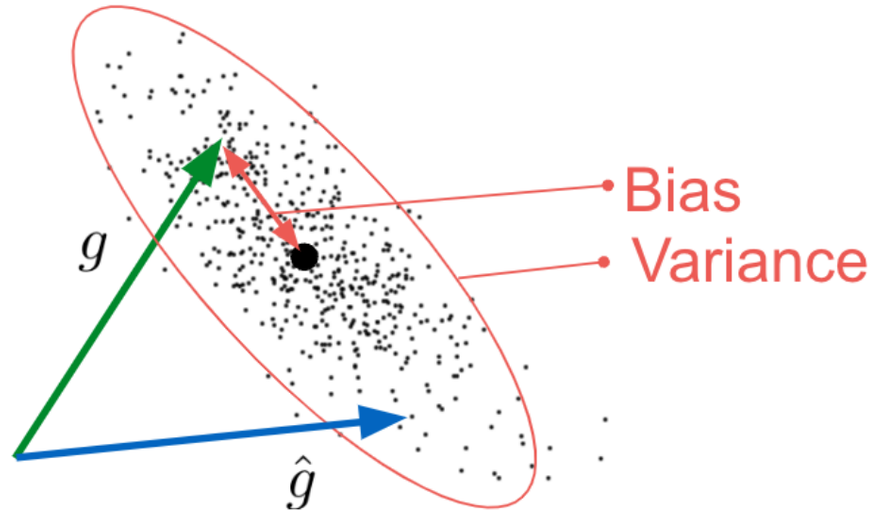
Advanced Deep Learning (BEV033DLA)
Lecture 6
Stochastic Gradient Descent

Czech Technical University in Prague

- **SGD**
  - Simple Analysis
    - Progress vs variance work and learning rate
    - Learning rate schedule
  - Variance reduction and momentum
- **Extra***
  - Implicit Regularization of SGD
  - p-norm steepest descent

SGD: General Setup

- Problem Setup:
  - **Data generator** p^*:
    - finite training set of size n plus sampler (plus randomized augmentation)
    - simulation, stream
  - Predictor: $f(x;\theta)$, loss: $L(\theta) = \mathbb{E}_{(x,y)\sim p^*}[l(y, f(x;\theta))]$
- **Gradient Descent**:
  - $g_t = \nabla_\theta L(\theta_t)$
  - $\theta_{t+1} = \theta_t - \alpha_t g_t$, α_t is learning rate
  - If the dataset is very large – lots of computations to make a small step
- **Stochastic Gradient Descent**:
  - Draw a batch of data $(x_i, y_i)_{i=1}^B$ from p^*
  - $\tilde{g}_t = \nabla_\theta \left(\frac{1}{B} \sum_i l(y_i, f(x_i, \theta_t)) \right)$
  - $\theta_{t+1} = \theta_t - \alpha_t \tilde{g}_t$
  - "Noisy" gradient:
    - $\mathbb{E}[\tilde{g}_t] = \nabla_\theta L(\theta_t)$ – unbiased if batch loss is unbiased
    - $\mathbb{V}[\tilde{g}_t] = \frac{1}{B}\mathbb{V}[\tilde{g}_t^1]$, where $\tilde{g}_t^1$ is a stochastic gradient with one sample: tradeoff between accuracy of approximation and computation cost



GD vs SGD

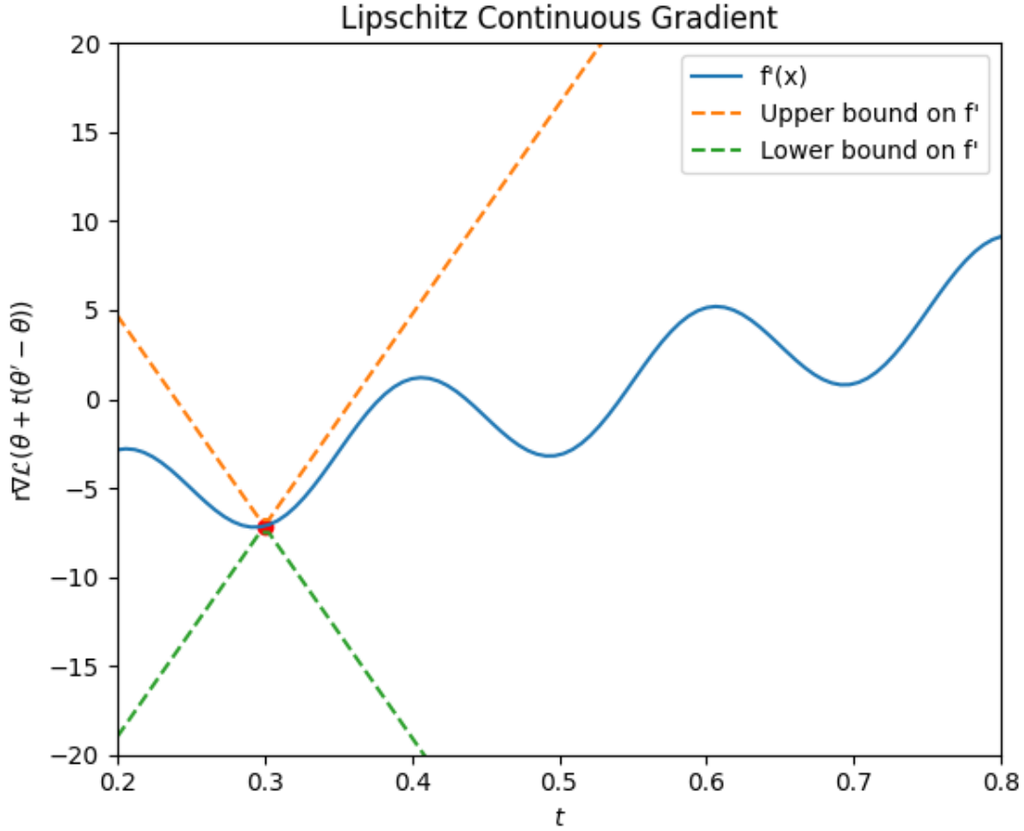
$\mathcal{L}(\theta)$

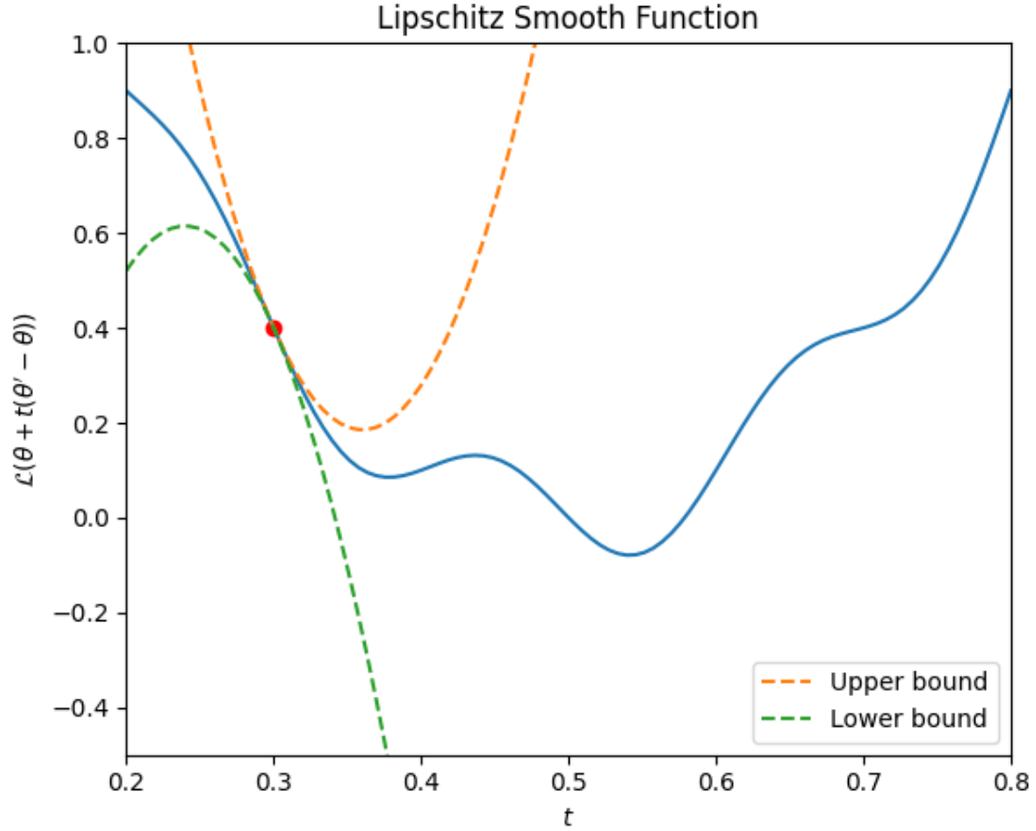
- Gradient Descent:
  - $\theta_{t+1} = \theta_t - \alpha g_t$
  - compute: $O(n)$ – full data
  - gradient variance: 0
- SGD:
  - $\theta_{t+1} = \theta_t - \alpha \tilde{g}_t$
  - compute: $O(B)$ – mini-batch
  - gradient variance: $\frac{1}{B}\sigma_0^2$

- So which one will be the fastest?

A Model to Measure Progress

- **Lipschitz smooth function**:
  - $\mathcal{L}(\theta)$ is Lipschitz smooth with constant ρ if its gradient is Lipschitz continuous: $\|\nabla\mathcal{L}(\theta') - \nabla\mathcal{L}(\theta)\| \leq \rho\|\theta - \theta'\|$ for all θ, θ'
  - Implies: $|\mathcal{L}(\theta') - \underbrace{(\mathcal{L}(\theta) + \nabla\mathcal{L}(\theta)(\theta - \theta'))}_{\text{linear approx at } \theta}| \leq \frac{\rho}{2}\|\theta_1 - \theta_2\|^2$
  - We will need the upper bound: $\mathcal{L}(\theta') - \mathcal{L}(\theta) \leq \nabla\mathcal{L}(\theta)(\theta - \theta') + \frac{\rho}{2}\|\theta_1 - \theta_2\|^2$





Work vs Variance Tradeoff

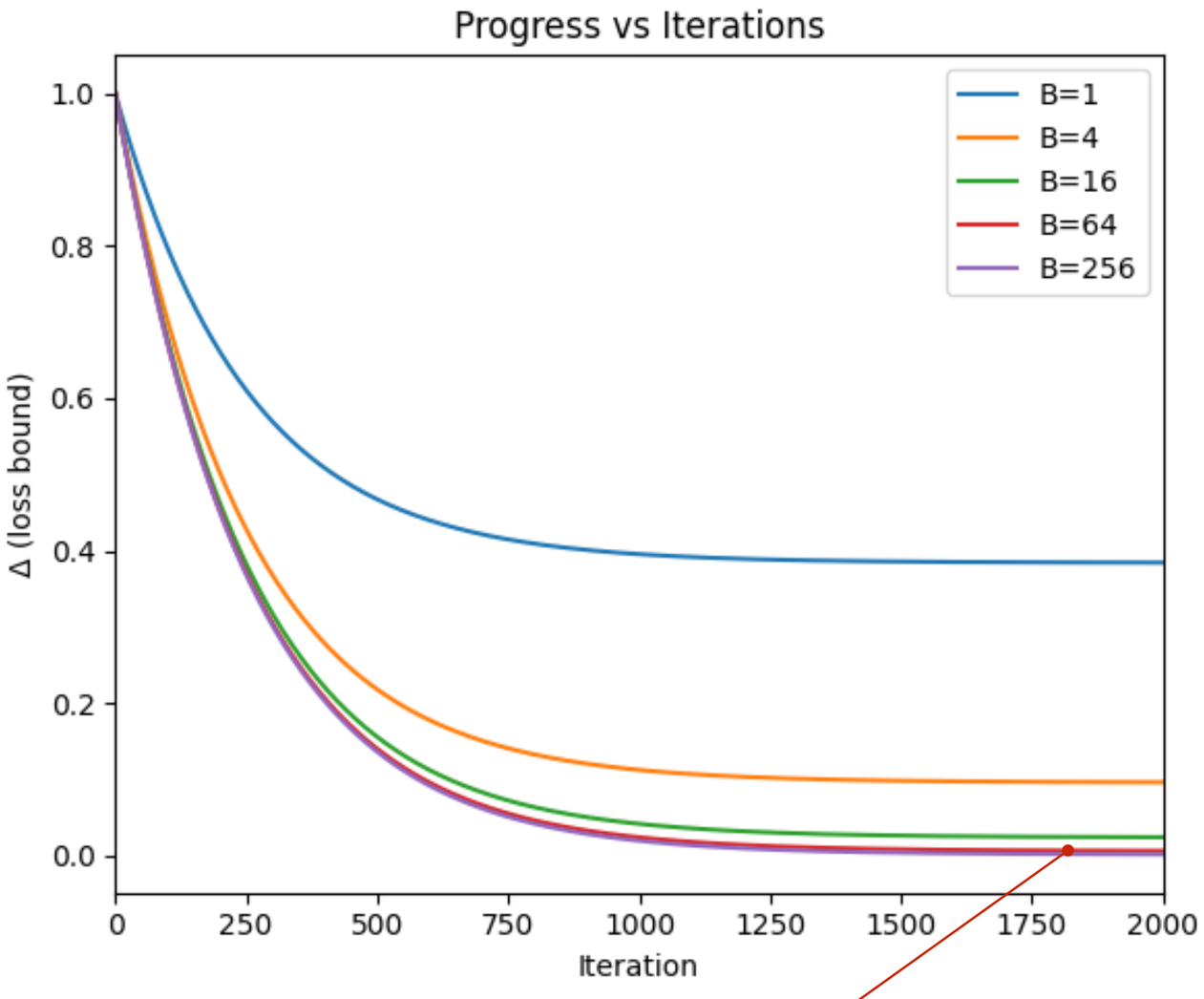
- **Lipschitz smooth function**:
  - $\mathcal{L}(\theta)$ is Lipschitz smooth with constant ρ if its gradient is Lipschitz continuous: $\|\nabla\mathcal{L}(\theta') - \nabla\mathcal{L}(\theta)\| \leq \rho\|\theta - \theta'\|$ for all θ, θ'
  - Implies: $|\mathcal{L}(\theta') - \underbrace{(\mathcal{L}(\theta) + \nabla\mathcal{L}(\theta)(\theta - \theta'))}_{\text{linear approx at } \theta}| \leq \frac{\rho}{2}\|\theta_1 - \theta_2\|^2$
  - We will need the upper bound: $\mathcal{L}(\theta') - \mathcal{L}(\theta) \leq \nabla\mathcal{L}(\theta)(\theta - \theta') + \frac{\rho}{2}\|\theta_1 - \theta_2\|^2$

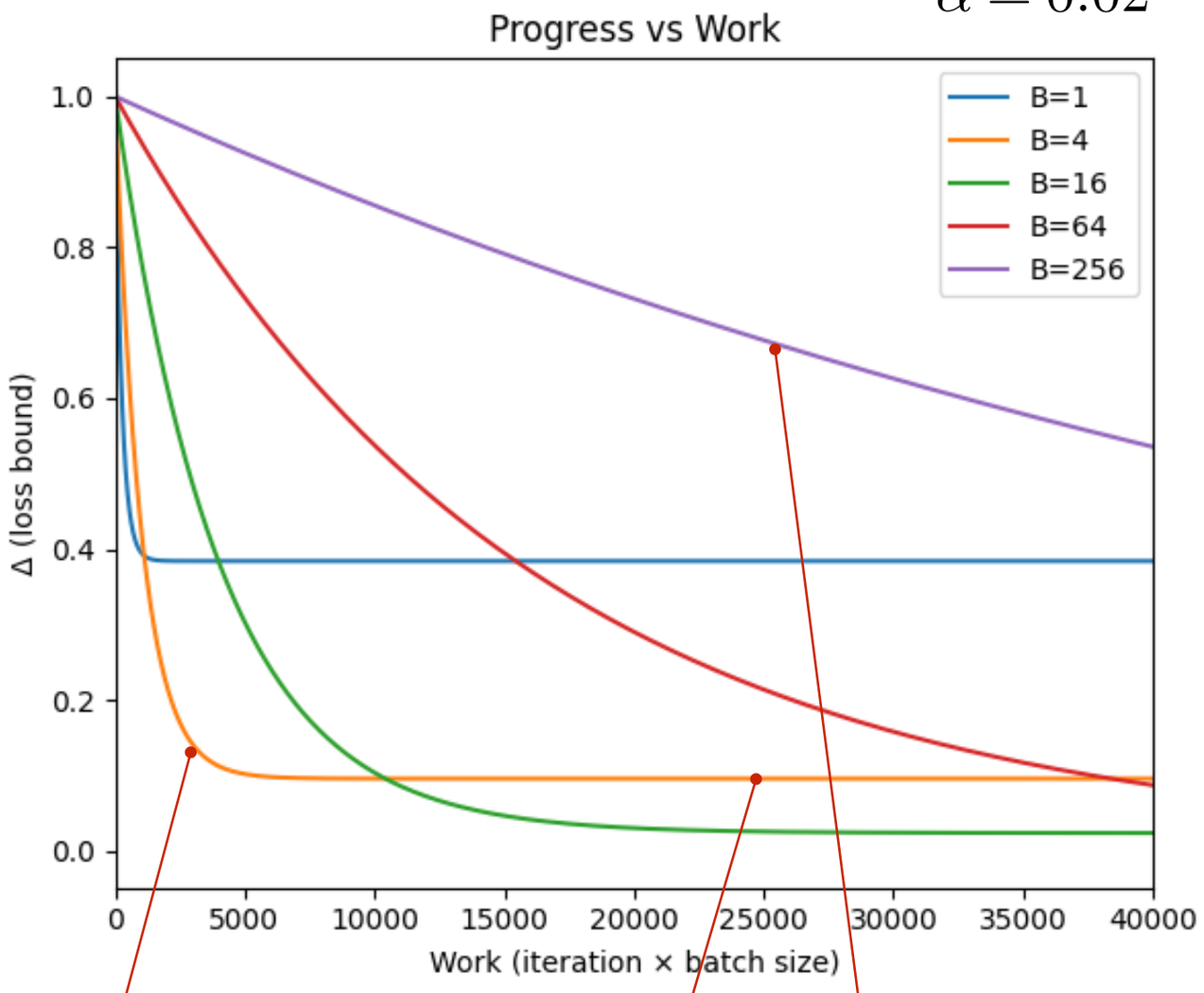
- **Improvement of SGD step** $\theta_{t+1} = \theta_t - \alpha\tilde{g}_t$:
  - $\mathcal{L}_t(\theta_t + 1) - \mathcal{L}(\theta_t) \leq -\alpha\langle g_t, \tilde{g}_t\rangle + \frac{\rho\alpha^2}{2}\|\tilde{g}_t\|^2$
  - Expected Improvement: $-\alpha\langle g_t, \mathbb{E}[\tilde{g}_t]\rangle + \frac{\rho\alpha^2}{2}\mathbb{E}\|\tilde{g}_t\|^2$
    $= -\alpha\|g_t\|^2 + \frac{\rho\alpha^2}{2}(\mathbb{V}[\tilde{g}_t] + \|g_t\|^2)$
    $\leq -(\alpha - \frac{\rho\alpha^2}{2})\|g_t\|^2 + \frac{\rho\alpha^2\sigma^2}{2}$, assuming $\mathbb{V}[\tilde{g}_t] \leq \sigma^2$
    ...
  - $\frac{1}{T}\sum_{k=0}^{T}\mathbb{E}\|g_t\|^2 \leq \underbrace{\frac{\mathcal{L}(\theta_0) - \mathcal{L}^*}{cT}}_{O(\frac{1}{T})} + \underbrace{\frac{\rho\alpha^2}{2c}\sigma^2}_{\text{Noise floor}}$, where $c = \alpha - \frac{\rho\alpha^2}{2}$

Work vs Variance Tradeoff

- Expected grad norm: $O(\frac{1}{cT}) + O(\frac{\rho\alpha^2\sigma^2}{2c})$, $c = \alpha - \frac{\rho\alpha^2}{2}$
  - Work per iteration $\propto B$ – batch size
  - Variance $\sigma^2 \propto \frac{1}{B}\sigma_0^2$, where σ_0^2 – variance for $B = 1$
  - For $\sigma^2 = 0$, we get just $O(\frac{1}{cT})$ – GD rate

$\alpha = 0.02$





diminishing returns

Fast O(1/T) rate

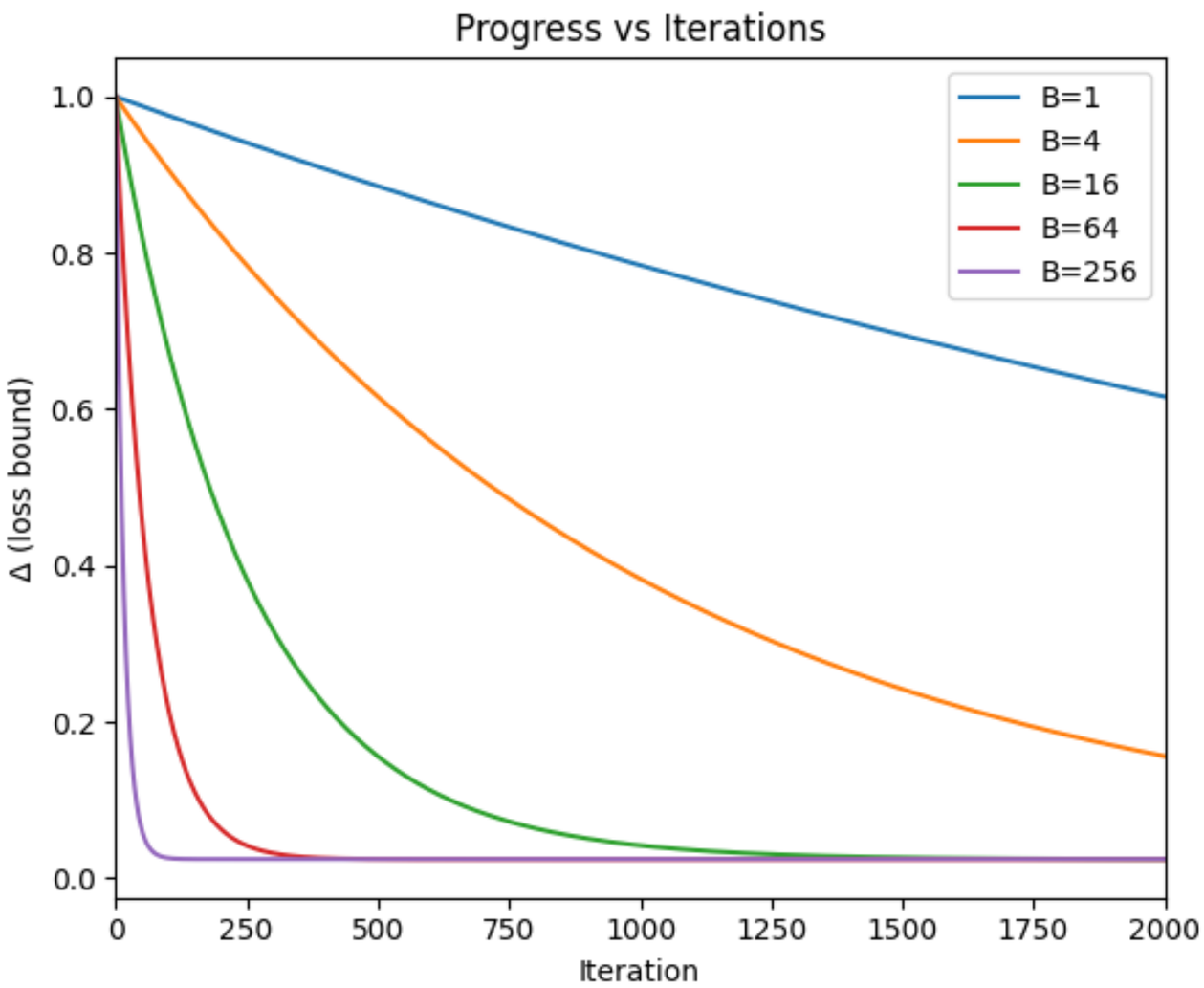
Slow O(1/T)

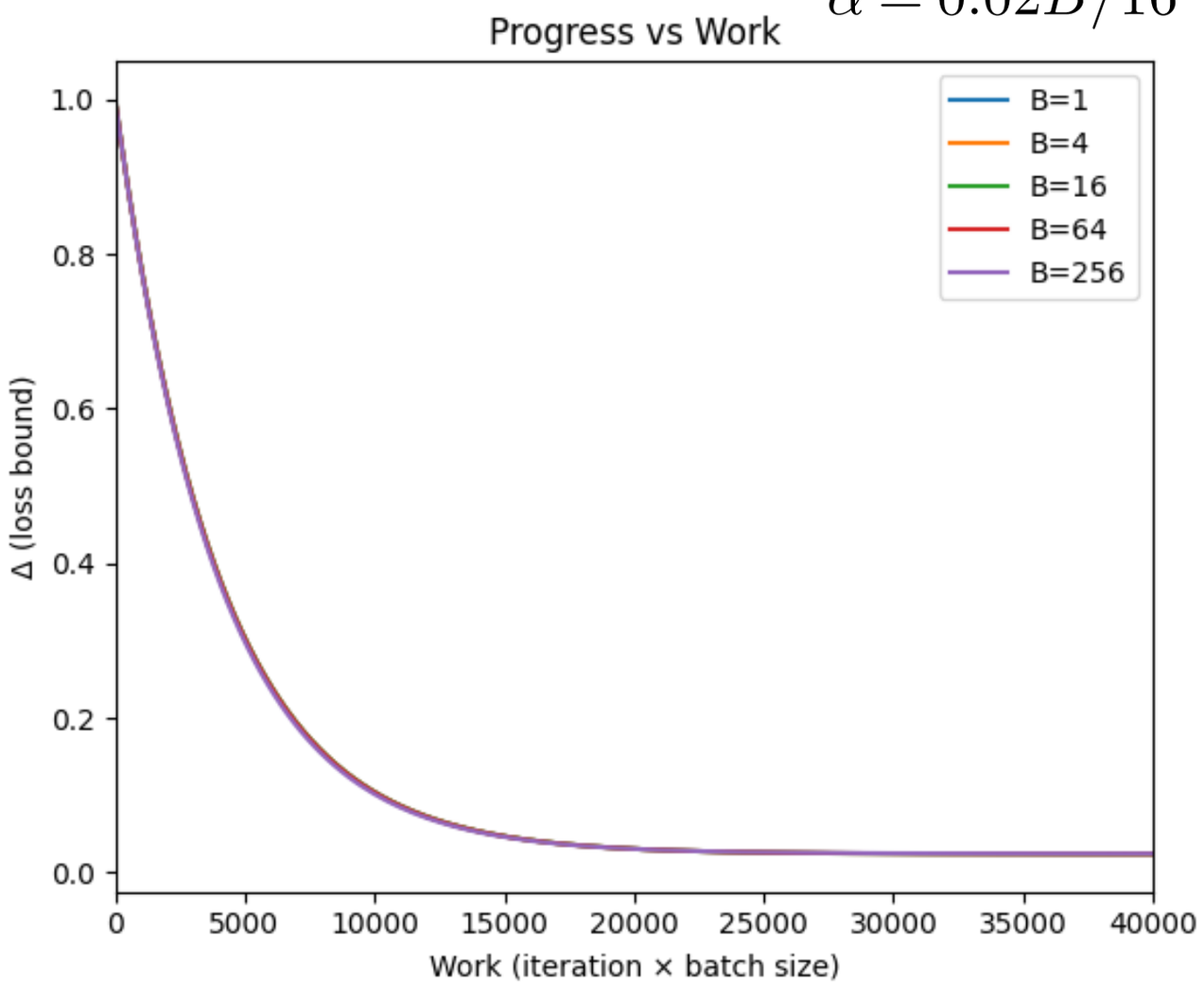
noise floor $O(\frac{\rho\alpha^2\sigma_0^2}{2cB})$

Learning Rate vs Batch Size

- Expected grad norm bound: $\frac{\Delta}{cT} + \frac{\rho\alpha^2\sigma^2}{2c}$, where $c = \alpha - \frac{\rho\alpha^2}{2}$, $\Delta\mathcal{L} = \mathcal{L}(\theta_0) - \mathcal{L}^*$
  - $\sigma^2 = \frac{\sigma}{B}$, noise floor $\frac{\rho\alpha^2\sigma_0^2}{2cB} \approx \frac{\rho\alpha\sigma_0^2}{2B}$
  - Same noise floor $\Rightarrow \alpha \propto B$

$\alpha = 0.02B/16$



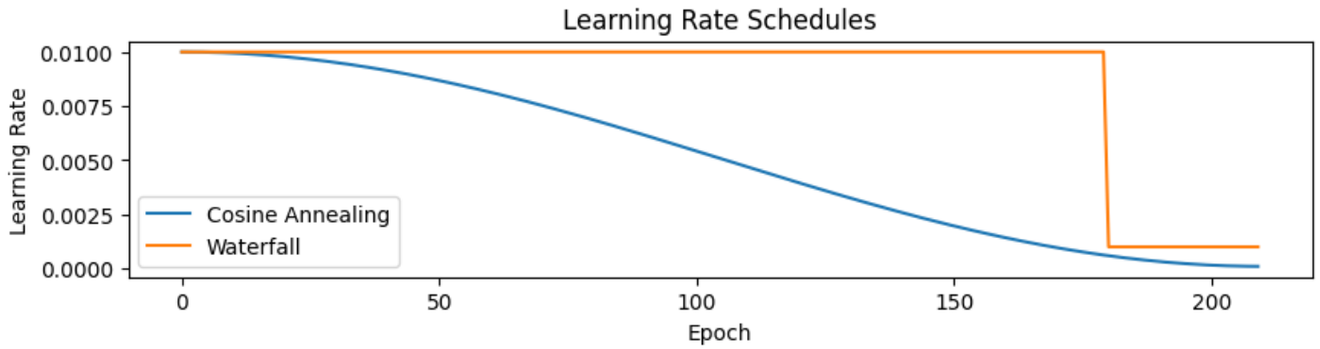


Notes: - works only for small learning rate -- eventually SGD wins

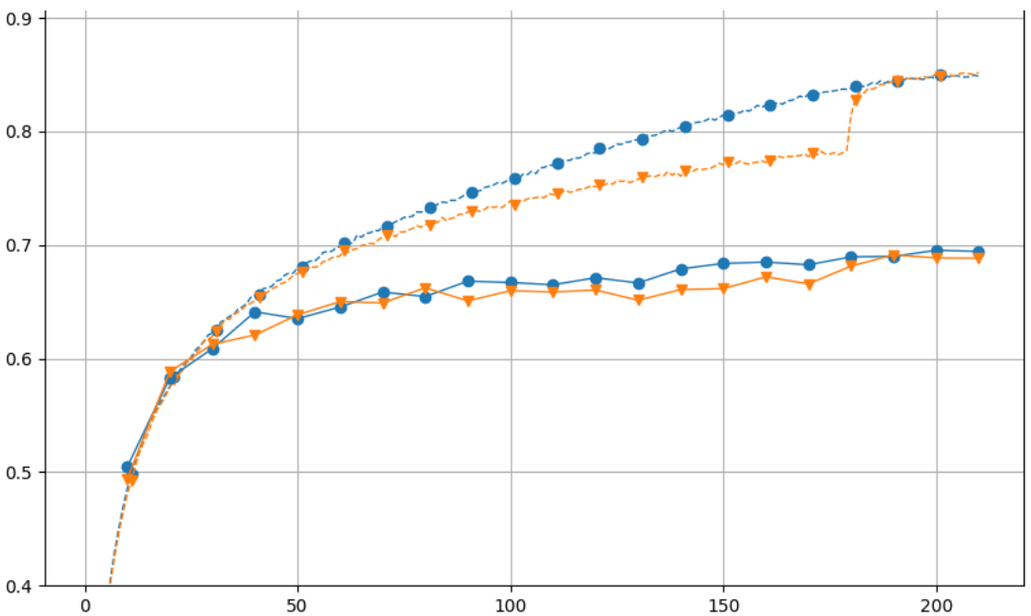
- a different rule is used for adaptive methods

Learning Rate Schedule

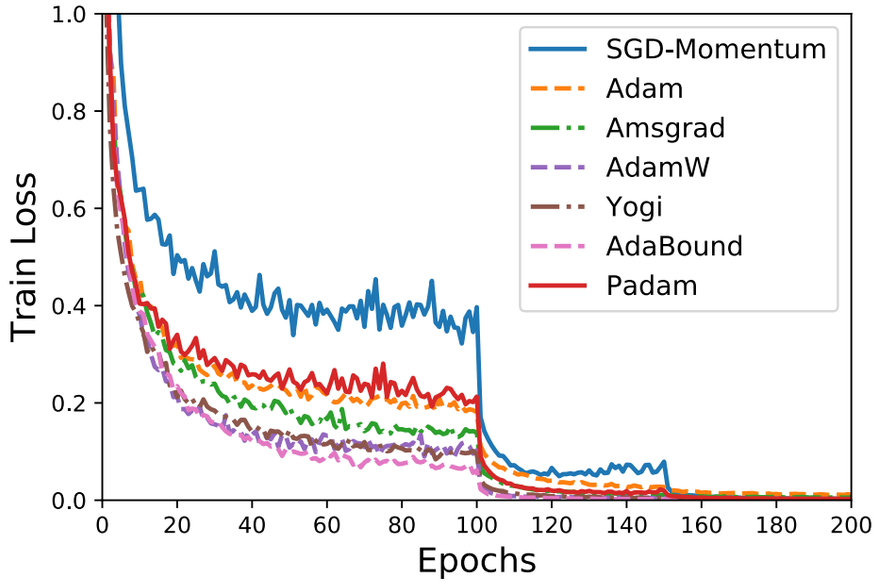
- **Step / Waterfall schedule:**
  - Decrease learning rate in steps, e.g: start with $\alpha = 0.1$ then decrease by factor of 10 at epochs 100 and 150
  - C.f. idea of fast convergence to a region: after step size decrease, 1/n rate replays
- **Cosine schedule:** - more convenient



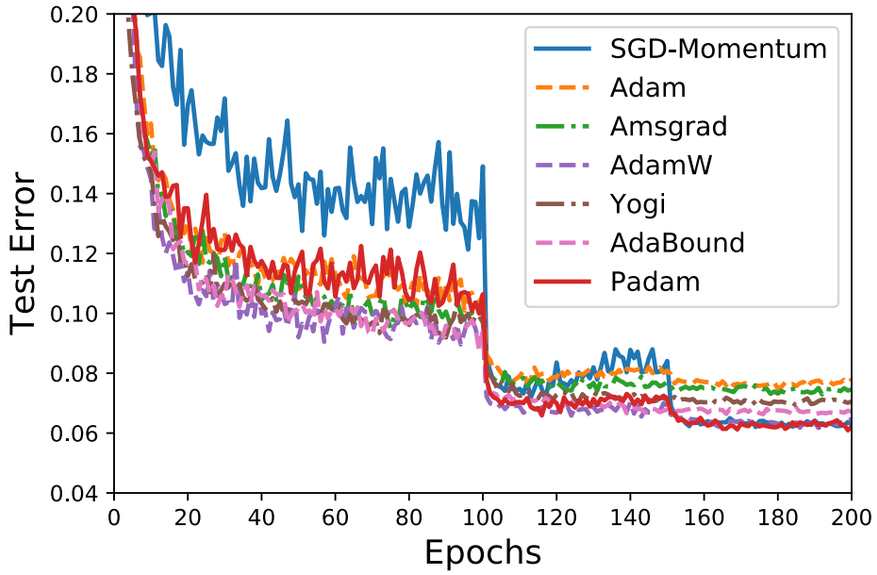
- **Often does not matter:**



Test Accuracy (Quantized training on ImageNet)



(a) Train Loss for VGGNet



(d) Test Error for VGGNet

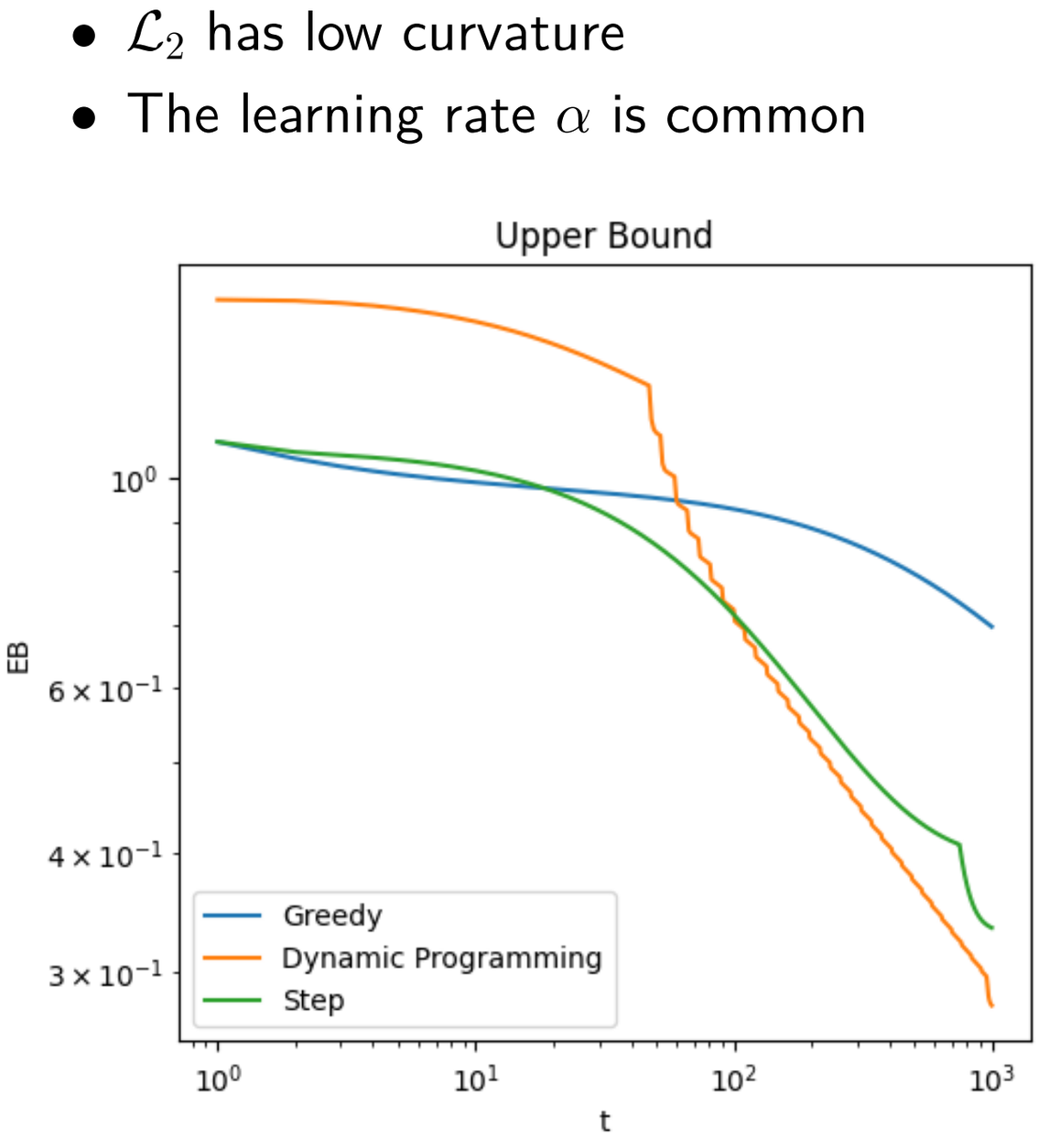
[Chen et al. "Closing the Generalization Gap of Adaptive Gradient Methods in Training Deep Neural Networks"]

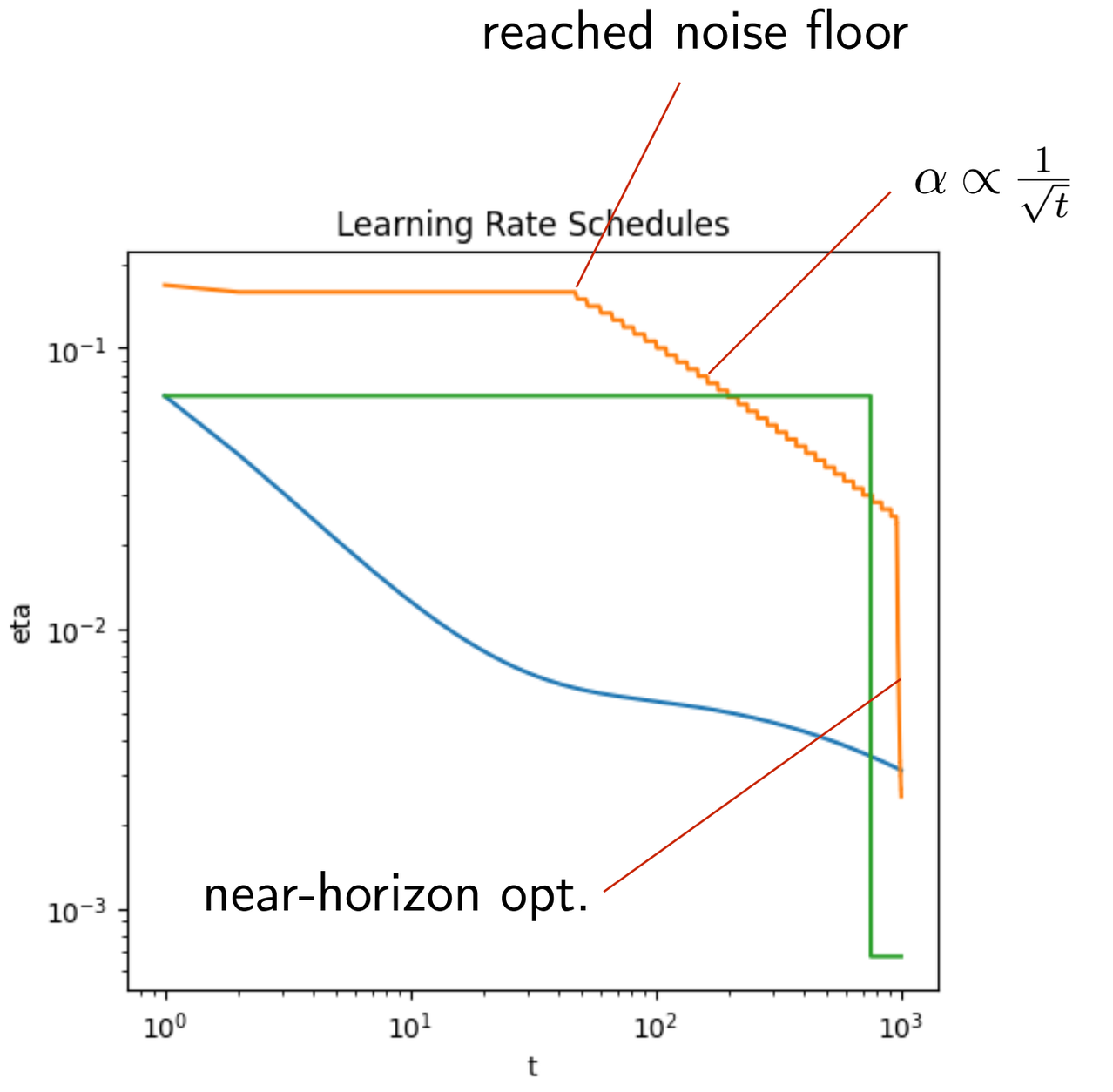
Learning Rate Schedule

- **Refined Toy Model**: expected improvement with two orthogonal components:

  $\mathcal{L}(\theta) = \mathcal{L}_1(\theta_1) + \mathcal{L}_2(\theta_2)$

  - $\mathcal{L}_1$ has high curvature
  - $\mathcal{L}_2$ has low curvature
  - The learning rate α is common





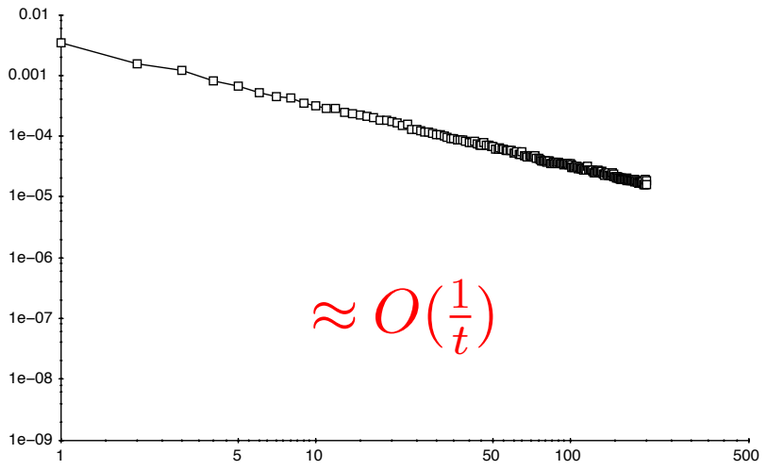
The optimal schedule rumps lr up, so that slow curvature component can be optimized while the high curvature one does not diverge. This hides the progress until we decrease lr

Dataset Samplers (with w/o replacement)

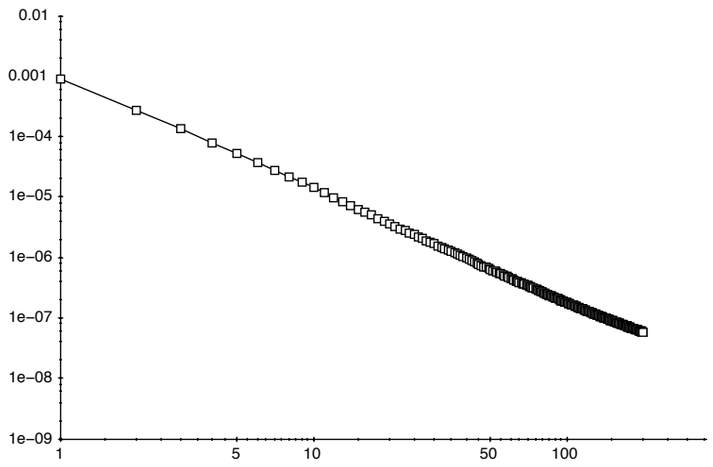
- How should we draw data points for SGD:
  - every time select randomly with replacement
  - shuffle the data once (c.f. incremental GD)
  - shuffle at each epoch but draw without replacement
- Empirical evidence:

Bottou (2009): "Curiously Fast Convergence of some Stochastic Gradient Descent Algorithms"

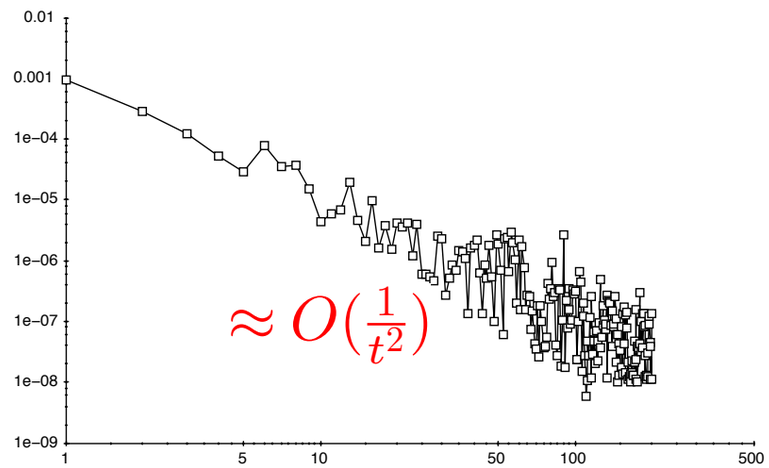
logistic regression $d = 47{,}152$, $n = 781{,}256$



Random selection: slope=−1.0003



Cycling the same random shuffle: slope=−1.8393



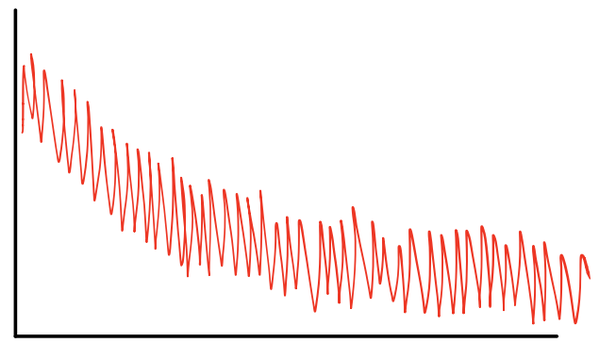
Random shuffle at each epoch: slope=−2.0103

- A simple consideration:

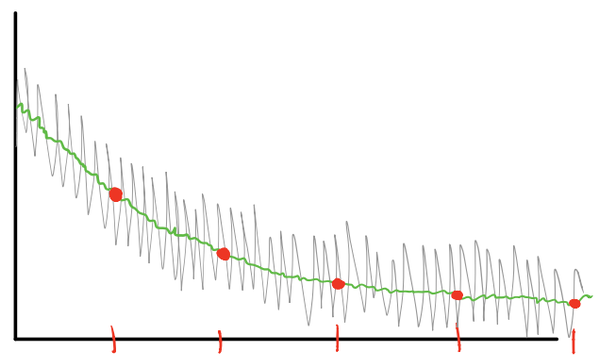
  Drawing n times with replacement from the dataset of size n some points may not be selected – efficiently using a subset of data per epoch.

Variance Reduction: Loss

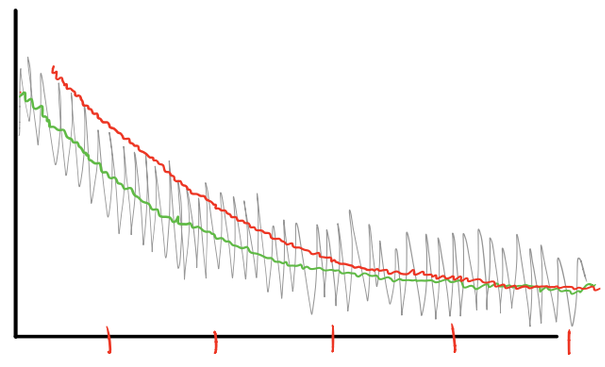
- Batch Estimate
  - Batch mean: $\tilde{L} = \frac{1}{M} \sum_{i \in I} l_i$
  - Unbiased, but high variance



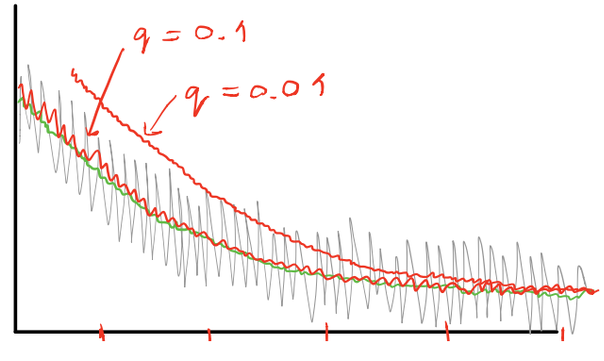
- Training data mean
  - $L = \frac{1}{n} \sum_{i=1}^{n} l_i$
  - Unbiased, zero variance, but may be too costly



- Average using all last known loss values
  - $\hat{L} := \frac{1}{n} \left(\sum_{i \in I} l_i^{\text{new}} + \sum_{i \notin I} l_i^{\text{old}} \right)$
  - Low variance, hysteresis 1 epoch
  - Need to remember losses for full dataset

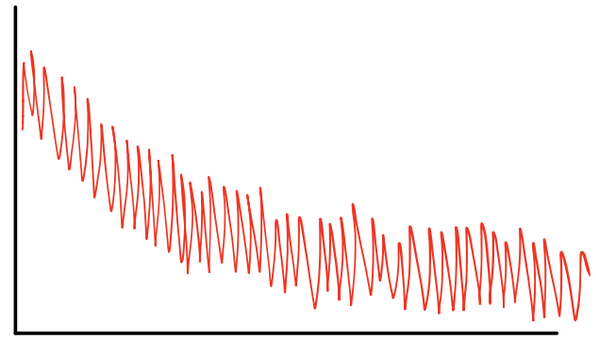


- Running Averaging
  - $\hat{L}^{t+1} := (1-q)\hat{L}^t + q\tilde{L}$
  - Variance-hysteresis tradeoff controlled by q
  - Need to remember only the running average loss

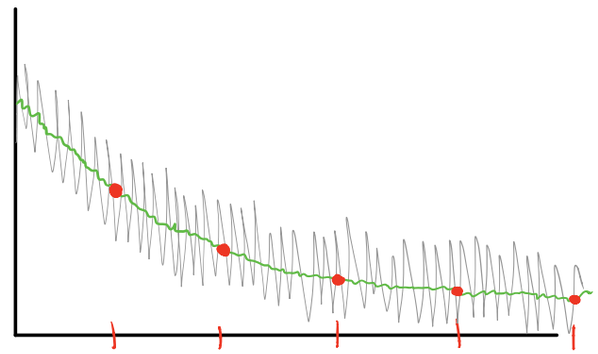


Variance Reduction: Gradient

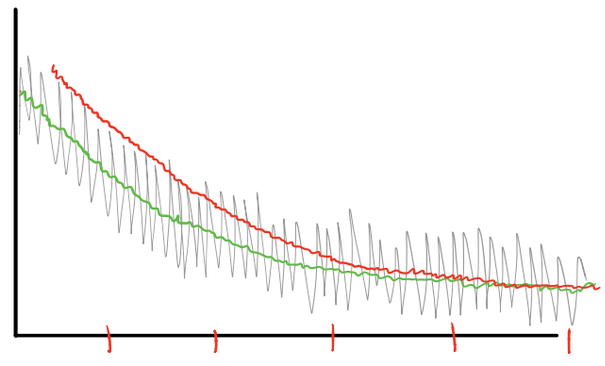
- **SGD**
  - Batch mean: $\tilde{g} = \frac{1}{M} \sum_{i \in I} \nabla l_i$
  - Need a small step size



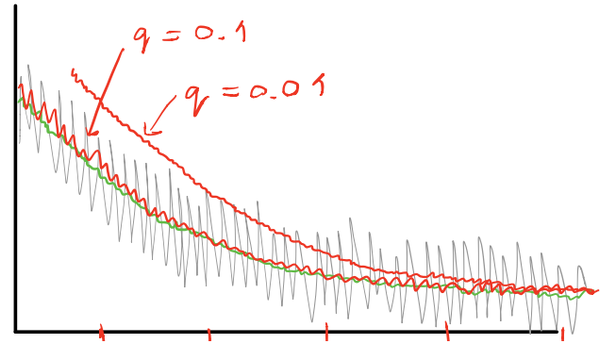
- **GD**
  - Full gradient: $g = \frac{1}{n} \sum_{i=1}^{n} \nabla l_i$
  - Too costly



- Stochastic Average Gradient (**SAG**)
  - $\tilde{g} := \frac{1}{n} \left(\sum_{i \in I} (\nabla l_i)^{\text{new}} + \sum_{i \notin I} (\nabla l_i)^{\text{old}} \right)$
  - Improved convergence rates (convex analysis)
  - Need to remember gradients



- **SGD** with filtered gradient (SGD with **momentum)**
  - $g := (1-q)g + q\tilde{g}$
  - Variance-hysteresis tradeoff controlled by q
  - Remember only the running average gradient

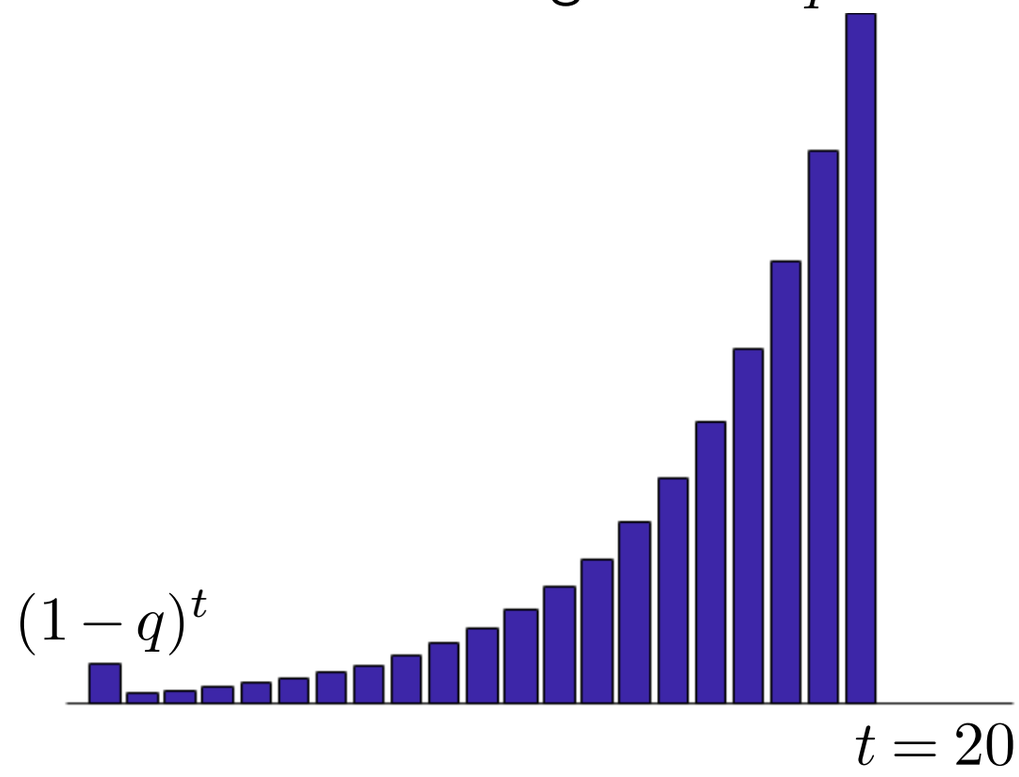


First Order Filter

- General setup:
  - X_k, $k = 1, \ldots, t$ – independent random variables
  - $q_t \in (0, 1]$
  - First order filter: $\mu_t = (1 - q_t)\mu_{t-1} + q_t X_t$
- Exponentially Weighted Average (**EWA**):
  - Constant $q_t = q$
  - $\mu_1 = (1 - q)\mu_0 + qX_1$
  - $\mu_2 = (1 - q)^2\mu_0 + (1 - q)qX_1 + qX_2$
  - $\ldots$
  - $\mu_t = (1 - q)^t\mu_0 + \sum_{1 \le k \le t} (1 - q)^{t-k} qX_k$

$$= w_0\mu_0 + \sum_{1 \le k \le t} w_k X_k$$

EWA weights $q = 0.2$



Running mean weights

- **Running mean**:
  - $q_t = \frac{1}{t}$
  - $\mu_1 = 0\mu_0 + X_1$
  - $\mu_t = \frac{t-1}{t}\mu_{t-1} + \frac{1}{t}X_t$
  - $\mu_{t+1} = \frac{t}{t+1}\mu_t + \frac{1}{t+1}X_{t+1} = \frac{t-1}{t+1}\mu_{t-1} + \frac{1}{t+1}(X_t + X_{t+1})$

- Averaging over past gradients reduces variance, but introduces a hysteresis bias

(*) EWA: How Much Variance Reduction?

- General setup
  - X_t – independent random variables
  - $q_t \in (0, 1]$
  - Running mean: $\mu_t = (1 - q_t)\mu_{t-1} + q_t X_t$ is a r.v.
- Expectation:
  - $\mathbb{E}[\mu_t] = (1 - q_t)\mathbb{E}[\mu_{t-1}] + q_t \mathbb{E}[X_t]$ – running average of expectations
  - $\mathbb{E}[\mu_t] = w_0 \mathbb{E}[\mu_0] + \sum_{k=1}^{t} w_k \mathbb{E}[X_k]$
  - In context of SGD with learning rate $\varepsilon \to 0$, all $E[X_k]$ are the same and μ_t is an unbiased estimate
- Variance:
  - $\mathbb{V}[\mu_t] = (1 - q_t)^2 \mathbb{V}[\mu_{t-1}] + q_t^2 \mathbb{V}[X_t]$
  - $\mathbb{V}[\mu_t] = w_0^2 \mathbb{V}_0 + \sum_{k=1}^{t} w_k^2 \mathbb{V}[X_k]$
  - Variance reduction of running mean: $\sum_{k=0}^{t} w_k^2 = \sum_{k=1}^{t} \frac{1}{t^2} = \frac{1}{t}$
  - Variance reduction of EWA: $\sum_{k=0}^{t} w_k^2 = \frac{q^2}{1-(1-q)^2}$ – in the limit of large t
- ($\star$) Equivalent window size of EWA: $n = \frac{2}{q} - 1$. E.g. $q = 0.1 \leftrightarrow n = 19$

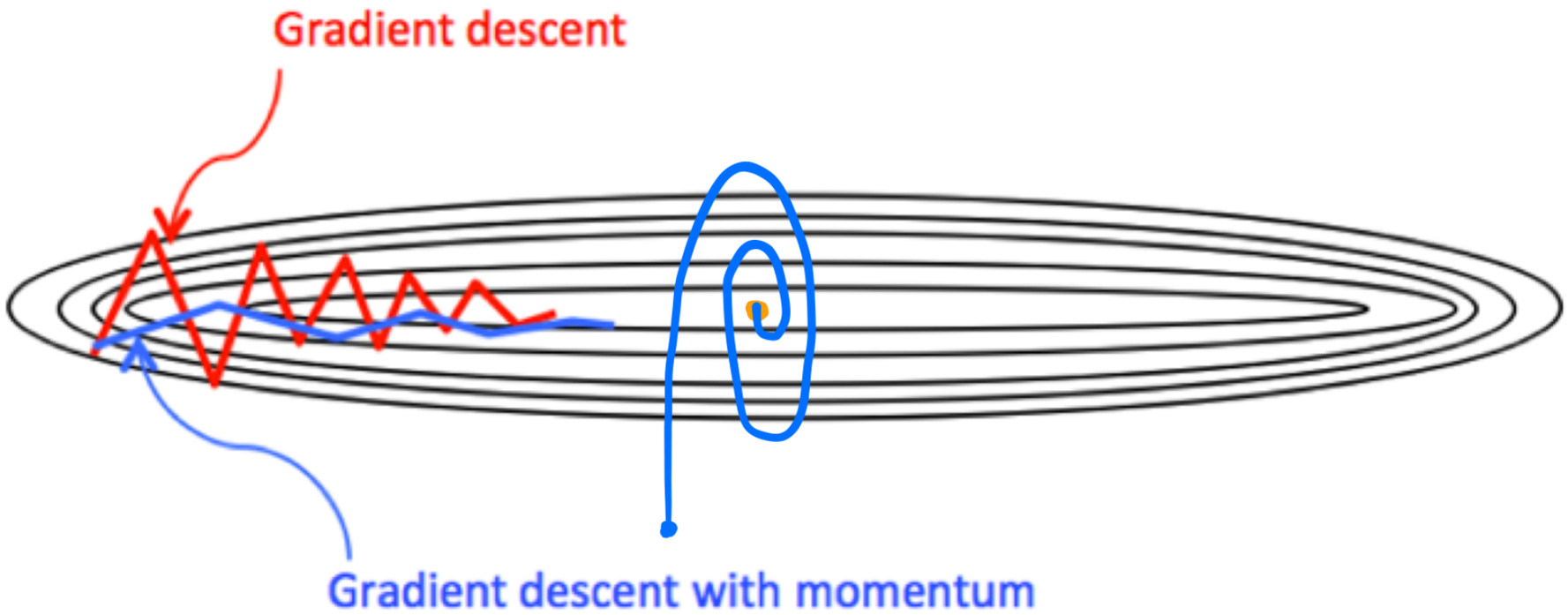
- ✦ Useful e.g. when adjusting for batch size changes

Hysteresis Bias

- With variance sufficiently low $\rightarrow$ GD with momentum. Consider $\tilde{g}$ is noise-free

  Equivalent form of SGD with EWA gradient ($\star$):

  - Velocity: $v_t := \mu v_{t-1} + \tilde{g}$
  - Step: $\theta_t = \theta_{t-1} - \varepsilon v_t$



- The **"heavy ball"** method
  - Friction ($\mu < 1$) and slope forces build up velocity
  - Cancels "noise" in the incorrect prediction of the function change, helpful to overcome plateaus
  - The inertia may lead to oscillatory behavior (not good)

(*) Nesterov Momentum -- mitigates inertia

- Common Momentum
  - Velocity: $v_{t+1} = \mu v_t + \tilde{g}(x_t)$
  - Step: $x_{t+1} = x_t - \varepsilon v_{t+1}$

The step consists of momentum and current gradient

The momentum part of the step is **known in advance**

Can make it before computing the gradient:

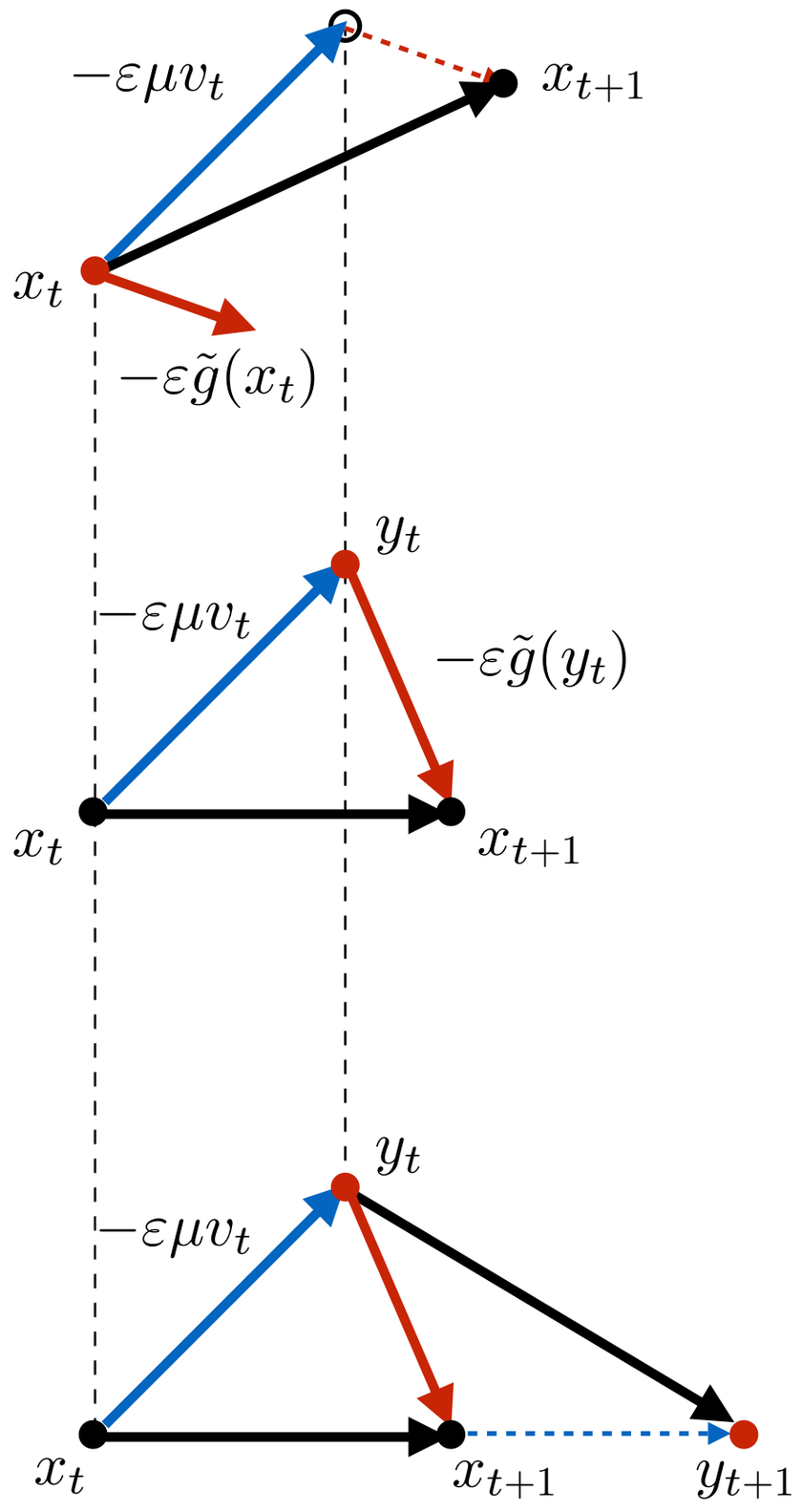
- Nesterov Momentum
  - Leading sequence: $y_t = x_t - \varepsilon\mu v_t$
  - Velocity: $v_{t+1} = \mu v_t + \tilde{g}(y_t)$
  - Step: $x_{t+1} = y_t - \varepsilon\tilde{g}(y_t)$

Takes advantage of the known part of the step

Less overshooting

- ($\star$) Can express as steps on the leading sequence alone:
  - Velocity: $v_{t+1} = \mu v_t + \tilde{g}(y_t)$
  - Step: $y_{t+1} = y_t - \varepsilon\big(\tilde{g}(y_t) + \mu v_{t+1}\big)$

  The two sequences eventually converge



(*) Implicit Regularization

Implicit Regularization

- Logistic and multinomial regression, SVM, boosting:

$$\operatorname{argmin}_w \underbrace{\mathcal{L}(w) + \lambda \|w\|_p^p}_{\text{regularized loss}} \xrightarrow{\lambda \to 0} \frac{w}{\|w\|_p} \to \max \text{ margin w.r.t. } \|\cdot\|_p \quad [1]$$

Common GD for unregularized $\min_w \mathcal{L}(w)$:

- $w_{t+1} = w_t - \alpha \nabla \mathcal{L}(w_t)$
- Proximal steps with 2-norm

$$\xrightarrow{t \to \infty} \frac{w^t}{\|w^t\|} \to \max \text{ margin w.r.t. } \|\cdot\|_2 \quad [2]$$

- Linear model, loss with unique finite root (e.g. L_2):

$$\min_w \mathcal{L}(w) := \sum_{n=1}^N \ell\left(\langle w, x_n \rangle, y_n\right) \xrightarrow{t \to \infty} w^t \to \text{ nearest optimum to } w^0 \text{ in } \|\cdot\|_p \quad [3]$$

w_t – steepest p-norm GD

- P-norm SGD works as implicit p-norm regularization, converging closer to max p-norm margin / interpolation solutions
- There are generalizations to over-parameterized nonlinear models and other divergences

---

[1] Rosset et al. (2004) Margin Maximizing Loss Functions

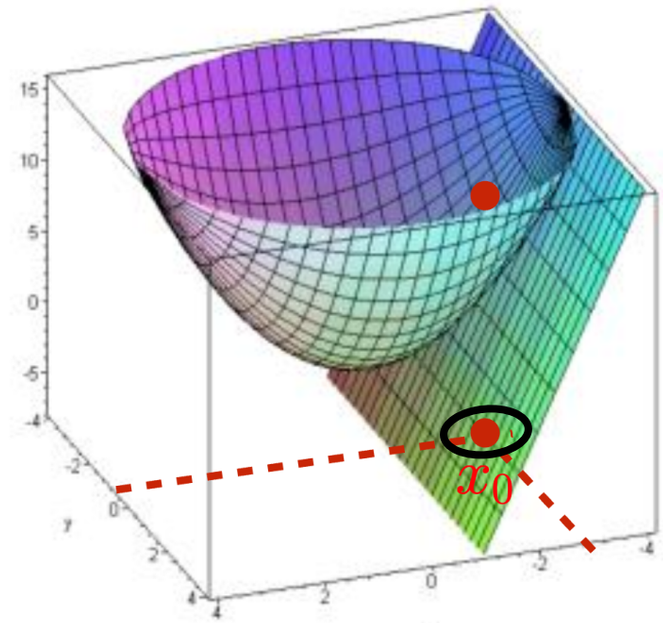
[2] Soudry et al. (2018) "The Implicit Bias of Gradient Descent on Separable Data"

[3] Gunasekar et al. (2018) "Characterizing Implicit Bias in Terms of Optimization Geometry"

Proximal Step Problem

- Let's revisit how do we find the step Δx for GD
  - Approximate: $f(x_t + \Delta x) \approx f(x_t) + J_t \Delta x$. This approximation is local.
- Make a step by solving **Proximal Problem**:

$$x_{t+1} = x_t + \underset{\Delta x}{\operatorname{argmin}} \left(f(x_t) + J\Delta x + \frac{1}{2\alpha}\|\Delta x\|_2^2 \right)$$



$$0 = \frac{\partial}{\partial \Delta x} = J_t + \frac{1}{\alpha}\Delta x^\top$$

$$\Delta x = -\alpha J_t^\top$$

$x_{t+1} = x_t - \alpha J_t^\top -$ **steepest (common) GD**

- **p-norm steepest GD,** $p > 1$ **:**

$$\min_{\Delta x} \left(f(x_t) + J_t \Delta x + \frac{1}{p\alpha}\|\Delta x\|_p^p \right)$$

$$\Delta x_i = -\alpha \operatorname{sign}(J_{ti}) |J_{ti}|^{\frac{1}{p-1}}$$

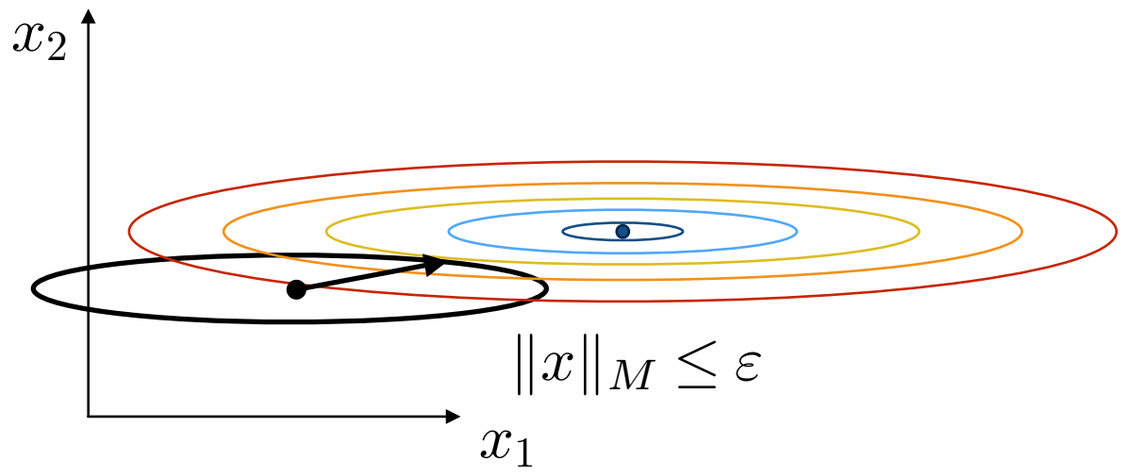
- **Machalanobis norm steepest GD**:

$$\min_{\Delta x} \left(f(x_t) + J_t \Delta x + \frac{1}{2\alpha}\|\Delta x\|_M^2 \right)$$

$\|\Delta x\|_M = (\Delta x^\mathsf{T} M \Delta x)^{\frac{1}{2}}$ – Mahalanobis distance

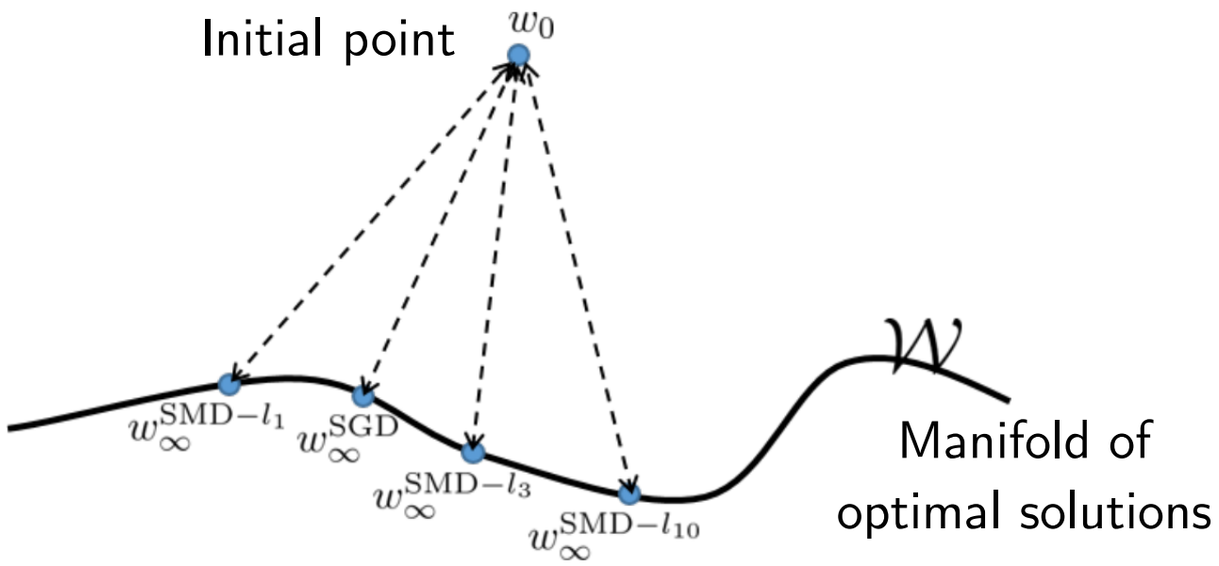
$$\Delta x = -\alpha M^{-1} J_t^\mathsf{T}$$

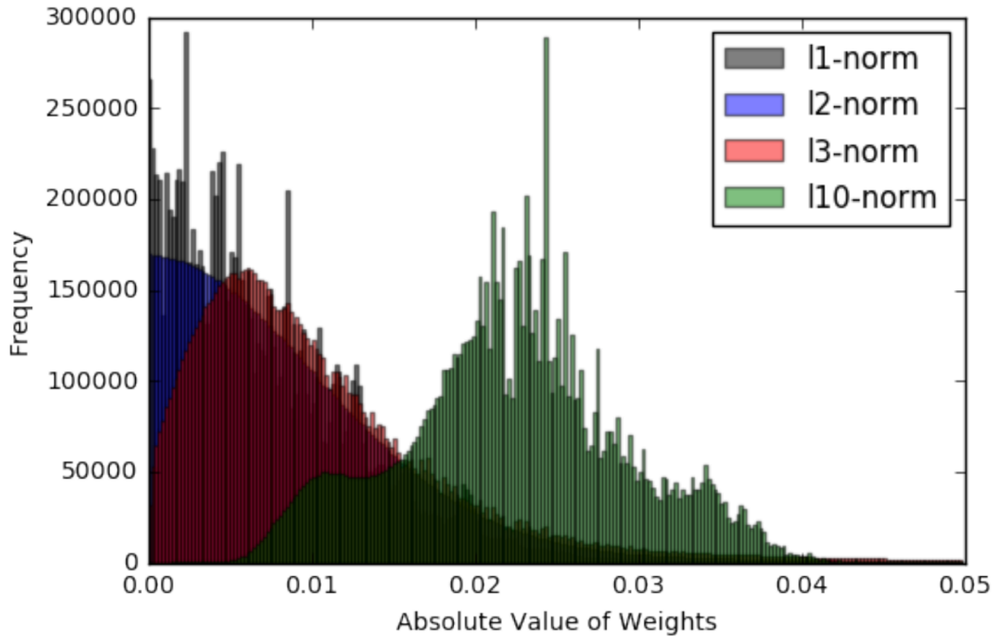
Equivalent to GD in coordinates $\tilde{x} = M^{1/2} x$



Implicit Regularization by SGD / SMD

- Consider step proximal problem: $\min_x \langle \nabla f(x_0), x - x_0 \rangle + \lambda \|x - x_0\|_p^p$
  - i.e., p-norm stochastic mirror descent
- Using different p leads to solutions with different properties

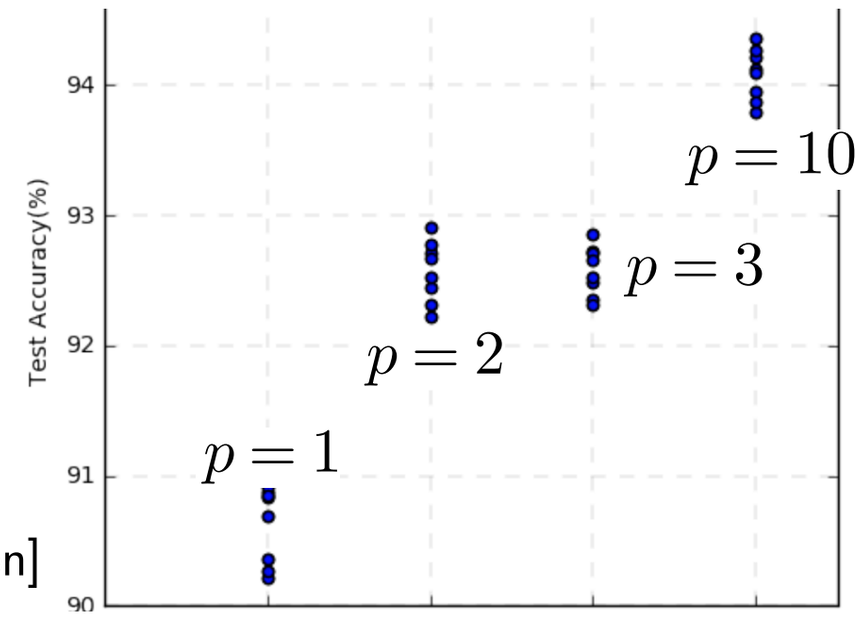




- Iterates tend to $\mathrm{argmin}_{w \in \mathcal{W}} \|w - w_0\|_p^p$, the closest point in the respective norm

	SMD 1-norm	SMD 2-norm (SGD)	SMD 3-norm	SMD 10-norm
1-norm BD	141	9.19×10^3	4.1×10^4	2.34×10^5
2-norm BD	3.15×10^3	562	1.24×10^3	6.89×10^3
3-norm BD	4.31×10^4	107	53.5	1.85×10^2
10-norm BD	6.83×10^{13}	972	7.91×10^{-5}	2.72×10^{-8}

- Different sparsity and generalization

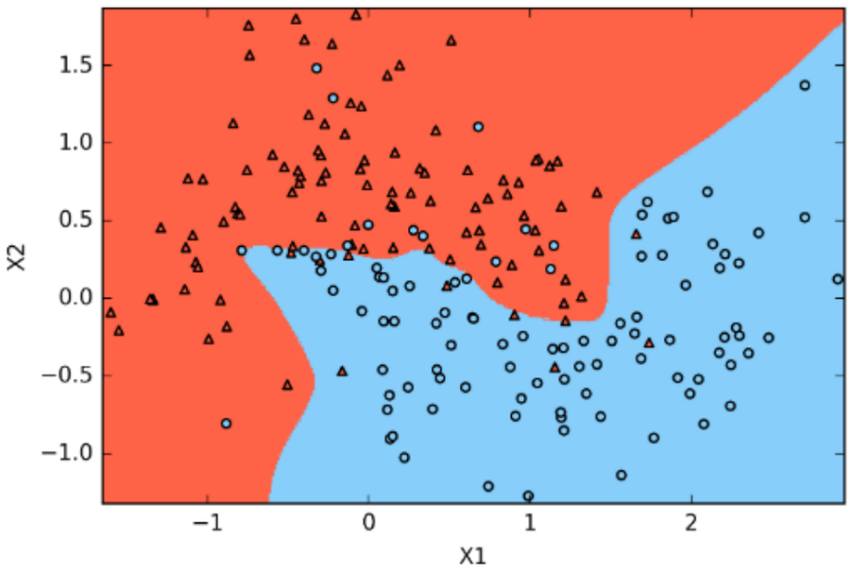


[Azizan et al. (2019) Stochastic Mirror Descent on Overparameterized Nonlinear Models: Convergence, Implicit Regularization, and Generalization]

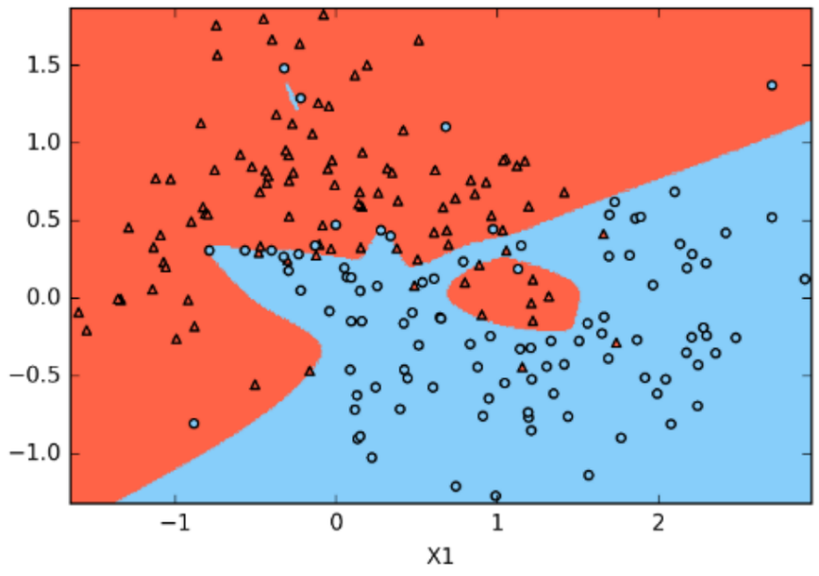
Smaller Batch Size -> More Regularization

- Typically choose batch size to fully utilize parallel throughput (in GPUs means ~10^4 independent arithmetic computations in parallel)
- Limited by memory
- Smaller batch -> noisier gradient -> implicit regularization

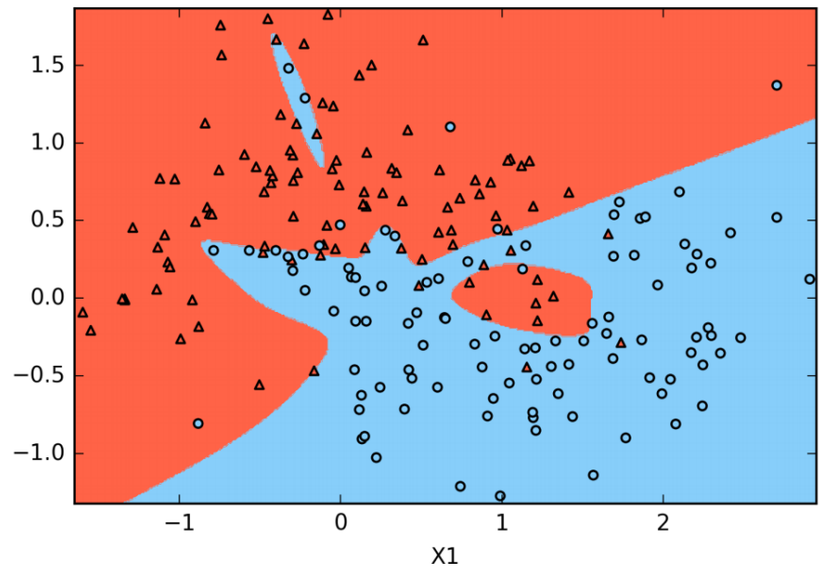
Synthetic data



Decision boundary of batch size 1

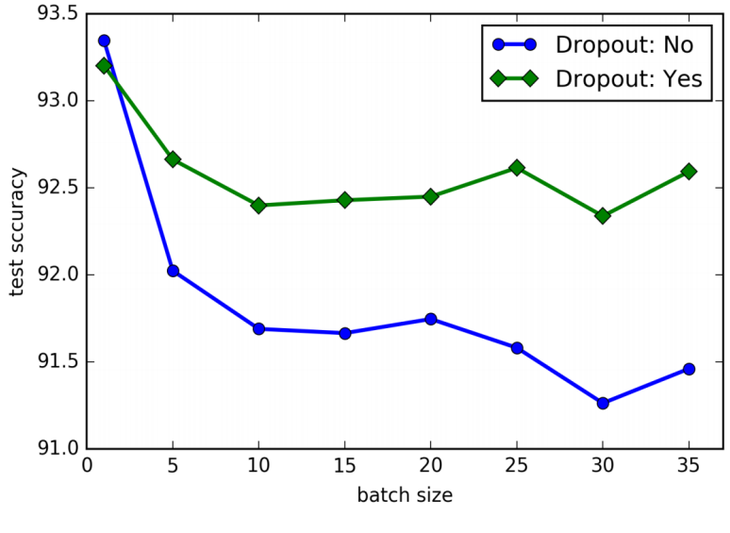


Decision boundary of batch size 5



Decision boundary of batch size 30

NLP data



Lei et al. (2018) "Implicit Regularization of Stochastic Gradient Descent in Natural Language Processing: Observations and Implications"