Deep Learning (BEV033DLE)
Lecture 8
Adaptive Optimization Methods

Czech Technical University in Prague

- Loss Landscape
  - Local optimization in high dimension
- Reparameterizations
  - Change of Basis
  - Neural Teleportation, Path SGD, Natural Gradient
- Adaptivity by Normalization
  - Trust Region Problem, Adam-like methods

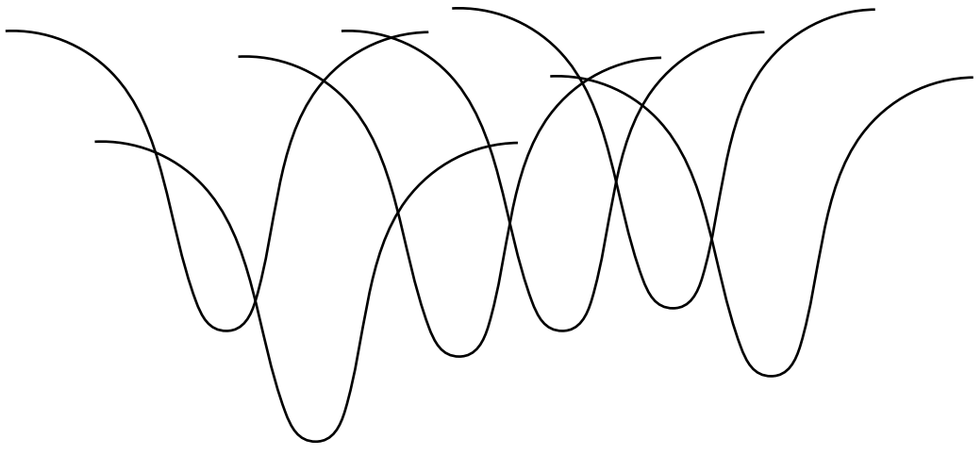
Loss Landscape

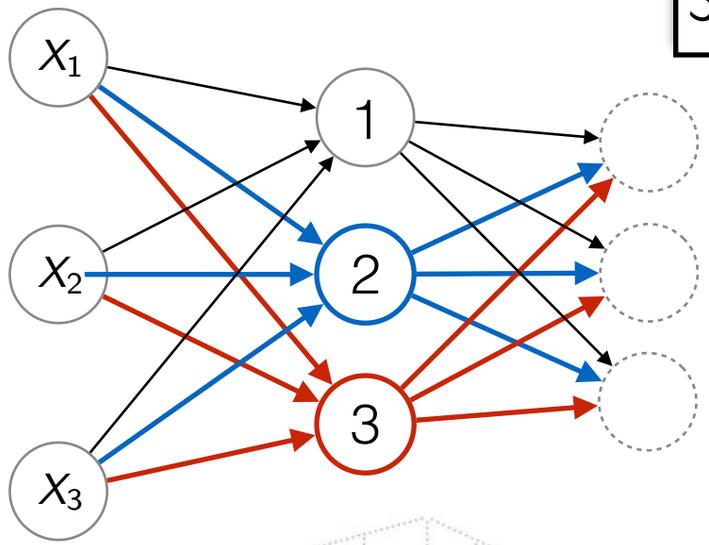
Local Minima

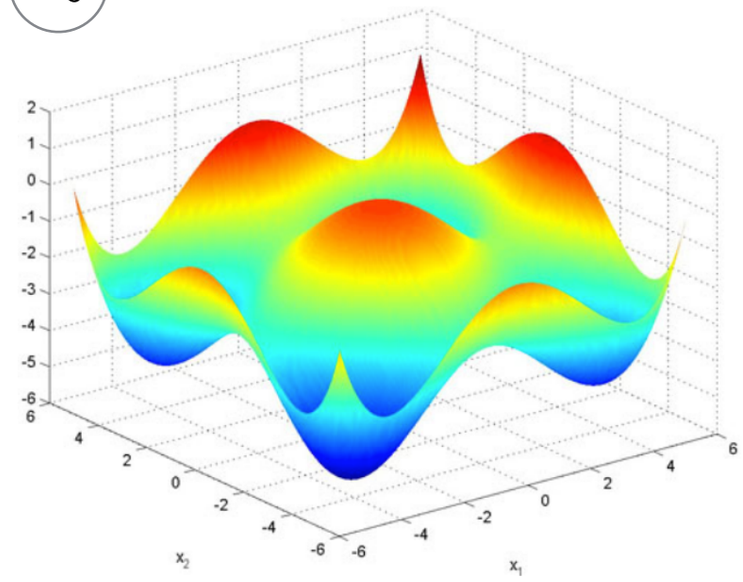
- There are several reasons for local minima
  - **Symmetries** (Permutation invariances)
    - Fully connected layer with **n** hidden units:
      n! permutations
    - Convolutional layer with **c** channels:
      c! permutations
    - In a deep network many equivalent local minima, but all of them are equally good -- no need to avoid
  - Loss function is a **sum of many non-convex terms**:

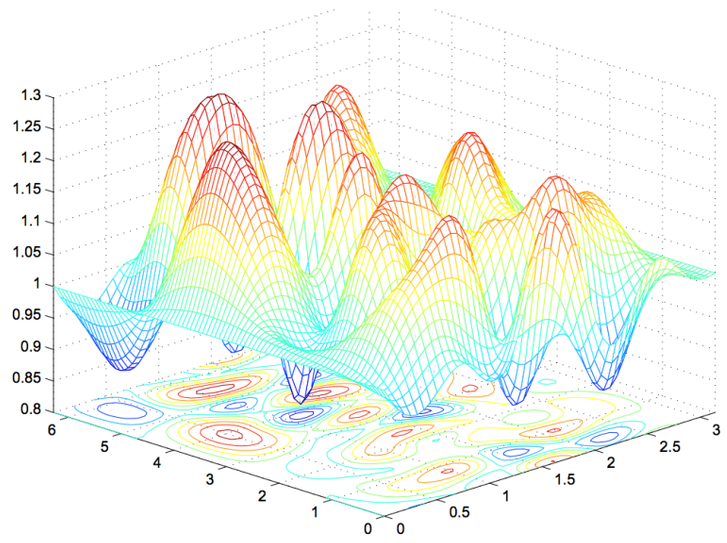
$$L(\theta) = \sum_i l(y_i, f(x_i; \theta))$$

often convex non-linear









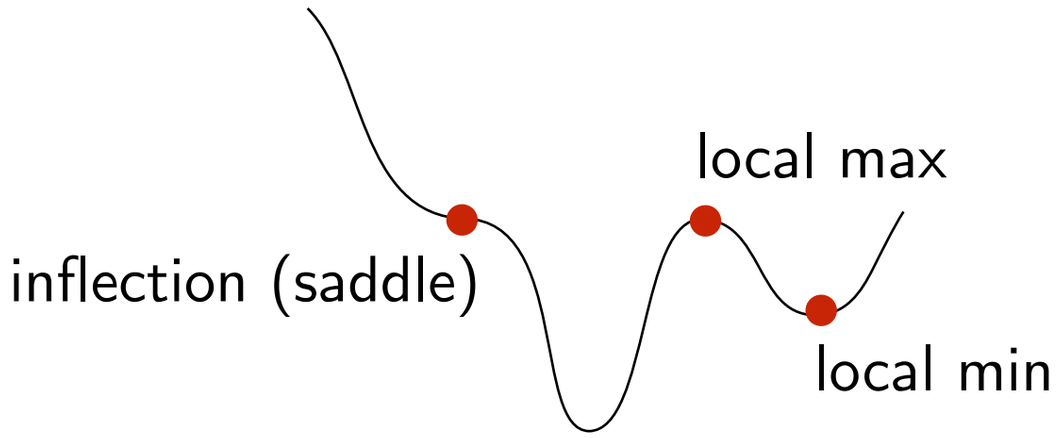
Stationary Points in High Dimensions

Let $f(x)\colon \mathbb{R}^n \to \mathbb{R}$ – differentiable,

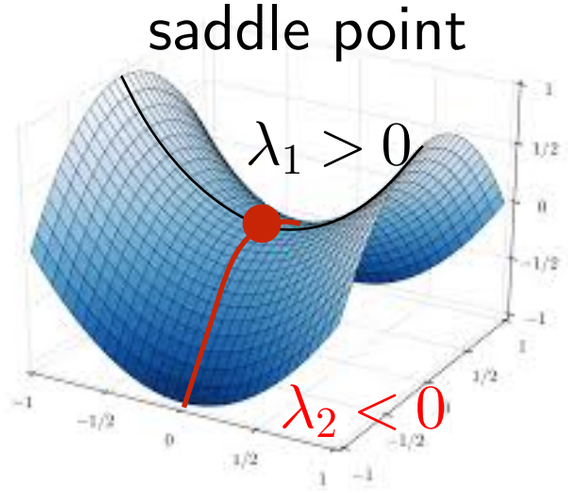
Stationary point: the gradient at x is zero

Saddle point: the gradient at x is zero but not a local extremum

1D



2D



Let $f(x+\Delta x) \approx f(x) + J\Delta x + \Delta x^{\mathsf{T}} H \Delta x$

Let H have eigenvalues $\lambda_1, \ldots \lambda_n$

Index: α — the fraction of negative eigenvalues

$\alpha = 0 \Rightarrow$ local minimum

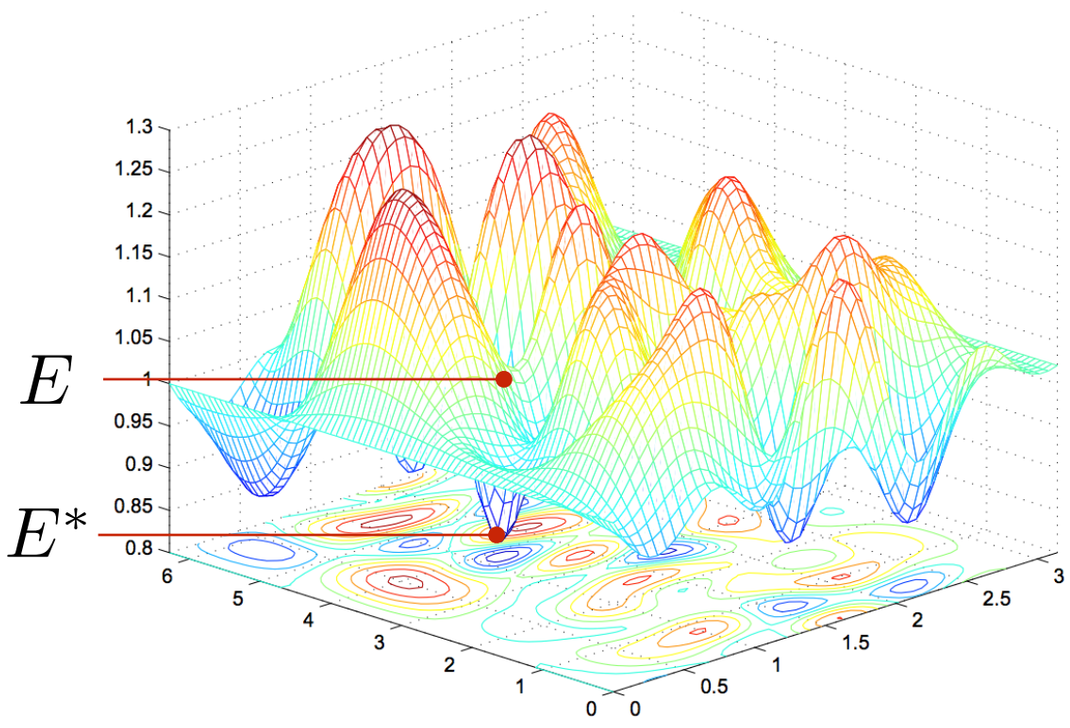
$\alpha = 1 \Rightarrow$ local maximum

$0 < \alpha < 1 \Rightarrow$ saddle point

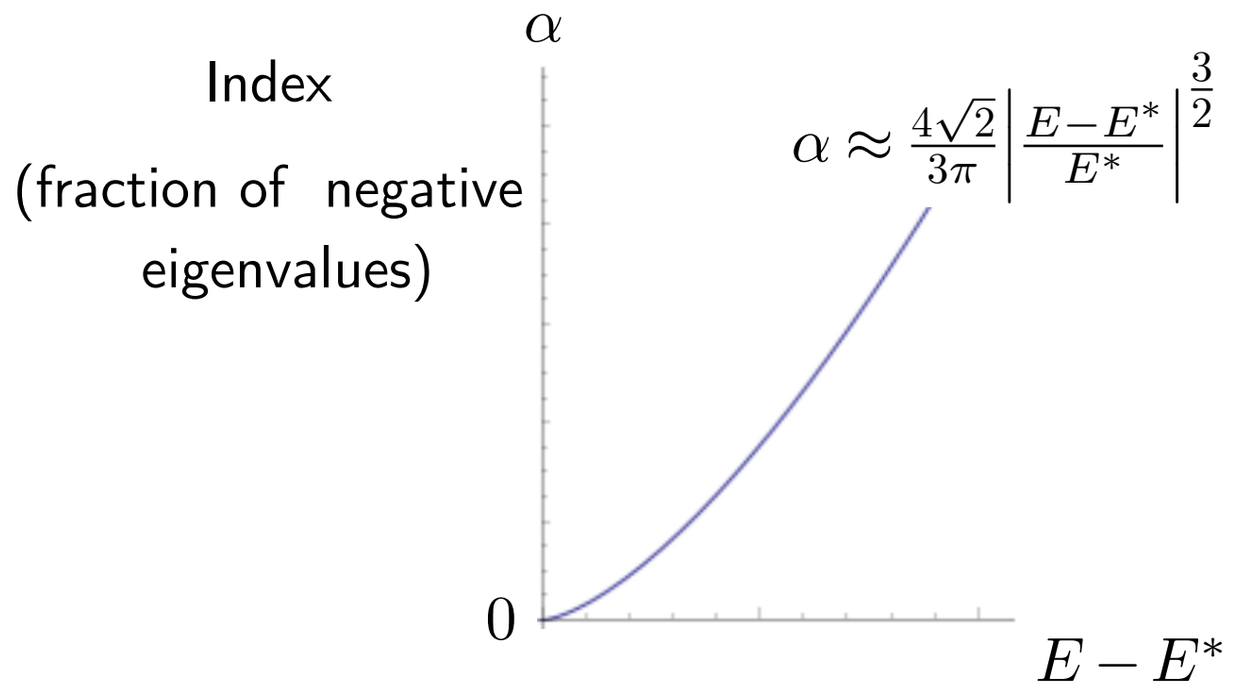
Stationary Points in High Dimensions

✦ Insights from Theoretical Physics --- **Gaussian Random Fields**:

- local minima are exponentially more rare than saddle points
- they become likely at lower energies (loss values)



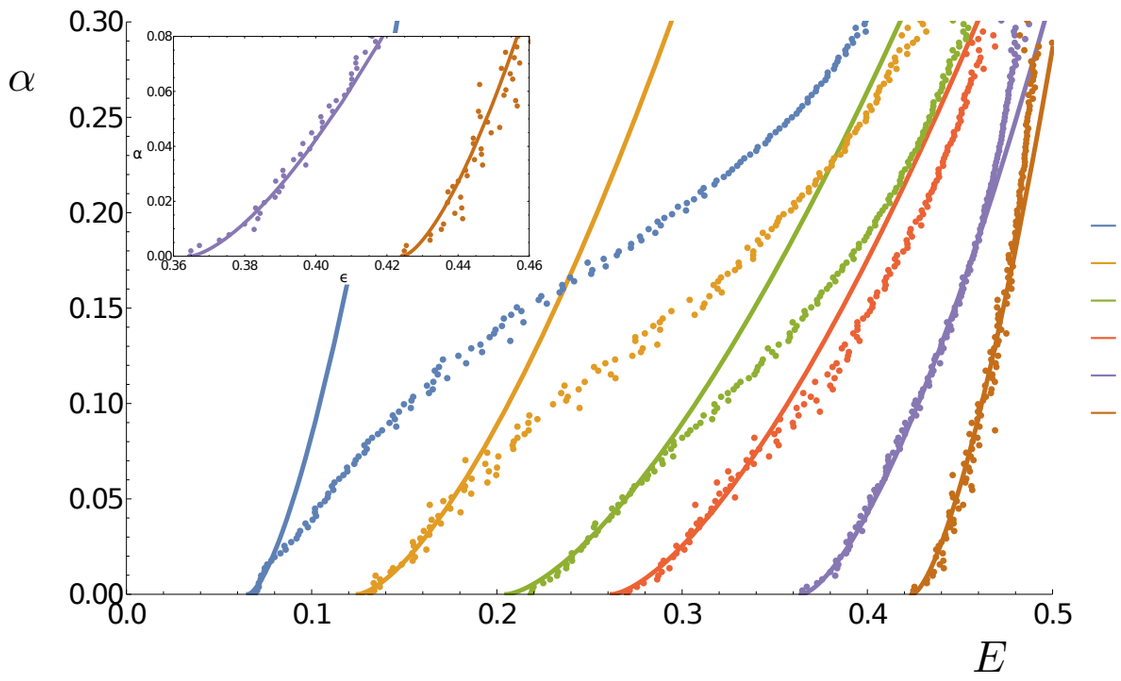
Asymptotic relation for small alpha:



[Bray & Dean (2007) The statistics of critical points of Gaussian fields on large-dimensional spaces]

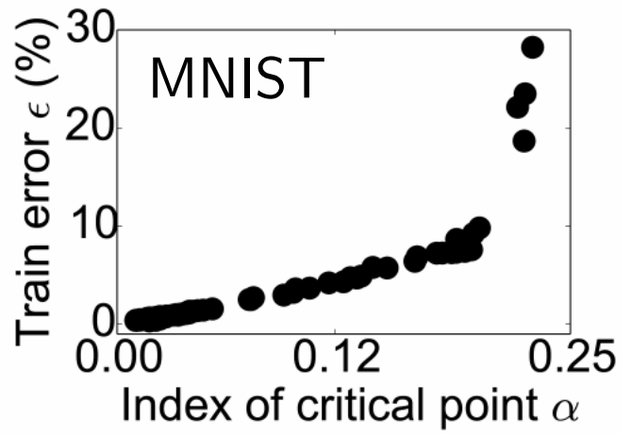
Stationary Points in High Dimensions

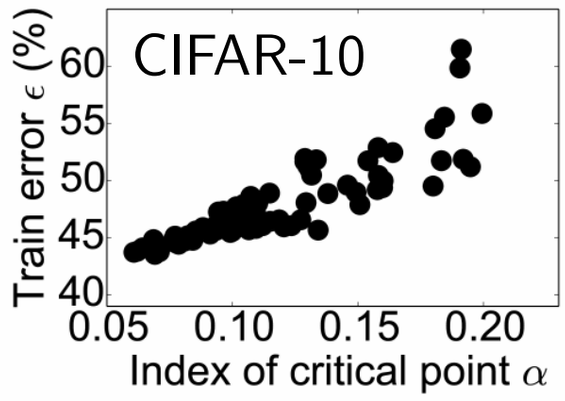
- Experimental Confirmations in Neural Networks



[Pennington & Bahri 2017]

- 1 hidden layer
- good agreement for small alpha (as expected)





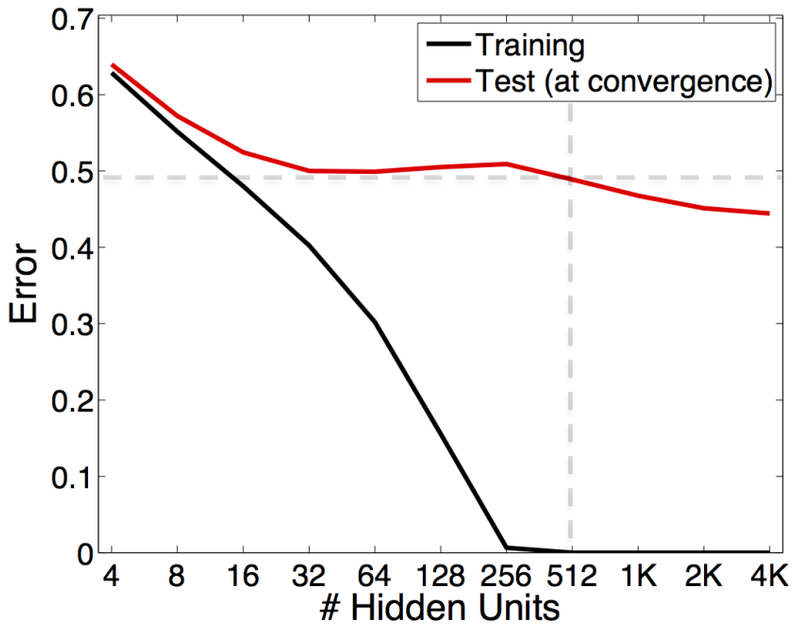
[Dauphin et. al. 2017]

[Pennington & Bahri (2017) Geometry of Neural Network Loss Surfaces via Random Matrix Theory]

[Dauphin et. al. (2017) Identifying and attacking the saddle point problem in high-dimensional non-convex optimization]

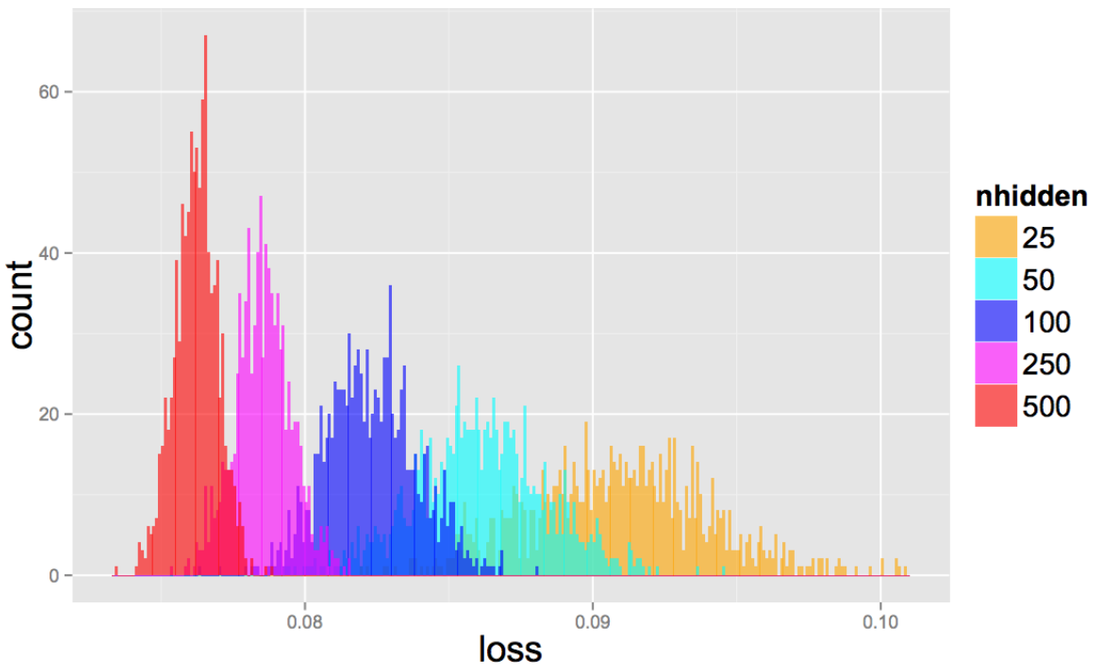
High Dimensionality Helps Optimization

Achieve 0 training error
with sufficiently large networks



[Neyshabur (2015)]

Histogram of SGD trials (MNIST)



[Choromanska et al. (2015):
The Loss Surfaces of Multilayer Networks]

✦ **Summary**:

- Local minima are rare and appear to be good enough
- But we need (highly) over-parameterized models to have this easy training
- We hope that over-parameterized models will still generalize well
- Maybe, optimization should worry a bit about efficiency around saddle points

Reparameterizations

Recall: GD Under Reparameterization

- Let $f : \mathbb{R}^n \to \mathbb{R}$
- Change of coordinates (linear reparameterization) $\tilde{w} = A^{-1}w$
  - Reparameterized function: $g(\tilde{w}) = f(A\tilde{w}) = f(w)$
  - Gradient: $\nabla_{\tilde{w}} g(\tilde{w}) = \nabla_{\tilde{w}} f(A\tilde{w}) = A\nabla_w f(w)$
  - Satisfies: $\langle \nabla_w f(w), w \rangle = \langle \nabla_{\tilde{w}} g(\tilde{w}), \tilde{w} \rangle$
  - GD in reparameterized coordinates:

    $\tilde{w}_{t+1} = \tilde{w}_t - \alpha \nabla_{\tilde{w}} g(\tilde{w}_t)$

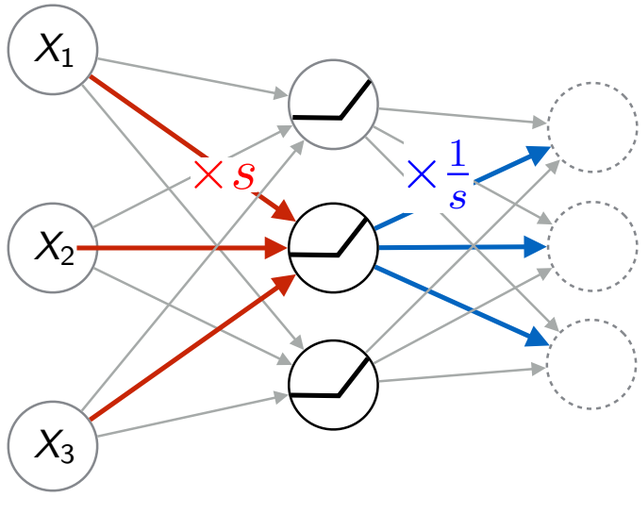
    is equivalent to preconditioned GD:

    $w_{t+1} = w_t - \alpha \left(AA^\mathsf{T}\right) \nabla_w f(w_t)$

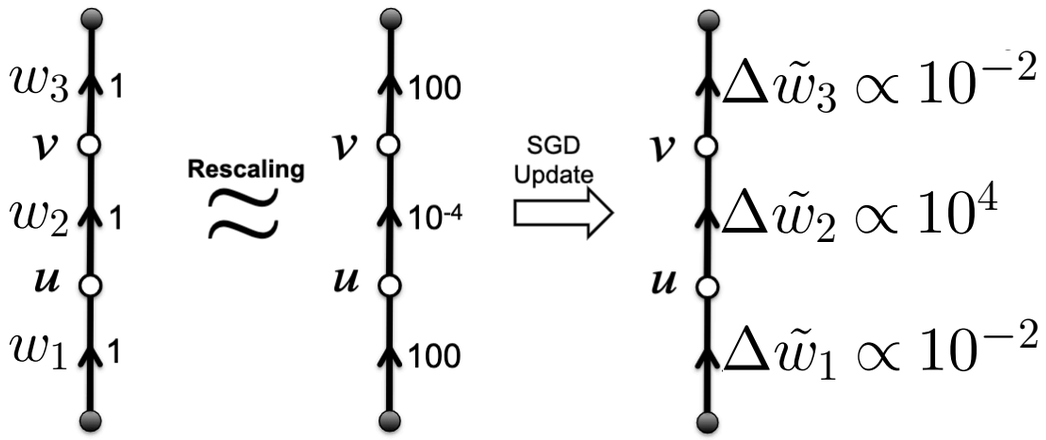
- Example: $\tilde{w} = sw$
  - Reparameterized function: $g(\tilde{w}) = f(\frac{\tilde{w}}{s})$
  - Gradient: $\nabla_{\tilde{w}} g(\tilde{w}) = \frac{1}{s} \nabla_w f(w)$
  - GD in $\tilde{w}$ is equivalent to: $w_{t+1} = w_t - \alpha \frac{1}{s^2} \nabla_w f(w_t)$

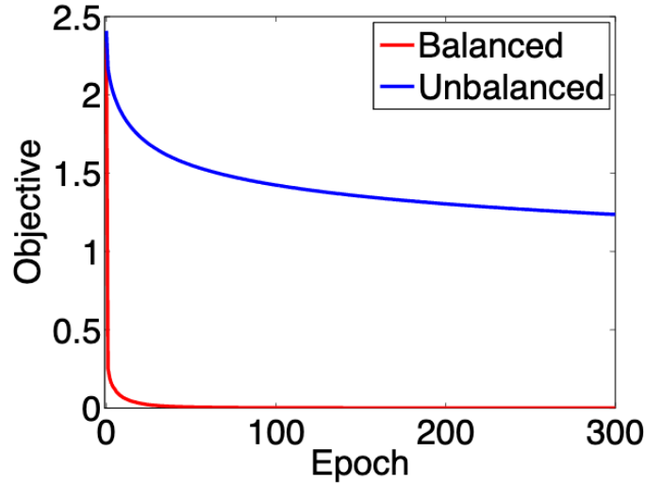
Example: Invariant Reparameterizations

- Special case,
  - mapping $w \mapsto \tilde{w}$, preserving the function value: $f(\tilde{w}) = f(w)$
  - ReLU activation function is *1-homogenous*
    for $s > 0$ $\mathrm{ReLU}(sx) = \max(0, sx) = s\max(0, x)$
  - scale can be different per unit (per channel in conv network)
  - $f(\tilde{w}) = f(w)$
  - $\frac{df(\tilde{w})}{d\tilde{w}} = \frac{df(w)}{dw}\frac{dw}{d\tilde{w}}$



- Example:





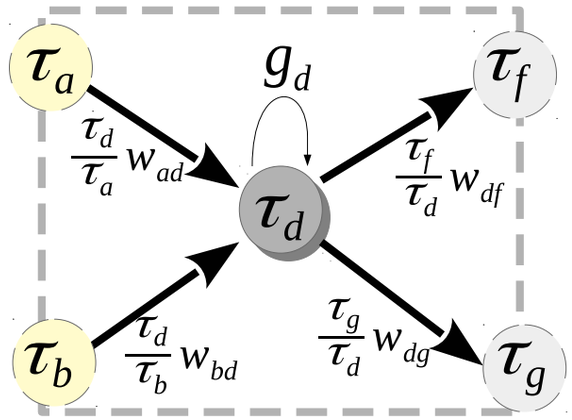
(a) Training on MNIST

- Unbalanced initialization (like in the example above):
  - pre-activation statistics are changed, but this has no effect
  - preconditioning of GD, changing the local learning rate

[Neyshabur et al. (2015) Path-SGD: Path-Normalized Optimization in Deep Neural Networks]

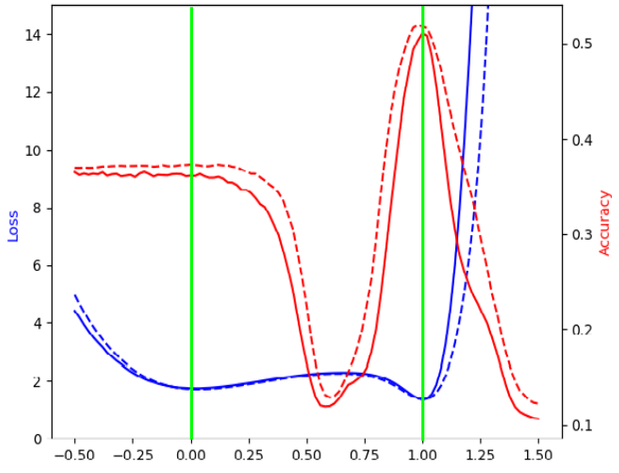
Example: Neural Teleportation

- Neural Teleportation is a change of basis:
  - assign scale τ to each neuron (at random)
  - change weights as $\tilde{w}_{i\to j} = \frac{\tau_j}{\tau_i} w_{i\to j}$
  - change activation function as $\tilde{\rho}(x) = \tau_i \rho(\frac{x}{\tau_i})$
  - the resulting network $g(\tilde{w})$ with activations $\tilde{\rho}$ is equivalent to the original network $f(w)$ with activations ρ

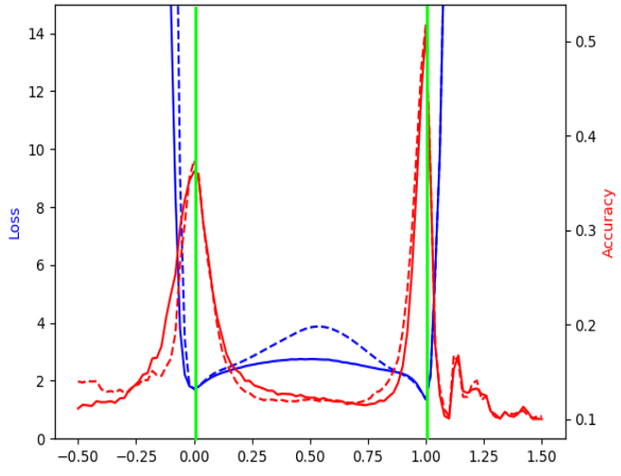


- Can change flatness of minima and learning dynamics (improves?):

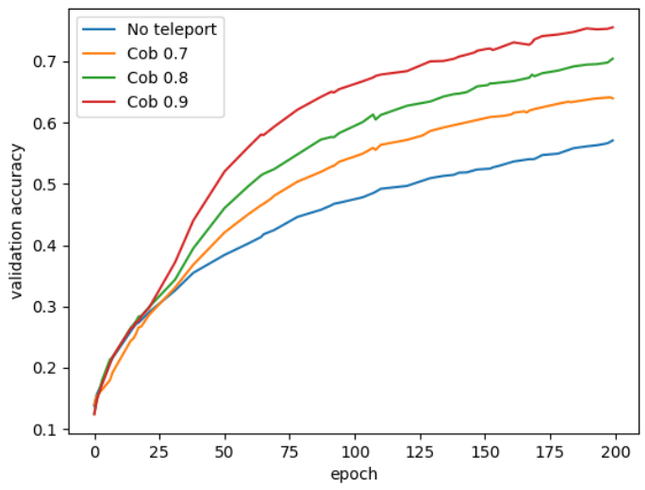
before teleport



after teleport



teleport at epoch 30, ResNet on CIFAR10



[Armenta et al. (2021) Neural Teleportation]

- We know why GD changes:
  - Gradient: $\frac{dg(\tilde{w})}{d\tilde{w}_{ij}} = \frac{df(w)}{dw_{ij}} \frac{\tau_i}{\tau_j}$
  - GD equivalent to: $w_{ij}^{t+1} = w_{ij}^t - \alpha \left(\frac{\tau_i}{\tau_j}\right)^2 \nabla_{w_{ij}} f(w^t)$ different (randomized) learning rates per weight

Path-SGD

- Idea: consider path-based divergence:

$$D(w, w^t) = \left(\sum_{\text{path } \tau} \left(\prod_{\text{edge } e \in \tau} w_e - \prod_{\text{edge } e \in \tau} w_e^t \right)^p \right)^{\frac{2}{p}}$$

  invariant to rescaling (of both arguments)

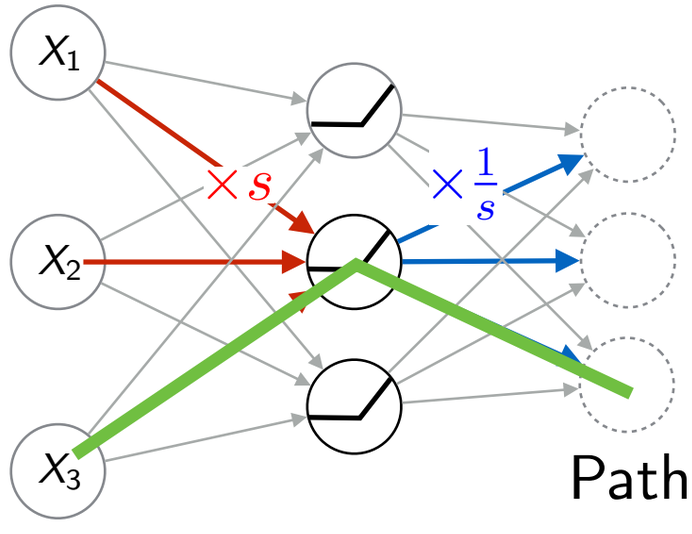
- Proximal step:

$$w^{t+1} = w^t + \underset{\Delta w}{\operatorname{argmin}} \left[\langle \nabla_w f(w^t), \Delta w \rangle + \frac{1}{2\alpha} D(w^t + \Delta w, w^t) \right]$$

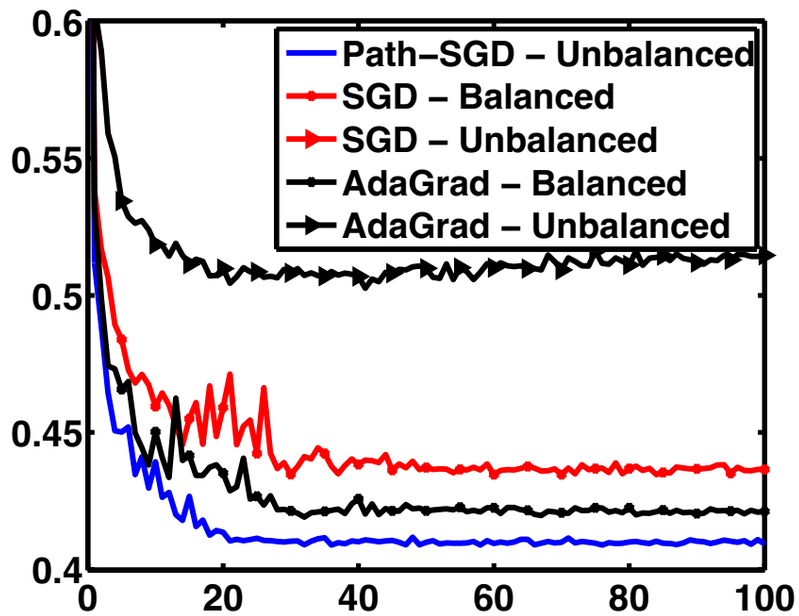
  Δw equivariant to rescaling —

  it does not matter in which reparameterization we make steps,

  we get equivalent sequences (and performance)



- Limitations:
  - Hard to solve exactly =>
    efficient approximation (equivariance preserved)
  - Specialized to ReLU / Leaky ReLU networks
  - Non-residual?



[Neyshabur et al. (2015) Path-SGD:
Path-Normalized Optimization in Deep Neural Networks]

Natural Gradient

- Setup:
  - Probabilistic predictor $p_\theta(y|x)$
  - Maximum likelihood learning: $\mathcal{L}(\theta) = -\mathbb{E}_{(x,y)\sim p^*} \log p_\theta(y|x) \to \min_\theta$
- Proximal problem:

$$\min_{\Delta\theta} \Big(\langle \nabla_\theta \mathcal{L}(\theta_t), \Delta\theta \rangle + \tfrac{1}{\alpha} \mathbb{E}_{x\sim p^*} \underbrace{\Big[D_{\mathrm{KL}}(p_{\theta_t}(\cdot|x) \,\|\, p_{\theta_t+\Delta\theta}(\cdot|x)) \Big]}_{\text{invariant to reparameterizations}} \Big)$$

  - optimal step $\Delta\theta$ is equivariant to reparameterizations
  - If $\alpha \to 0$ then $\Delta\theta \to 0$ and

    $\mathbb{E}_{x\sim p^*} \Big[D_{\mathrm{KL}}(p_{\theta_t}(\cdot|x) \,\|\, p_{\theta_t+\Delta\theta}(\cdot|x)) \Big] \to \frac{1}{2} \Delta\theta^\mathsf{T} F(\theta_t) \Delta\theta + o(\|\Delta\theta\|^2)$ – locally quadratic,

    where F is the (expected) Fisher information matrix:

    $F(\theta) = \mathbb{E}_{x\sim p^*} \Big[\mathbb{E}_{y\sim p_\theta(y|x)} \Big[\Big(\frac{d \log p_\theta(y|x)}{d\theta} \Big)^{\otimes 2} \Big] \Big]$

    Natural Gradient step: $\Delta_\theta = -\alpha F(\theta_t)^{-1} \nabla_\theta \mathcal{L}(\theta_t)$
- Practical aspects:
  - In principle, need only the gradients to approximate
  - Results in a large matrix, difficult to maintain and inverse, needs to be approximated

Adaptivity by Normalization

Trust Region Problem

- Similar to proximal problem, but constrained optimization form:

$$\min_{\|\Delta x\|_2 \leq \varepsilon} \left(f(x_0) + J\Delta x \right)$$

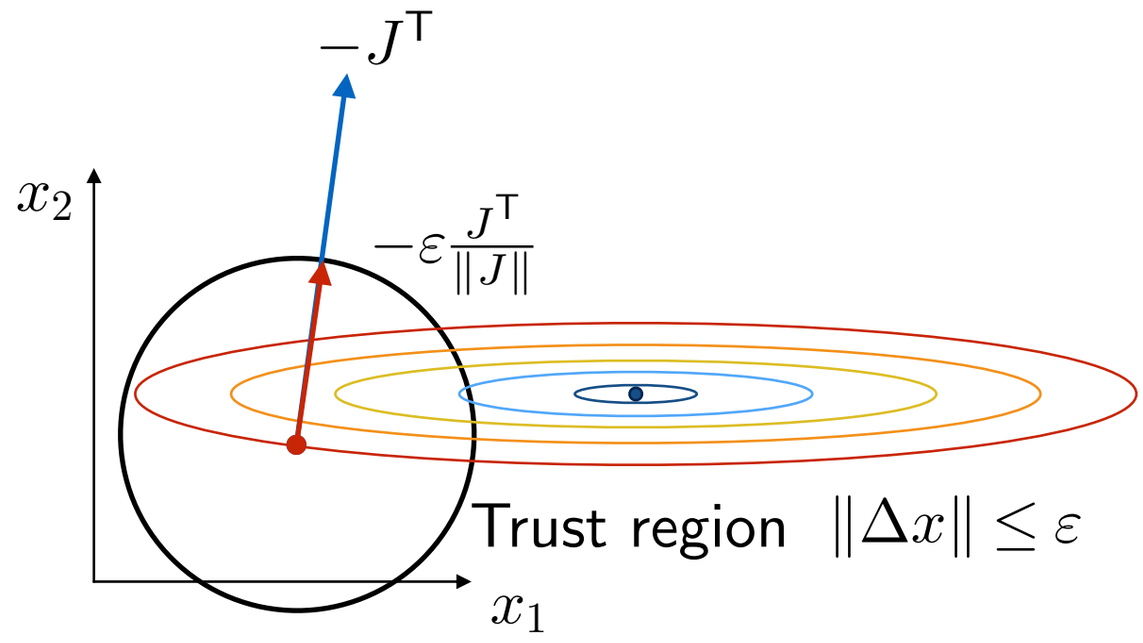
Equivalent to:

$$\max_{\lambda \geq 0} \min_{\Delta x} \left(J\Delta x + \lambda(\|\Delta x\|_2^2 - \varepsilon^2) \right)$$

Step direction: $\Delta x = -\frac{1}{2\lambda} J^\mathsf{T}$

$\|\Delta x^\mathsf{T}\|^2 = \varepsilon^2 \rightarrow \lambda = \frac{1}{2\varepsilon}\|J\|_2$

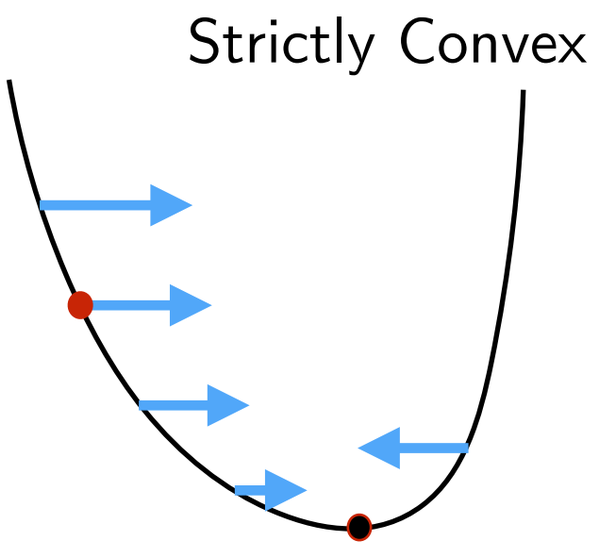
Trust region step: $\Delta x = -\varepsilon \frac{J^\mathsf{T}}{\|J\|_2}$



- How does it behave under change of basis?
  - In reparameterized coordinates: $\tilde{x} = sx$, $\tilde{J} = \frac{1}{s}J$, $\tilde{\Delta} x = -\varepsilon \frac{\tilde{J}}{\|\tilde{J}\|_2}$
  - Equivalent to $\Delta x = \frac{1}{s}\Delta\tilde{x} = -\varepsilon \frac{1}{s}\frac{\tilde{J}}{\|\tilde{J}\|_2} = -\varepsilon \frac{1}{s}\frac{J}{\|J\|_2}$ – not equivariant, but better than $\frac{1}{s^2}$
- An update $\Delta x = -\varepsilon \frac{J}{\|J\|_2^2}$ would be equivariant but not good in any space
- Second order methods $\Delta x = -\varepsilon G^{-1} J$ are equivariant but more costly (Newton, Gauss-Newton, Natural Gradient)

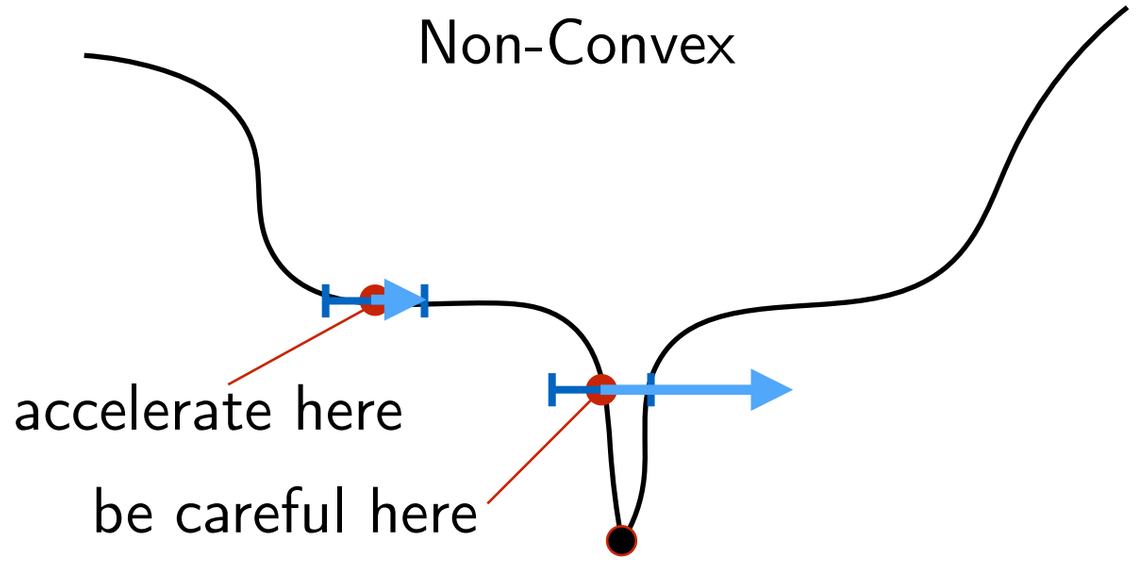
Differences of Convex vs. Non-Convex

Why to step proportionally to the gradient:



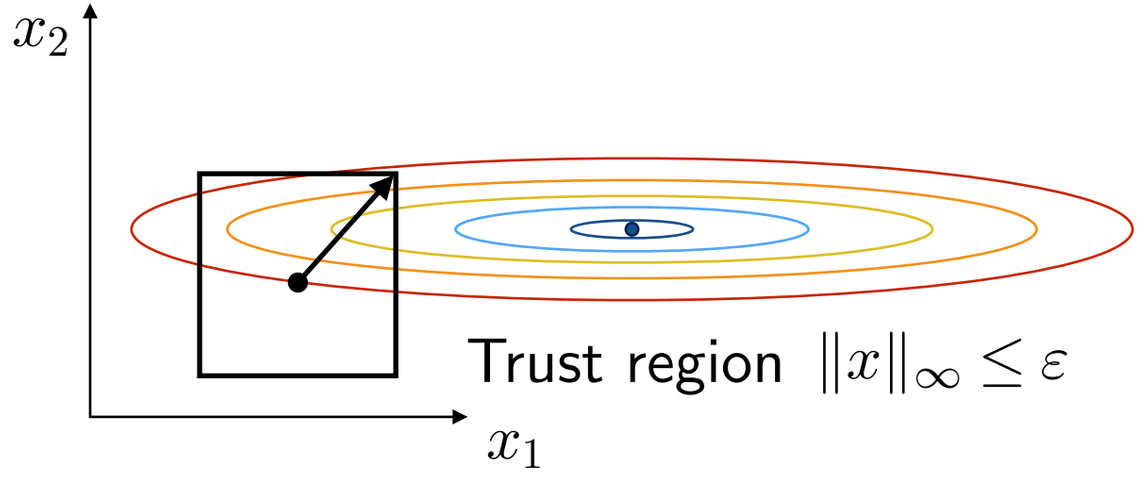
- No other stationary points than global minima
- **The further we are from the optimum, the larger is the gradient**: $\exists \mu > 0$
  - $\|\nabla f(x)\|^2 \geq \mu(f(x) - f^*)$
  - $\|\nabla f(x)\| \geq \mu|x - x^*|$
- Negative gradient points towards the optimum:
  - $\langle -\nabla f, x^* - x \rangle \geq f - f^* + \tilde{\mu}\|x - x^*\|^2$
  - Optimization need not be monotone in f

Why to normalize:



- Gradient carries no global information
  - Need bigger steps where gradient and curvature are low
  - Need smaller steps when gradient and curvature are high
- Makes sense to use **trust region steps**:
  - $\Delta x = -\frac{\nabla f}{\|\nabla f\|}$
  - If the trust region is ok, should guarantee a steady progress

Box Trust Regions



- This time solve for step as:
- $\min_{\|\Delta x_i\| \leq \varepsilon \ \forall i} \left(f(x_0) + J\Delta x\right)$

  (In overparametrized models expect many parameters to have independent effect)
- Equivalent to:

  $\max_{\lambda \geq 0} \min_{\Delta x} \left(J\Delta x + \sum_i \lambda_i (\|\Delta x_i\|^2 - \varepsilon^2)\right)$

  $2\lambda_i \Delta x_i = -J_i$

  Step direction: $\Delta x_i = -\frac{1}{2\lambda_i}(\nabla f(x))_i$

  Trust region step: $\Delta x_i = -\varepsilon \frac{(\nabla f(x))_i}{|(\nabla f(x))_i|}$

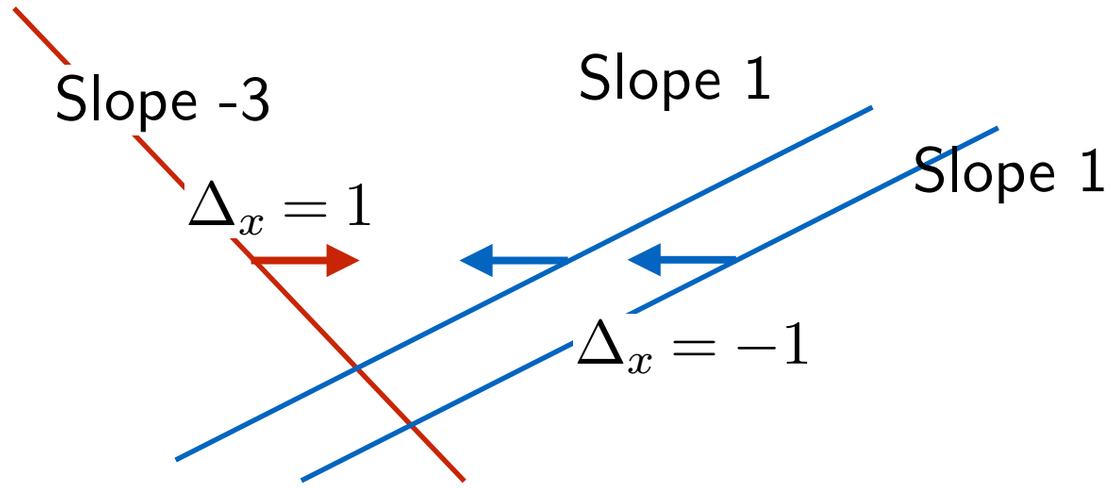
Non-Convex Stochastic

- Trust region steps: $\Delta x_i = -\varepsilon \frac{(\nabla f(x))_i}{|(\nabla f(x))_i|}$
- Problem: breaks in the stochastic setting
- Example

  $f(x) = (-3x) + (x) + (x+1)$, chose 1 summand at a time with equal probability

  If we normalize stochastic gradients,

  will move in the wrong direction!



- Want the steps to follow the descent direction on average
  - Cannot adjust the stochastic gradient "too much nonlinearly"
  - This example was used to show that Adam may fail to converge to a stationary point and motivated theoretical improvements

Adaptive Methods

- **Practical Solution**: approximate expectations with running averages:

  $$\Delta x = -\varepsilon \frac{\mathbb{E}[\nabla f]}{\|\mathbb{E}[\nabla f]\|}$$

  Furhter approximate $\|\mathbb{E}[\nabla f]\| = \sqrt{(\mathbb{E}[\nabla f])^2} \leq \sqrt{(\mathbb{E}[(\nabla f)^2])}$

- **Adagrad**:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\varepsilon}{\sqrt{t}} \frac{\tilde{g}_{t,i}}{\sqrt{\mathsf{Mean}\left(\tilde{g}^2_{1:t,i}\right)}}$$

- **RMSProp**:

$$\theta_{t+1,i} = \theta_{t,i} - \varepsilon \frac{\tilde{g}_{t,i}}{\sqrt{\mathsf{EWA}\left(\tilde{g}^2_{1:t,i}\right)}}$$

- **Adam**:

$$\theta_{t+1,i} = \theta_{t,i} - \varepsilon \frac{\mathsf{EWA}_{\beta_1}\left(\tilde{g}_{1:t,i}\right)}{\sqrt{\mathsf{EWA}_{\beta_2}\left(\tilde{g}^2_{1:t,i}\right)}}$$

- In Adagrad:
  $\frac{1}{\sqrt{t}}$ guarantees convergence. Other methods would also need this in theory but are typically presented and used with constant ε
  The flat average appears not very practical
- In Adam:
  EWA with $\beta_1 = 0.9$ works as common momentum (20 batches averaging)
  EWA with $\beta_2 = 0.999$ (2000 batches averaging) makes the normalization smooth enough