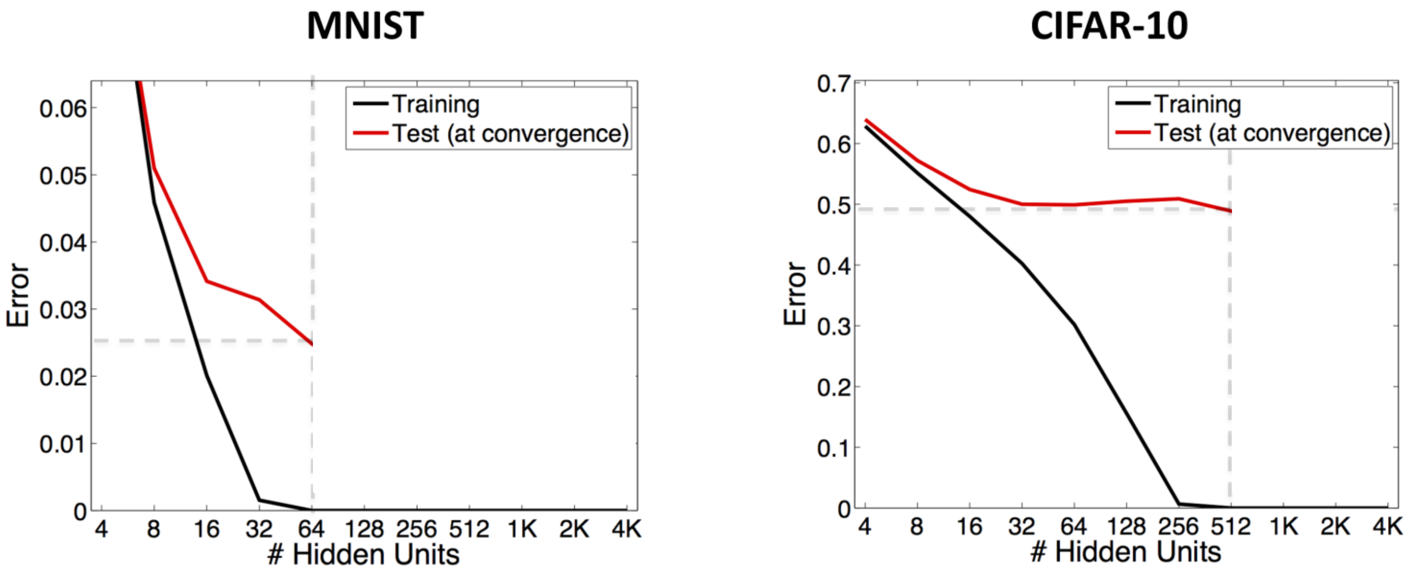
First-Order Optimization Algorithms for Machine Learning

Over-Parameterized Models

Summer 2020

"Hidden" Regularization in Neural Networks

- Fitting single-layer neural network with SGD and no regularization:



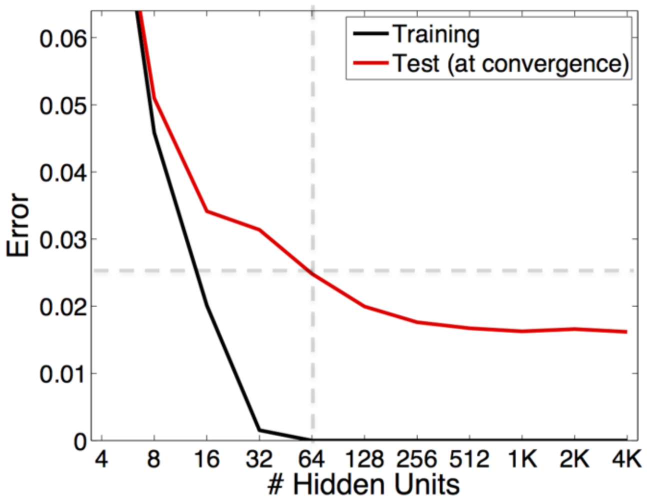
- Training goes to 0 with enough units: we're finding a global min.
  - Even though objective function is highly non-convex.
- What should happen to training and test error for larger #hidden?

https://www.neyshabur.net/papers/inductive_bias_poster.pdf

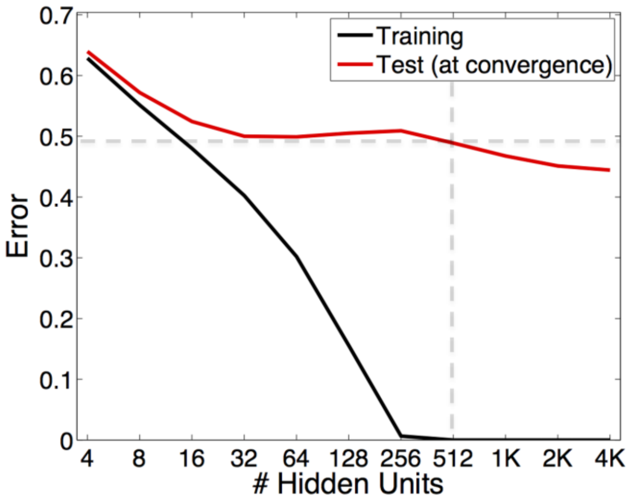
"Hidden" Regularization in Neural Networks

- Fitting single-layer neural network with SGD and no regularization:

MNIST



CIFAR-10

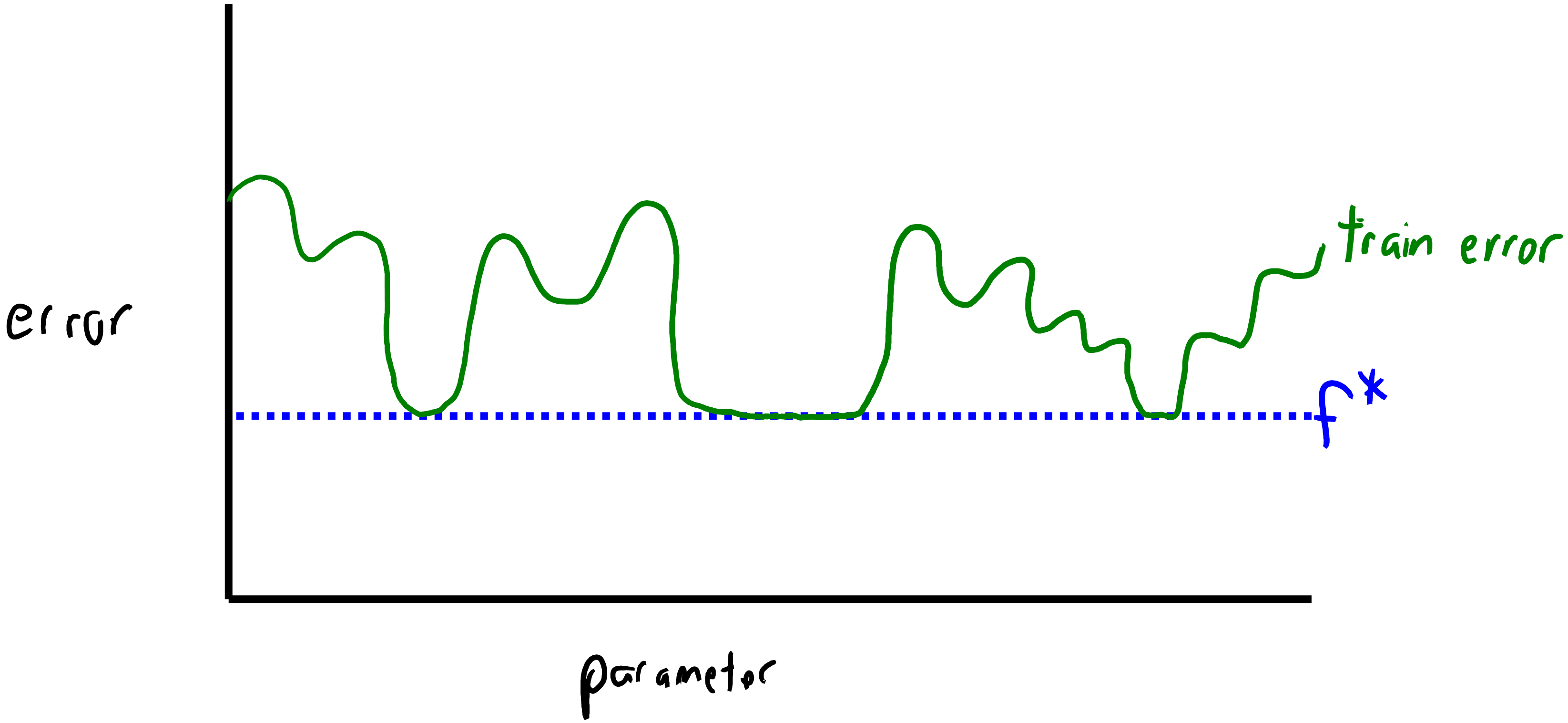


- Test error continues to go down!?! Where is fundamental trade-off??
- There exist global mins with large #hidden units with test error = 1.
  – But among the global minima, SGD is somehow converging to "good" ones.

https://www.neyshabur.net/papers/inductive_bias_poster.pdf

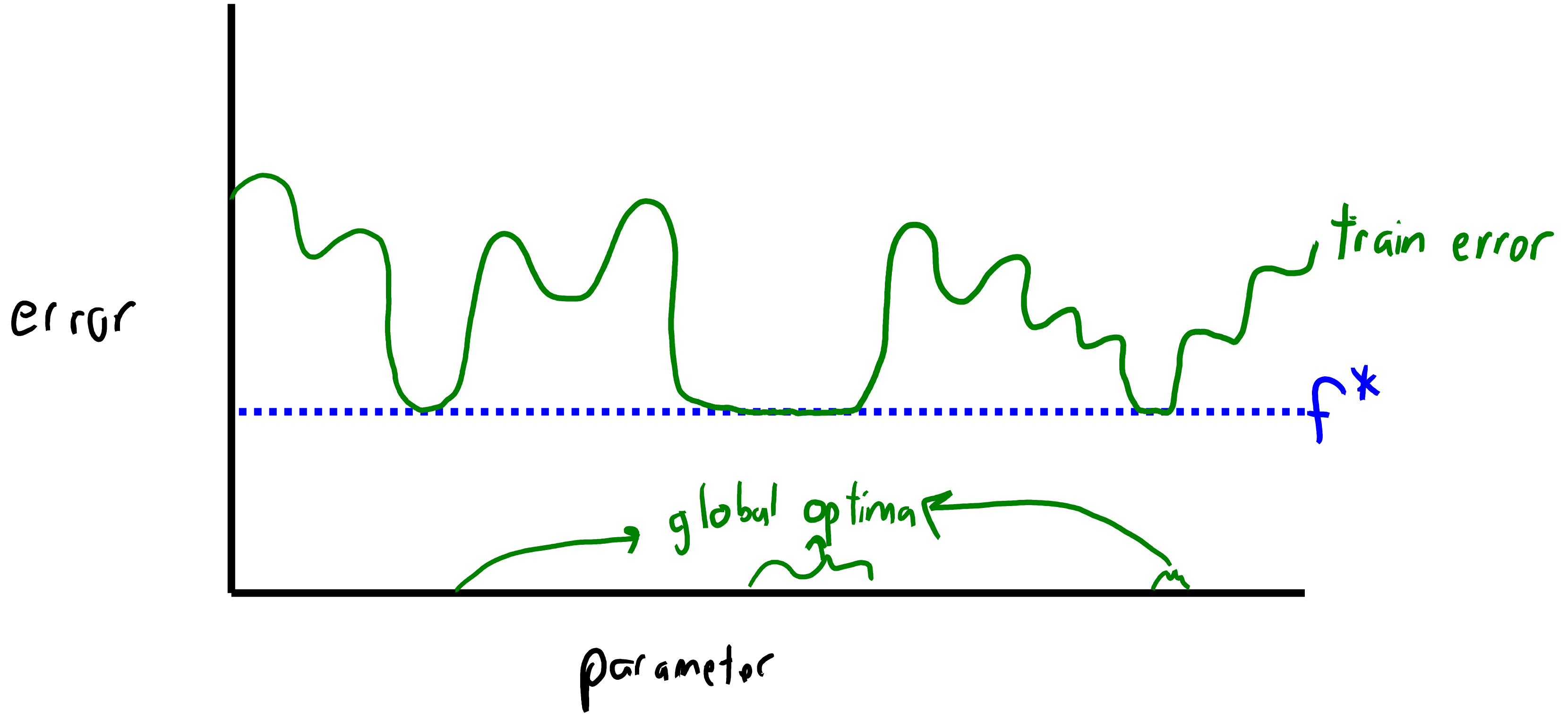
Multiple Global Minima?

- For standard objectives, there is a global min function value f*:



Multiple Global Minima?

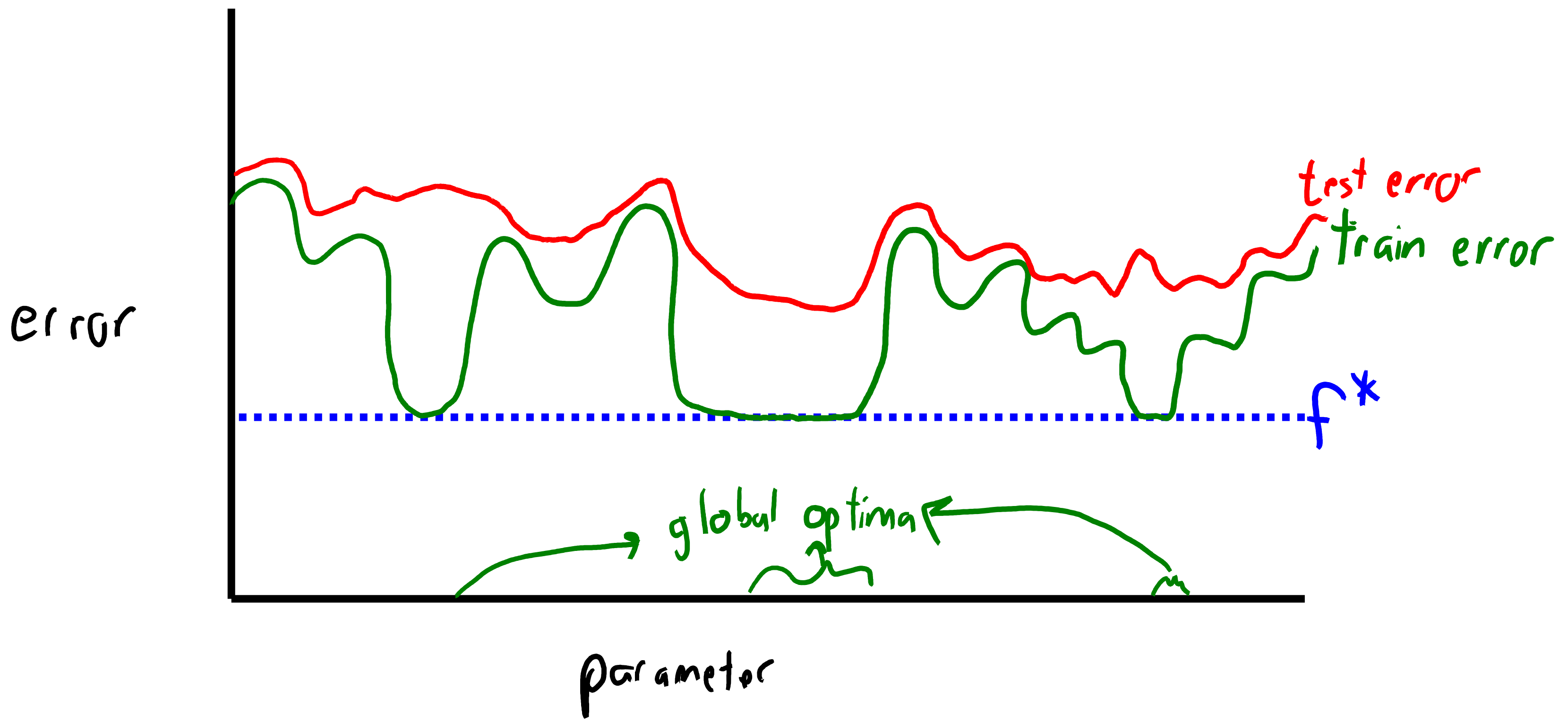
- For standard objectives, there is a global min function value f*:



- But this may be achieved by many different parameter values.

Multiple Global Minima?

- Now consider the test error:



  - These training error "global minima" may have very-different test errors.
  - Maybe some of these global minima might even be more "regularized" than others.

Implicit Regularization of SGD

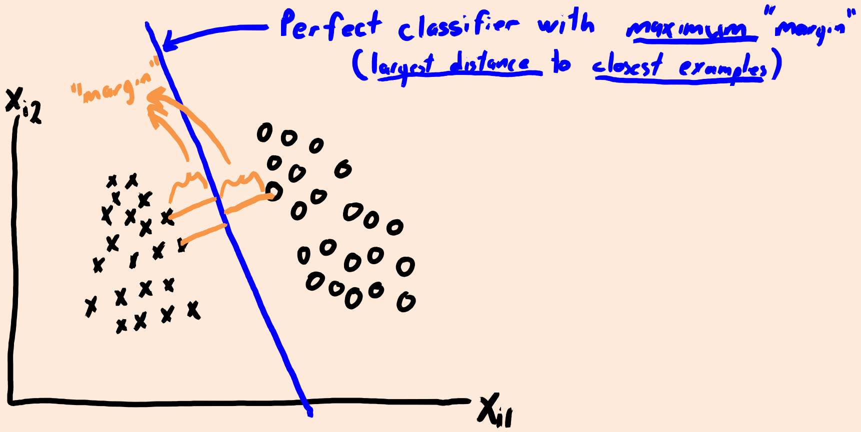
- There is growing evidence that using SGD regularizes parameters.
  - We call this the "implicit regularization" of the optimization algorithm.

- Experiments indicate SGD implicitly regularizes neural networks.
  - But we don't have a complete theory for how SGD is regularizing.
  - Beyond empirical evidence, we know this happens in simpler cases.

- Known example of implicit regularization in a simpler case:
  - Consider a least squares problem where there exists a 'w' where Xw=y.
    - Residuals are all zero, we fit the data exactly.
  - You run [stochastic] gradient descent starting from w=0.
  - Converges to solution Xw=y that has the minimum L2-norm.
    - So using SGD is equivalent to L2-regularization here, but regularization is "implicit".

Implicit Regularization of SGD

- Known example of implicit regularization in a simpler case:
  - Consider a logistic regression problem where data is linearly separable.
    - We can fit the data exactly.
  - You run gradient descent from any starting point.
  - Converges to max-margin solution of the problem.
    - So using gradient descent is equivalent to encouraging large margin on separable data.

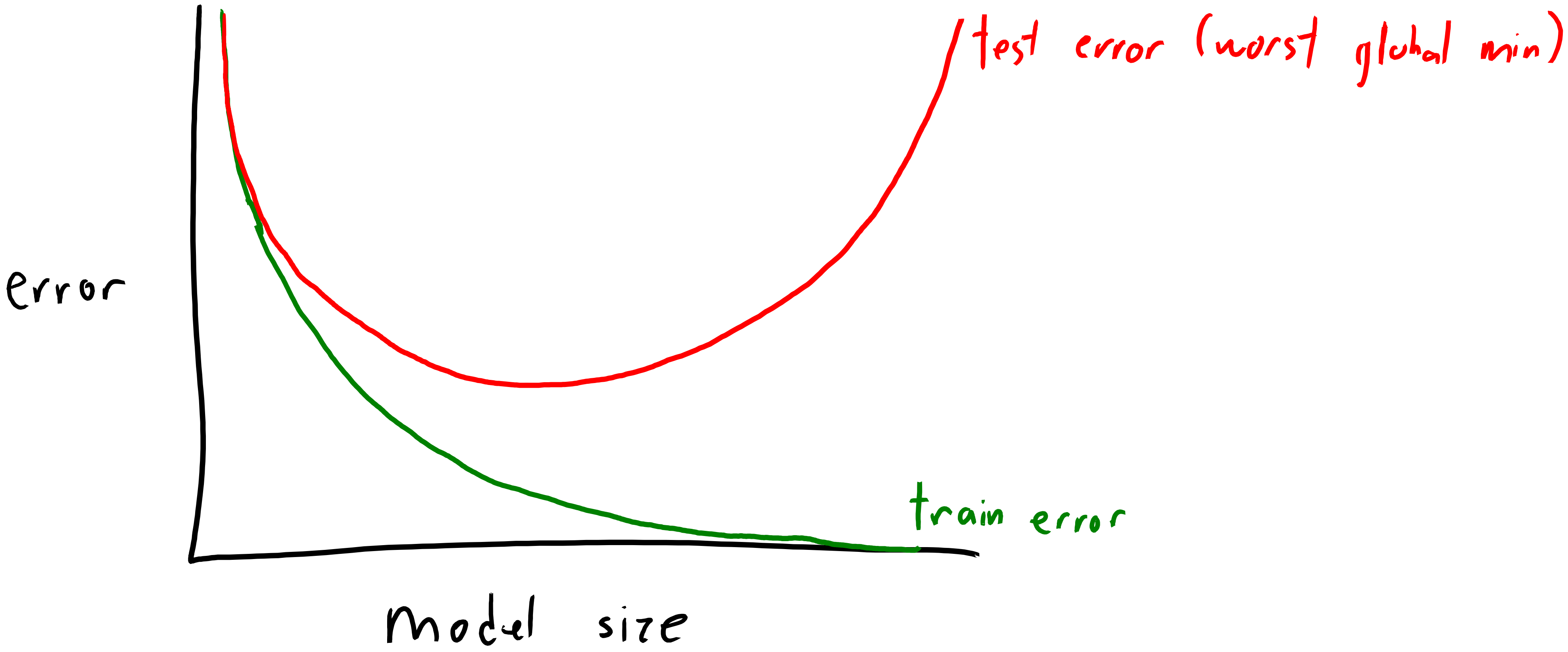


- Similar result known for boosting and matrix factorization.
  - Implicit regularization tends to also achieved with momentum,
    but may not be maintained if we use "adative" methods like AdaGrad/Adam.

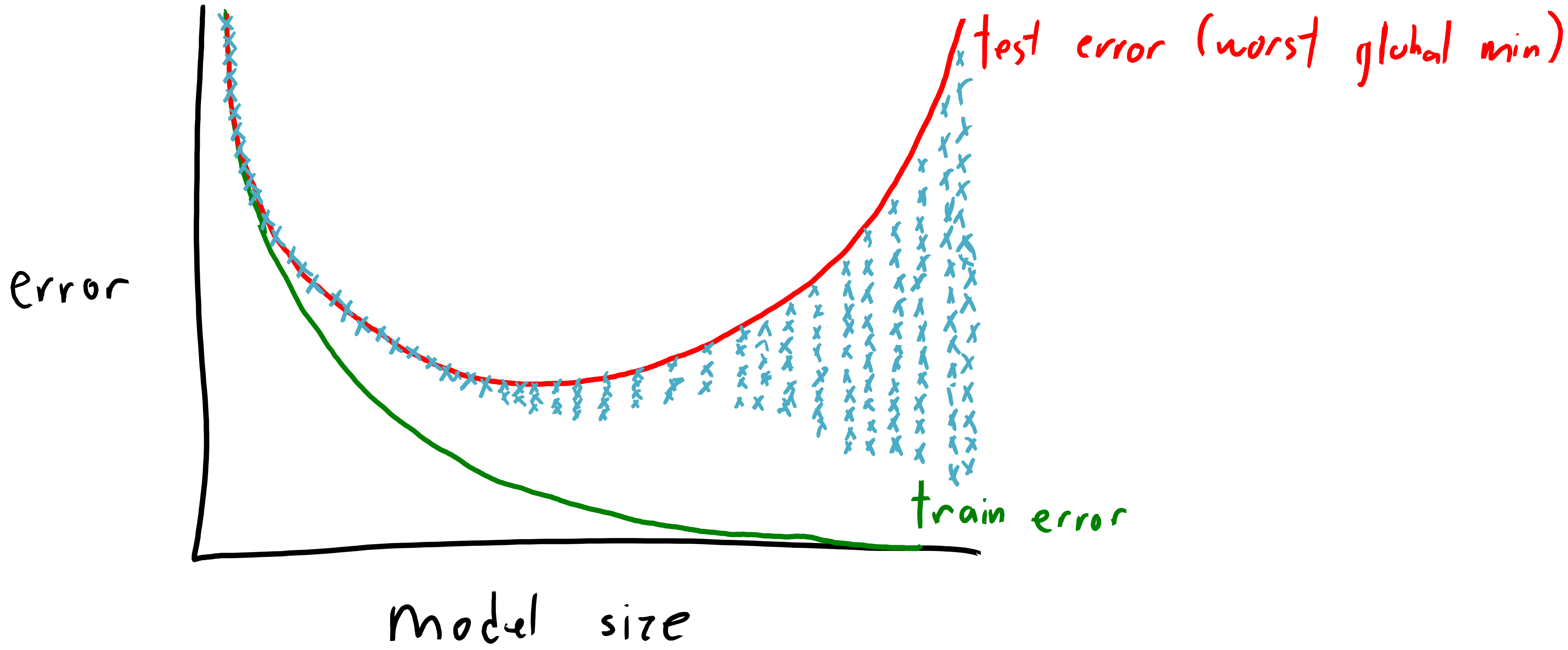
Double Descent Curves

- What is going on???

Worst vs. Best "Global Minimum"

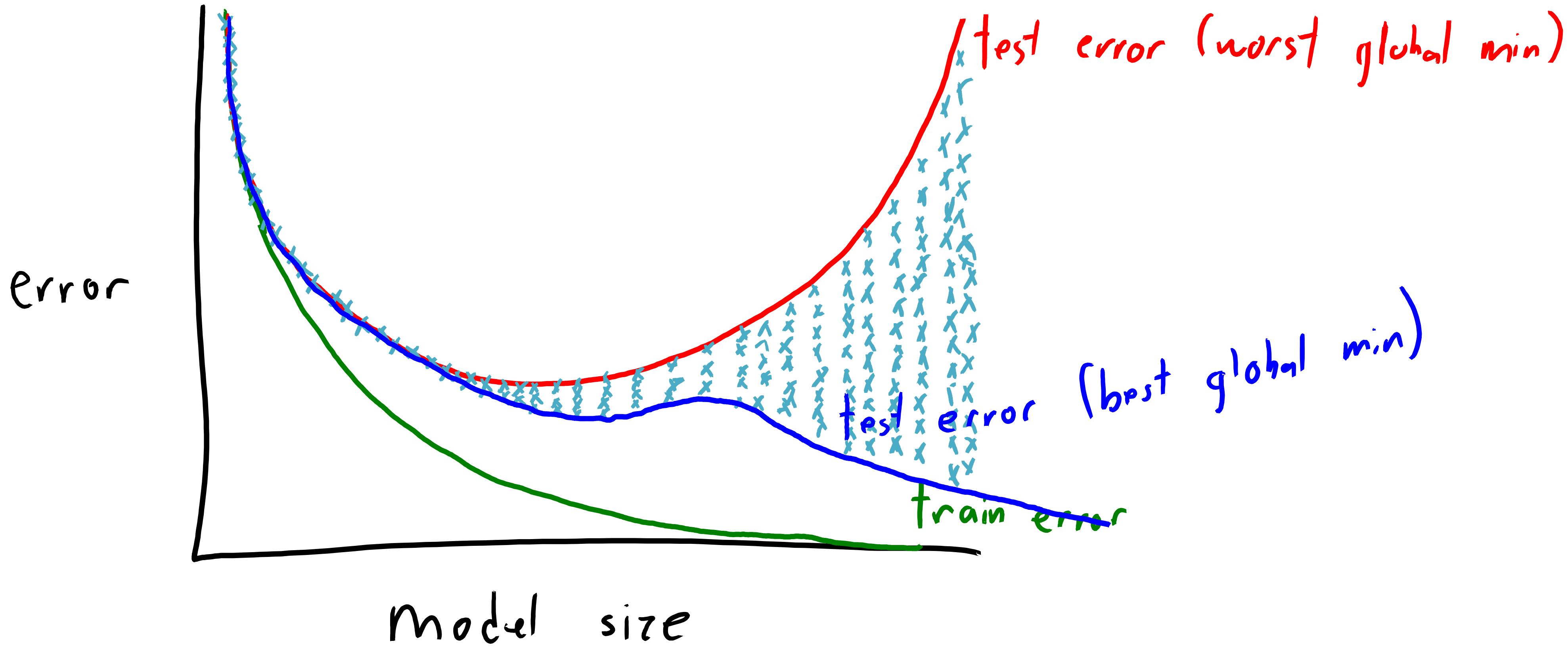


Worst vs. Best "Global Minimum"



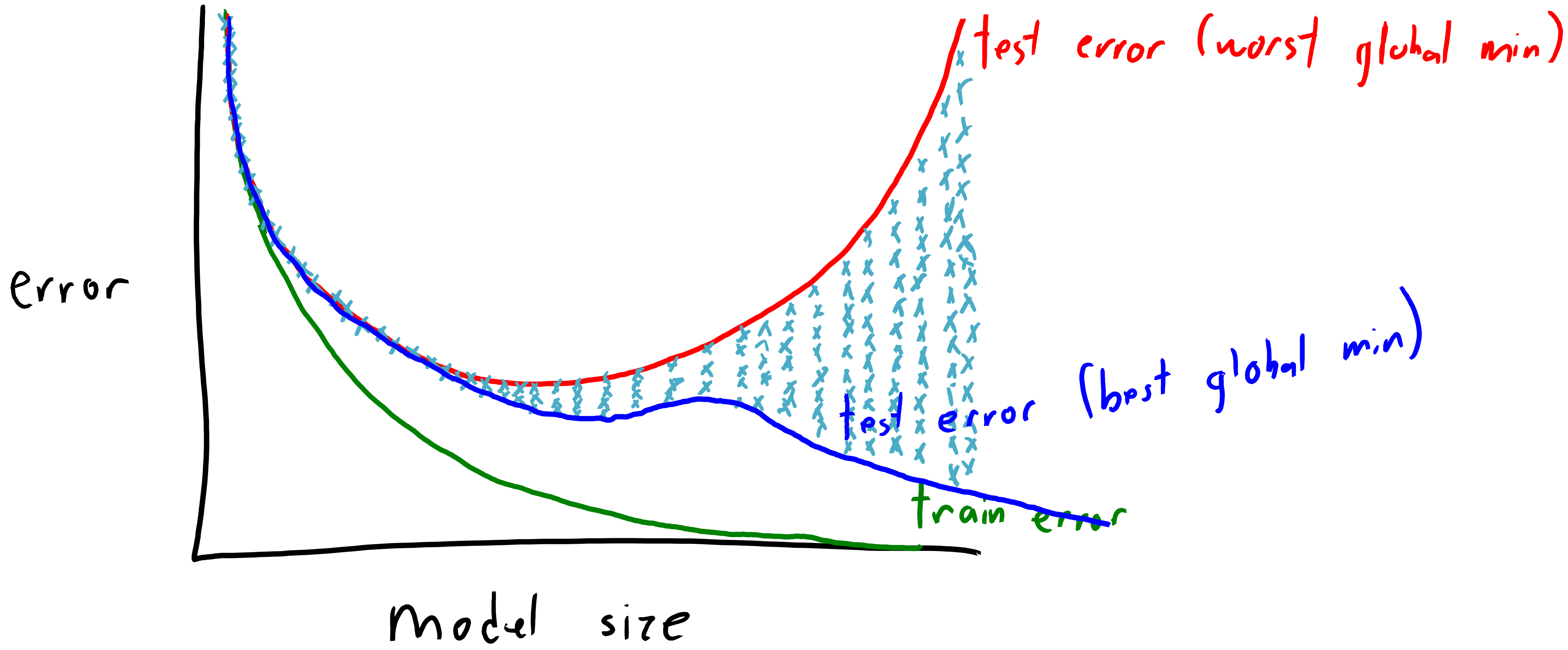
- Learning theory results analyze global min with worst test error.
  - Actual test error for different global minima be better than worst case bound.
  - Theory is correct, but maybe "worst overfitting possible" is too pessimistic?

Worst vs. Best "Global Minimum"



- Consider instead the global min with best test error.
  - With small models, "minimize training error" leads to unique (or similar) global mins.
  - With larger models, there is a lot of flexibility in the space of global mins (gap between best/worst).
- Gap between "worst" and "best" global min can grow with model complexity.

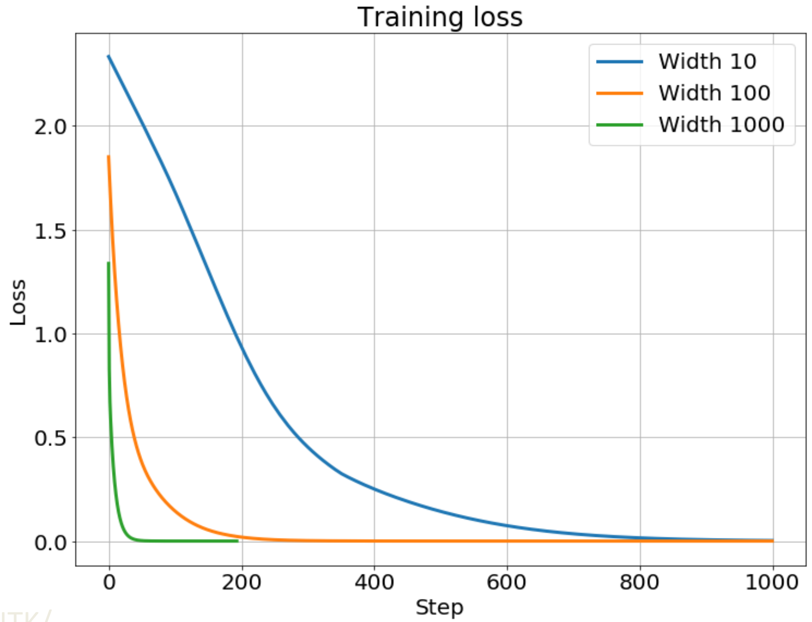
Worst vs. Best “Global Minimum”

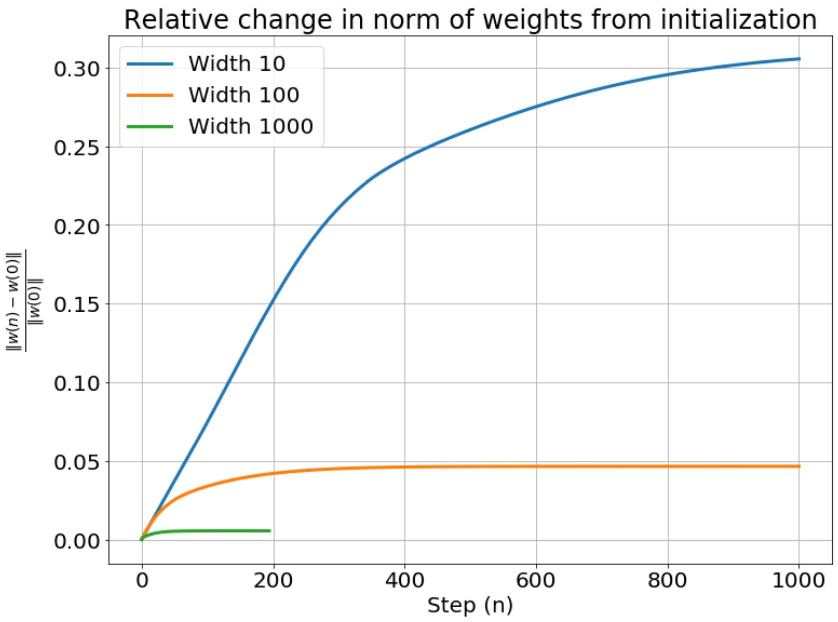


- Can get “double descent” curve in practice if parameters roughly track “best” global min shape.
  - One way to do this: increase regularization strength λ as you increase model size.
- Maybe “neural network trained with SGD” has “more implicit regularization for bigger models”?
  - But this behavior is not specific to implicit regularization of SGD and not specific to neural networks.

Implicit Regularization of SGD (as function of size)

- Why would implicit regularization of SGD increase with dimension?
  - H1: maybe SGD finds low-norm solutions?
    - In higher-dimensions, there is flexibility in global mins to have a low norm?
  - H2: maybe SGD stays closer to starting point as we increase dimension?
    - This would be more like a regularizer of the form $||w - w^0||$.





(pause)

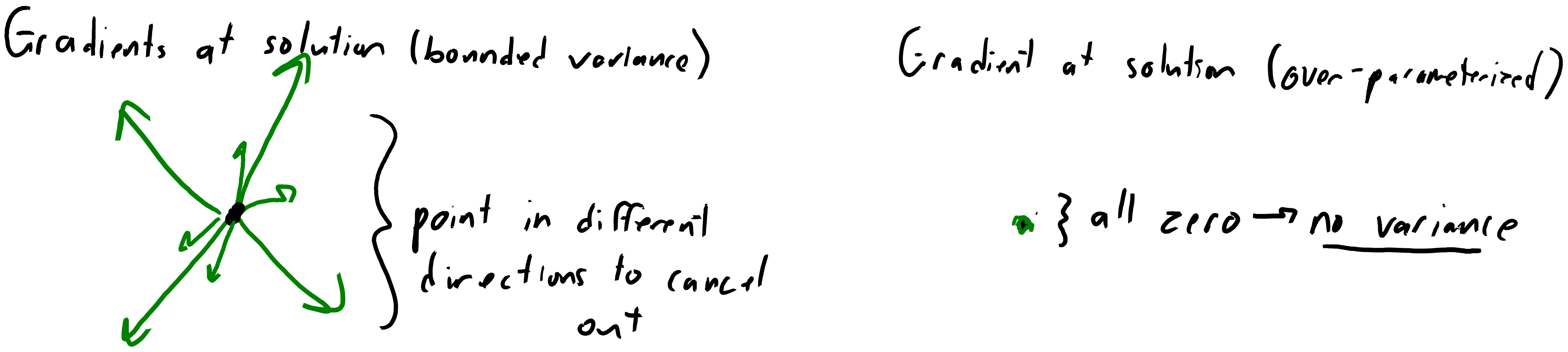
Over-Parameterized Models

- "Over-parameterization":
  - You have so many parameters that you can drive the loss to 0.
  - True for many modern deep neural networks.
    - Best models in many applications, implicit regularization may explain why they don't overfit.
  - Also true for linear models with a sufficiently-expressive basis/kernel.
    - You can make it true by making model more complicated.

- How does over-parameterization affect optimization?
  - Empirically and theoretically finding cases where SGD reaches global minimum.
  - Variance-reduced SGD doesn't seem to help with deep learning.
  - Adam optimizer seems to work well on many of these problems.
    - Despite it working poorly for many seemingly-easy problems.
    - Adam doesn't even converge under standard assumptions.

Strong Growth Condition (SGC)

- Over-parameterization changes behaviour of gradients at solution:



- Don't need SGD step size to go to zero in over-parameterized case.
  - We're going to show that plain SGD converges fast in over-parameterized case.
- Would explain why variance-reduction doesn't help for deep learning.
  - It's not needed, and might slow convergence.
  - And why Adam would work (acts more like a constant step size).

Strong Growth Condition (SGC)

- Recent works characterize over-parameterization in various ways.
- We'll consider the strong growth condition (SGC):

$$\mathbb{E}[\|g(x^k)\|^2] \leq \rho \|\nabla f(x^k)\|^2$$

  - Used by Tseng and Solodov in the 90s to analyze SGD on neural networks.
    - Under SGC, they showed that SGD converges with a constant step size.
    - This is possible because it implies variance goes to zero at a solution.

- The SGC is a very-strong assumption:
  - Assumes that gradient is zero at the solution for every training example:

$$\nabla f(x^k) = 0 \Rightarrow \text{every } g(x^k) = 0$$

  - Model is over-parameterized enough to "interpolate" (fit exactly) the data.

Convergence Rates under SGC

- Recall our expected progress by using SGD in descent lemma:

$$\mathbb{E}[f(w^{k+1})] \leq f(w^k) - \alpha_k \|\nabla f(w^k)\|^2 + \alpha_k^2 \frac{L}{2}\mathbb{E}[\|\nabla f_{i_k}(w^k)\|^2]$$

$\leq \rho \|\nabla f(w^k)\|^2$ under SGC

- Using SGC we get a progress bound of:

$$E[f(w^{k+1})] \leq f(w^k) - \alpha_k\left(1 - \frac{\alpha L \rho}{2}\right)\|\nabla f(w^k)\|^2$$

- Implications:
  - Decrease E[f(w^k)] for any constant step size $\alpha_k \leq \frac{2}{L\rho}$ (no need to have decreasing step size).
  - Convergence rate is basically same as deterministic gradient descent:
    - O((1 – $\mu/L\rho$)^k) for PL functions instead of O(1/k) (faster than VR methods for small ρ, without "finite data" assumption).
      - In this setting you can show that $1 \leq \rho \leq L_{max}/\mu$, so rate is "between gradient descent and an 'unaccelerated' gradient descent".
    - O(1/k) rate for convex functions instead O(1/$\sqrt{k}$) (again without "finite data" assumption).
    - O(1/k) rate for $||\nabla f(w^k)||^2$ instead of O(1/$\sqrt{k}$) (faster than fancier stochastic methods).

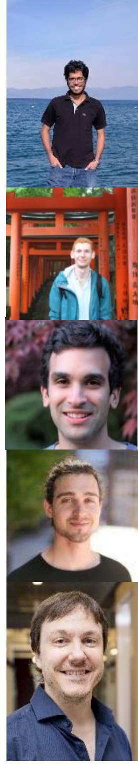
“Faster” SGD under the SGC (AI/Stats 2019)

- Sutskever, Martens, Dahl, Hinton [2013]:
  - Nesterov acceleration improves practical performance in some settings.
  - Acceleration is closely-related to momentum, which also helps in practice.
- Existing stochastic analyses only achieved partial acceleration.

Method	Regular	Accelerated	Comment
Deterministic	$\tilde{O}(n\kappa)$	$\tilde{O}(n\sqrt{\kappa})$	Unconditional acceleration
SGD + (var < σ^2)	$O\left(\frac{\sigma^2}{\epsilon}+\frac{\kappa}{\epsilon}\right)$	$O\left(\frac{\sigma^2}{\epsilon}+\sqrt{\frac{\kappa}{\epsilon}}\right)$	Faster if $\kappa > \sigma^2$
Variance Reduction	$\tilde{O}(n+\kappa)$	$\tilde{O}(n+\sqrt{n\kappa})$	Faster if $\kappa > n$
SGC + SGC	$\tilde{O}(\kappa)$	$\tilde{O}(\sqrt{\kappa})$	Unconditional acceleration

- Under SGC we show full acceleration (convex, appropriate parameters).
  - Special cases also shown by Liu and Belkin [2018], Jain et al. [2018]

"Painless" SGD under the SGC (NeurIPS 2019)



- Previous SGC/interpolation results relied on particular step-sizes.
  - Depending on values we don't know, like eigenvalues of Hessian.
- Existing methods to set step-size don't guarantee fast convergence.
  - Meta-learning, heuristics, adaptive, online learning, prob line-search.
- Under SGC, we showed you can can set the step-size as you go.
- Achieved (basically) optimal rate in a variety of settings:

Theorem 1 (Strongly-Convex). *Assuming interpolation, L-smoothness and μ strong-convexity of f, and convexity of the f_i, SGD with Armijo line-search with $c = 1/2$ in Equation 1 achieves the rate:*

$$\mathbb{E}\left[\|w_T - w^*\|^2\right] \leq \left(\max\left\{\left(1 - \frac{\mu}{L}\right), (1 - \eta_{max}\, \mu)\right\}\right)^T \|w_0 - w^*\|^2 .$$

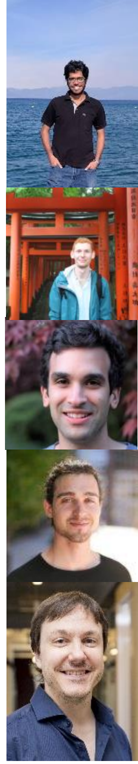
Theorem 2 (Convex). *Assuming interpolation and under L_i-smoothness and convexity of f_i's, SGD with Armijo line-search for all $c \geq 1/2$ in Equation 1 and iterate averaging achieves the rate:*

$$\mathbb{E}\left[f(\bar{w}_T) - f(w^*)\right] \leq \frac{c \cdot \max\left\{\frac{L_{max}}{2\,(1-c)}, \frac{1}{\eta_{max}}\right\}}{(2c-1)\, T} \|w_0 - w^*\|^2 .$$

Theorem 3 (Non-Convex). *Assuming the SGC with constant ρ and under L_i-smoothness of f_i's, SGD with Armijo line-search in Equation 1 with $c = \rho\, L_{max}$ and setting $\eta_{max} = 1$ achieves the rate:*

$$\min_{k=0,\ldots,T-1} \mathbb{E}\|\nabla f(w_k)\|^2 \leq \frac{\max\left\{\frac{L_{max}}{1-\rho\, L_{max}}, 2\right\} + 1}{T} \left[f(w_0) - f^*\right].$$

“Painless” SGD under the SGC (NeurIPS 2019)



- Key idea: Armijo line-search on the batch.
  - “Backtrack if you don’t improve cost on the batch relative to the norm of the batch’s gradient.”

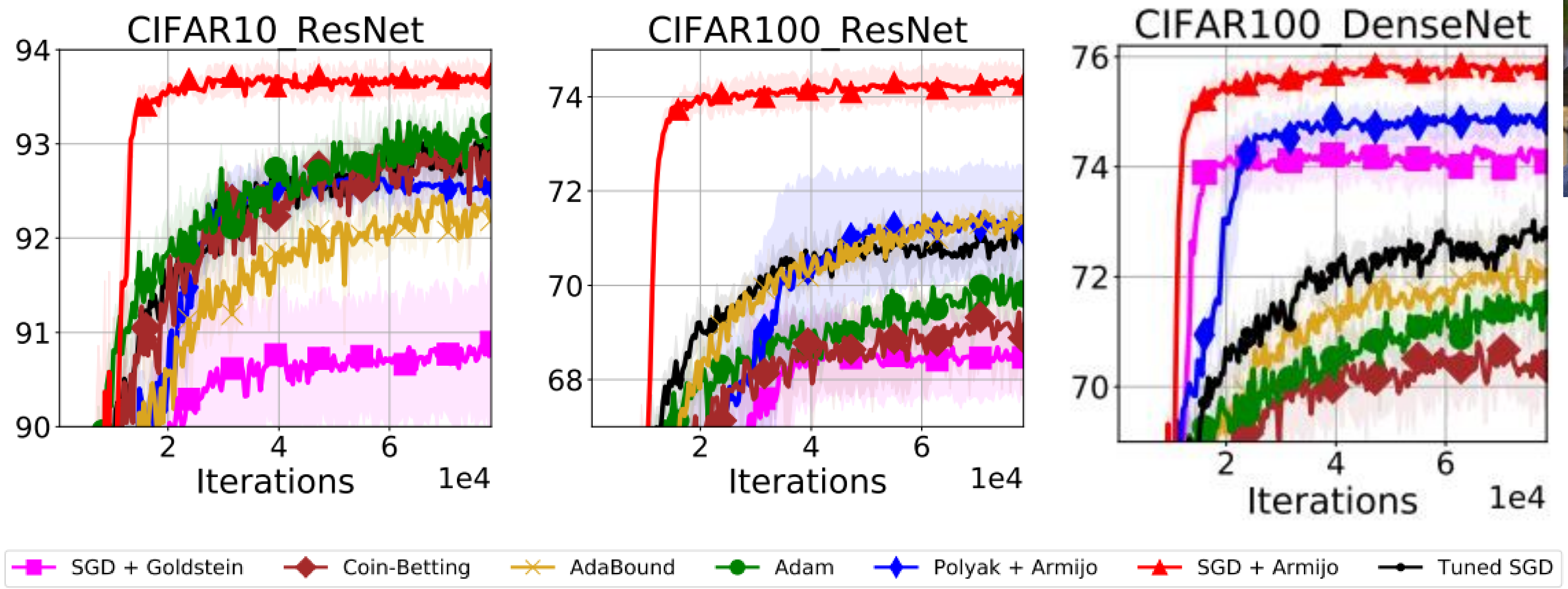
Algorithm 1 SGD+Armijo($f, w_0, \eta_{\max}, b, c, \beta, \gamma$, opt)

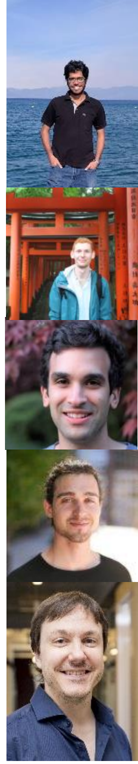
for $k = 1, \dots, T$ **do**
  $i_k \leftarrow$ sample mini-batch of size b
  $\eta \leftarrow$ reset$(\eta, \eta_{\max}, \gamma, b, k, \text{opt})/\beta$
  repeat
    $\eta \leftarrow \beta \cdot \eta$
    $w'_k \leftarrow w_k - \eta \nabla f_{ik}(w_k)$
  until $f_{ik}(w'_k) \leq f_{ik}(w_k) - c \cdot \eta \left\|\nabla f_{ik}(w_k)\right\|^2$
  $w_{k+1} \leftarrow w'_k$
end for
return w_{k+1}

- Backtracking guarantees steps are “not too big”.
- With appropriate initialization, guarantees steps are “not too small”.
  - Theory says that it’s at least as good as the best constant step-size.
- Requires an extra forward pass per iteration, and forward pass for each backtrack.
- We proposed a procedure to propose trial step sizes that works well in practice:
  - Slowly increases the step size, but median number of backtracking steps per iteration is 0.

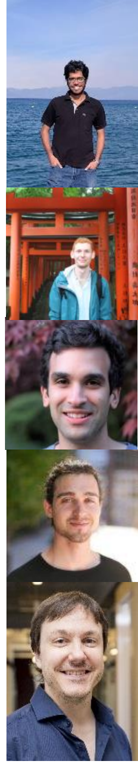
“Painless” SGD under the SGC (NeurIPS 2019)

- We did a variety of experiments, including training CNNs on standard problems.
  - Better in practice than any fixed step size, adaptive methods, alternative adaptive step sizes.





Discussion: Sensitivity to Assumptions



- To ease some of your anxiety/skepticism:
  - You don't need to run it to the point of interpolating the data, it just needs to be possible.
  - Results can be modified to handle case of being "close" to interpolation.
    - You get an extra term depending on your step-size and how "close" you are.
  - We ran synthetic experiments where we controlled the degree of over-parameterization:
    - If it's over-parameterized, the stochastic line search works great.
    - If it's close to being over-parameterized, it still works really well.
    - If it's far from being over-parameterized, it catastrophically fails.
  - Another group [Berrada, Zisserman, Pawan Kumar] proposed a similar method a few days later.
  - We've compared to a wide variety of existing methods to set the step size.

- To add some anxiety/skepticism:
  - My students said all the neural network experiments were done with batch norm.
  - They had more difficulty getting it to work for LSTMs ("first thing we tried" didn't work here).
  - Some of the line-search results have extra "sneaky" assumptions I would like to remove.

“Furious” SGD under the SGC (AI/Stats 2020)

- The reason “stochastic Newton” can’t improve rate is the variance.
- SGC gets rid of the variance, so stochastic Newton makes sense.
- Under SGC:
  - Stochastic Newton gets “linear” convergence with constant batch size.
    - Previous works required finite-sum assumption or exponentially-growing batch size.
  - Stochastic Newton gets “quadratic” with exponentially-growing batch.
    - Previous works required faster-than-exponential growing batch size for “superlinear”.

- The paper gives a variety of other results and experiments.
  - Self-concordant analysis, L-BFGS analysis, Hessian-free implementation.

SGD vs. Over-Parameterization

- For under-parameterized models, use variance reduction.
  - “Classic ML”.
- For over-parameterized models, don’t use variance reduction.
  - “Modern ML”.

- Try out the line-search, we want to make it a black box code.
  - It will helpful to know cases where it does and doesn’t work.

- Variance-reduction might still be relevant for deep learning:
  - Reducing Noise in GAN Training with Variance Reduced Extragradient. T. Chavdarova, G. Gidel, F. Fleuret, S. Lacoste-Julien [NeurIPS, 2019].

Summary

- Implicit regularization and double descent curves.
  - Possible explanations for why deep networks often generalize well.
- Over-parameterization:
  - Fast convergence of plain SGD with constant step size in this setting.
  - May explain weird optimization phenomenon in deep learning.
    - Why SGD is hard to be beat, why Adam works, why VR does not work.
  - Allows us to use tricks from deterministic setting:
    - Acceleration, line-search, second-order.

- The end (thanks for listening).