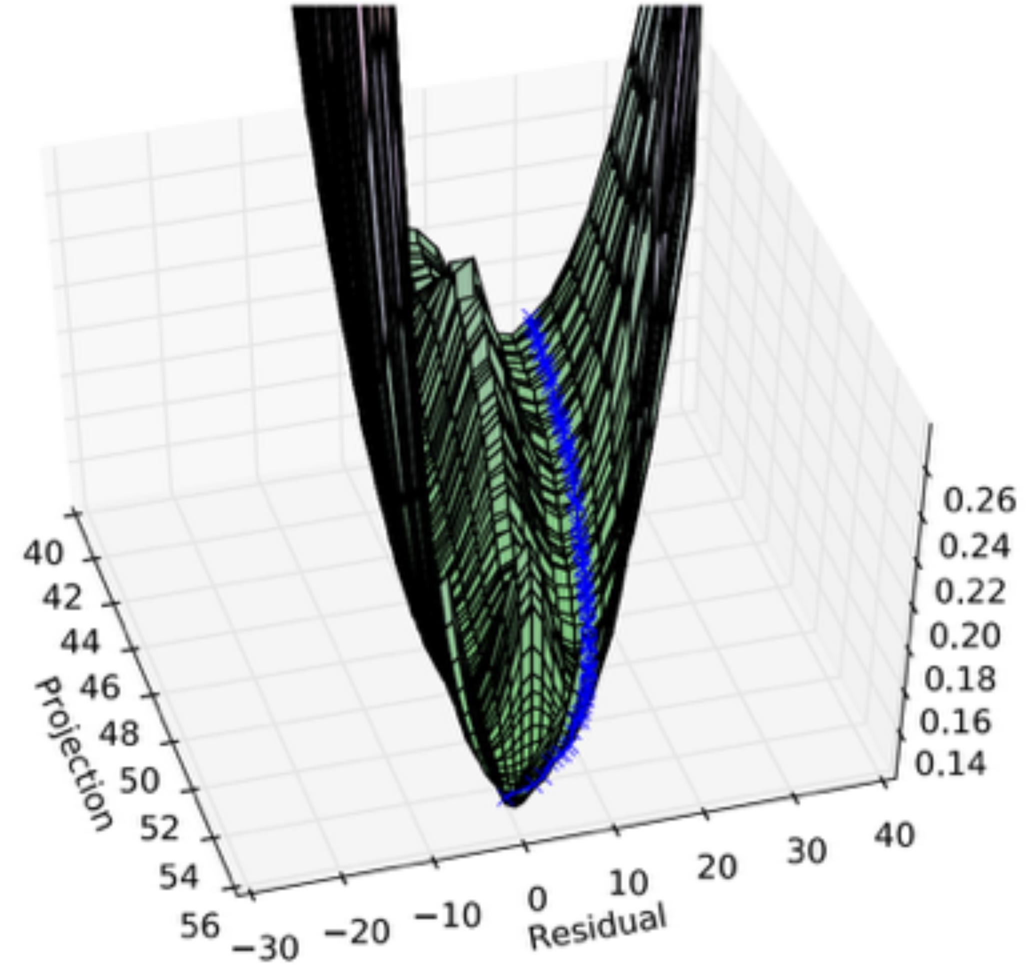
Factored Linear

(Goodfellow 2015)

3-D plot of adversarial maxout



SGD naturally exploits negative curvature!



Obstacles!

(Goodfellow 2015)

Lessons from visualizations

- For most problems, there exists a linear subspace of monotonically decreasing values

- For some problems, there are obstacles between this subspace the SGD path

- Factored linear models capture many qualitative aspects of deep network training