Introduction to Machine Learning
TTIC 31020

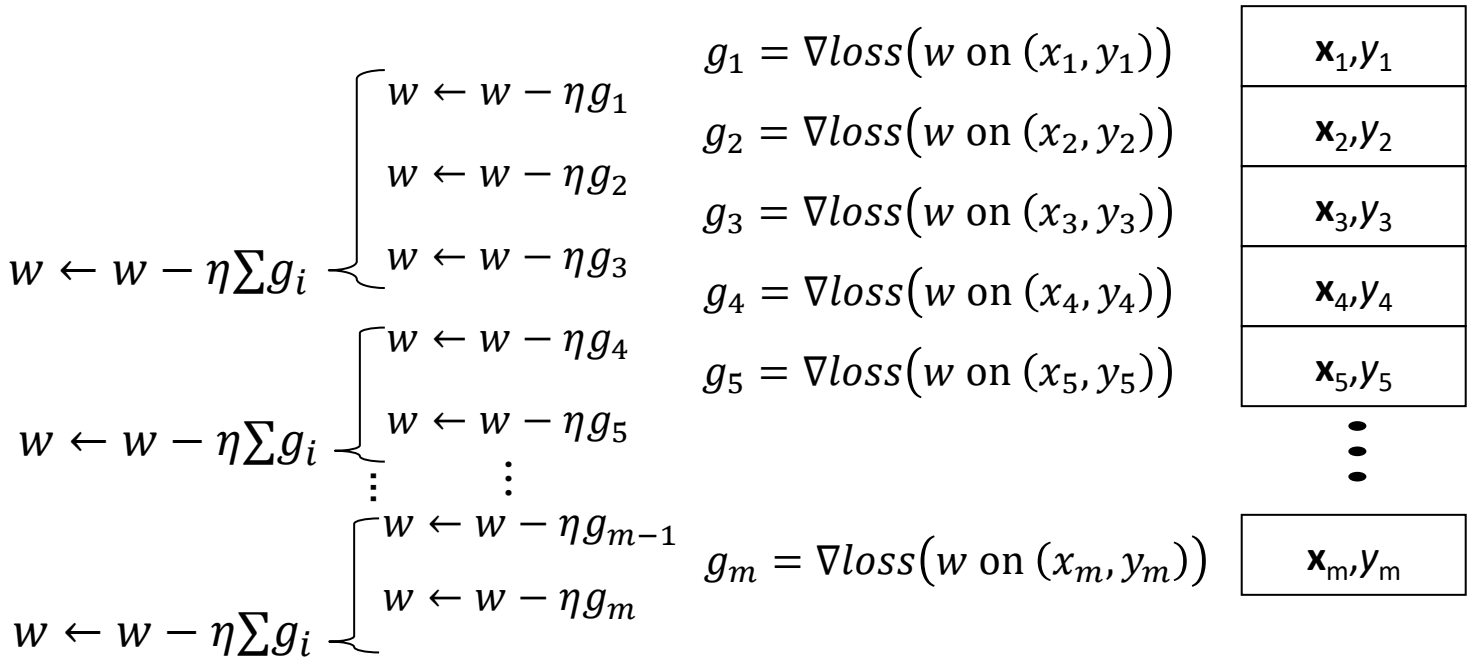
Prof. Nati Srebro

Lecture 10:
Stochastic Gradient Descent (SGD)
Stochastic Optimization

- Stochastic Gradient Descent Runtime: $O\left(\frac{B^2R^2}{\epsilon_{opt}^2}d\right)$
- Batch Gradient Descent Runtime: $O\left(\frac{B^2R^2}{\epsilon_{opt}^2}d \cdot m\right)$
- Mini-Batch Gradient Descent Runtime: $O\left(\frac{B^2R^2}{\epsilon_{opt}^2}b \cdot m\right)$

  Optimal b (in theory and in practice if counting #vector ops): $b = 1$

  In practice, moderate b just as good as $b = 1$, reduces overhead, allows parallelization



- Stochastic Gradient Descent Runtime: $O\left(\frac{B^2R^2}{\epsilon_{opt}^2}d\right)$
- Batch Gradient Descent Runtime: $O\left(\frac{B^2R^2}{\epsilon_{opt}^2}d \cdot m\right)$
- Mini-Batch Gradient Descent Runtime: $O\left(\frac{B^2R^2}{\epsilon_{opt}^2}b \cdot m\right)$

  Optimal b (in theory and in practice if counting #vector ops): $b = 1$

  In practice, moderate b just as good as $b = 1$, reduces overhead, allows parallelization

- GD actually better for smooth objectives, and under other assumptions
- Also alternate methods, eg Newton, approx. Newton (including BFGS) for smooth objective, Interior Point methods for handling non-smoothness/constraints
- Goal of much of optimization: $\log 1/\epsilon$ dependence (and many of the above achieve this)

- How small should ϵ_{opt} be?
- What about $L_{\mathcal{D}}(w)$, which is what we really care about?

Overall Analysis of $L_{\mathcal{D}}(w)$

- Recall for ERM: $L_{\mathcal{D}}(\hat{w}) \leq L_{\mathcal{D}}(w^*) + 2\sup_w |L_{\mathcal{D}}(w) - L_S(w)|$

  $\hat{w} = \arg\min_{\|w\| \leq B} L_S(w)$

  $w^* = \arg\min_{\|w\| \leq B} L_{\mathcal{D}}(w)$

- For ϵ_{opt} suboptimal ERM $\bar{w}$: → i.e. SGD with small number T

  $L_{\mathcal{D}}(\bar{w}) \leq \underbrace{L_{\mathcal{D}}(w^*)}_{\epsilon_{aprox}} + \underbrace{2\sup_w |L_{\mathcal{D}}(w) - L_S(w)|}_{\epsilon_{est} \leq 2\sqrt{\frac{B^2R^2}{m}}} + \underbrace{\left(L_S(\bar{w}) - L_S(\hat{w})\right)}_{\epsilon_{opt} \leq \sqrt{\frac{B^2R^2}{T}}}$

  It doesn't really matter how small ϵ_{opt} is, since we have a pretty big estimation error from ϵ_{est}.

- Take $\epsilon_{opt} \approx \epsilon_{est}$, i.e. $\#iter\ T \approx sample\ size\ m$
- To ensure $L_{\mathcal{D}}(w) \leq L_{\mathcal{D}}(w^*) + \epsilon$:

$$T, m = O\left(\frac{B^2R^2}{\epsilon^2}\right)$$

Q: How come in ML, we are okay w/ such crude estimates?

A: Because finite samples → est. err. is high.

SGD as an Ultimate Optimization Algorithm

- Runtime of SGD is $O(T \cdot d) = O\left(\frac{B^2 R^2}{\epsilon^2} d\right) = O(m \cdot d)$

- Linear in the number of required examples m
  - vs full GD which requires quadratic time $O(Tmd) = O(m^2 d)$
  - or interior point or matrix inversion methods, which require cubic or worse runtime

on each new iteration, we have to process the entire data set again.

- SGD runtime is linear in time it takes to read data set

  ➔Can't be improved beyond small constant factor **without additional assumptions**

  We do care about marginal improvements, but these are all variations of SGD.

Note: Any non-SGD algorithm will basically read the entire data set on each iteration.

As before: we motivated SGD by trying to minimize the empirical error

i.e. $\min_{\|w\|\le B} L_S(w)$.

But we don't actually care about empirical error, we care about the population error.

SGD as a Learning Algorithm: SGD on $L_{\mathcal{D}}(w)$

Claim: We can use SGD to minimize the $\min_w L_{\mathcal{D}}(w)$ population error!

we are directly estimating the population error!

use $g^{(t)} = \nabla_w loss\left(h_{w^{(t)}}(x); y\right)$ for random $y, x \sim \mathcal{D}$ ➔ $\mathbb{E}\left[g^{(t)}\right] = \nabla L_{\mathcal{D}}(w)$

Initialize $w^{(0)} = 0$

At iteration t:

- Draw $x_t, y_t \sim \mathcal{D}$
- $w^{(t+1)} \leftarrow w^{(t)} - \eta_t \nabla_w loss\left(h_{w^{(t)}}(x); y\right) = w^{(t)} - \eta_t loss'\left(\left\langle w^{(t)}, \phi(x)\right\rangle; y\right)\phi(x)$

Return $\bar{w}^{(T)} = \frac{1}{T}\sum_{t=1}^{T} w^{(t)}$

$h_w(x) = \langle w, \phi(x)\rangle$

SGD suboptimality guarantee: $L_{\mathcal{D}}\left(\bar{w}^{(T)}\right) \le \inf_{\|w\|_2 \le B} L_{\mathcal{D}}(w) + \sqrt{\frac{B^2R^2}{T}}$

➔ $m = T = O\left(\frac{B^2R^2}{\epsilon^2}\right)$ → since we draw a new sample on each iteration.

Our objective is now the population error, and our suboptimality guarantee is now on the population error

Note: We need $g^{(t)}$ to be unbiased, thus, we have to draw x_t, y_t iid.

Stochastic Optimization: $\min_{\omega} F(\omega) := \mathbb{E}_z[f(\omega, z)]$

based only on stochastic information on F

- Only unbiased estimates of $F(\omega), \nabla F(\omega)$
- No direct access to F

E.g., fixed $f(\omega, z)$ but $\mathcal{D}$ unknown We only see instances of $f(w,z)$, $z \sim$ Dunknown

- Optimize $F(\omega)$ based on iid sample $z_1, z_2, \ldots, z_m \sim \mathcal{D}$
- $g = \nabla f(\omega, z_t)$ is unbiased estimate of $\nabla F(\omega)$

Traditional applications

- Optimization under uncertainty
  - Uncertainty about network performance
  - Uncertainty about client demands
  - Uncertainty about system behavior in control problems
- Complex systems where its easier to sample then integrate over z

↳ easier to get unbiased estimates

Stochastic Optimization: $\min_{\omega} F(\omega) := \mathbb{E}_z[f(w,z)]$

Learning *using* Stochastic Optimization

- Underlying goal: $\min_h L_{\mathcal{D}}(h)$ this is our learning rule.
- Learning rule: $\mathrm{ERM}(S) = \widehat{w}_B = \arg\min_{\|w\|_2} L_S(w)$ or $RERM(S) = \widehat{w}_\lambda = \arg\min L_S(w) + \lambda\|w\|_2^2$

  ➔ Inductive bias specified explicitly by $\|w\|_2 \le B$ or $+\lambda\|w\|_2^2$

  Generalization in terms of $\|w\|_2$ complexity control: $L_{\mathcal{D}}(\widehat{w}_B) \le L_{\mathcal{D}}(w^*) + O\left(\sqrt{\frac{B^2R^2}{m}}\right)$ ($\|w^*\| \le B$)
- Use stochastic optimization on $F(w) = L_S(w)$ (maybe$+\lambda\|w\|^2$) to implement "min" in learning rule
- SGD (as an algorithm for implementing "min"): runtime $O(md)$, linear in data set size

Learning *as* Stochastic Optimization

- Optimization objective: $F(w) = \min_w L_{\mathcal{D}}(w)$ $= \mathbb{E}_{(x,y)\sim\mathcal{D}}[h_w(x) \neq y]$
- Learning rule: $SGD(S) = \overline{w}^m = \frac{1}{m}\sum_{t=1}^m w^{(t)}$ where $w^{(t+1)} \leftarrow w^{(t)} - \eta\nabla\ell\left(h_{w^{(t)}}(x_t), y_t\right)$
- Directly guarantees generalization: $L_{\mathcal{D}}\left(\overline{w}^{(T)}\right) \le L_{\mathcal{D}}(w^*) + O\left(\sqrt{\frac{B^2R^2}{T}}\right)$ with $T = m$

Stochastic Optimization: $\min_{\omega} F(\omega) := \mathbb{E}_z[f(\omega, z)]$

based on i.i.d samples $z_1, z_2, z_3, \ldots \sim \mathcal{D}$

- Distribution $\mathcal{D}$ unknown; No direct access to $F(\omega)$
- Can obtain unbiased estimates of $F(\omega)$, $\nabla F(\omega)$, etc from z_i

Supervised Learning *as* Stochastic Optimization

$$\min_{h:X\to Y} L(h) = E_{x,y\sim\mathcal{D}}[\underbrace{loss(h(x), y)}_{f(h,(x,y)) = loss(h(x),y)}]$$

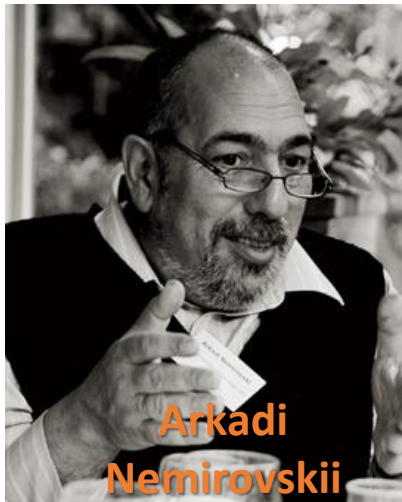
based on sample $(x_1, y_1), \ldots, (x_m, y_m) \sim \mathcal{D}$

- $\omega = h \in Y^X$ → optimization variable.
- $z = (x, y)$ → stochasticity.
- $f(h, z) = loss(h(x); y)$

Vapnik's "General Learning" *is* Stochastic Optimization

$$\min_h L(h) = \mathbb{E}_z[\ell(h, z)]$$

based on sample $z_1, z_2, \ldots \sim \mathcal{D}$



Generalized (Statistical) Learning ≡ Stochastic Optimizatoin

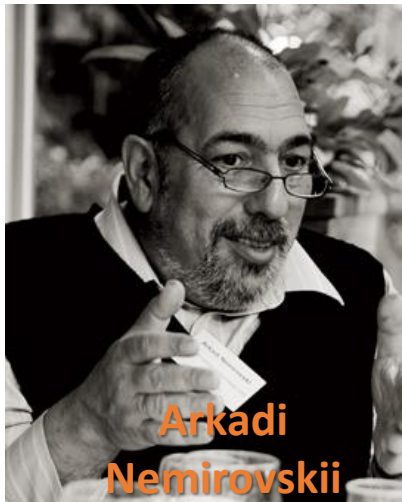
$$\min_{h\in\mathcal{H}} F(h) = \mathbb{E}_{z\sim\mathcal{D}}\left[f(h,z)\right] \text{ based on } z_1,\ldots,z_m \sim iid\ \mathcal{D}$$

- Supervised learning:
  - $z = (x, y)\ ,\ x \in \mathcal{X}, y \in \mathcal{Y}$
  - $h\colon \mathcal{X} \to \mathcal{Y}$
  - $f(h,z) = loss(h(x); y)$
- "Unsupervised learning", e.g. k-means clustering:
  - $z = x \in \mathbb{R}^d$,
  - $h = (\mu[1], \mu[2], \ldots, \mu[k]) \in \mathbb{R}^{d\times k}$ specified k cluster centers
  - $f\left((\mu[1], \mu[2], \ldots, \mu[k])\ , x\ \right) = \min_i \|\mu[i] - x\|^2$
- Density estimation:
  - $z = x$ in some measurable space $\mathcal{Z}$ (e.g. $\mathbb{R}^d$)
  - h specifies probability density $p_h(z)$
  - $f(h,z) = -\log p_h(z)$
- Learning a good route with random traffic:
  - $z =$ traffic delays on each road segment
  - $h =$ route chosen (indicator over road segments in route)
  - $f(h,z) = \langle h, z\rangle =$total delay along route

We have a distribution and want to centers of masses, i.e. means.

want x to be close to some mean.

want $p_h(z)$ to be high for what we observe.



Statistical Learning

vs

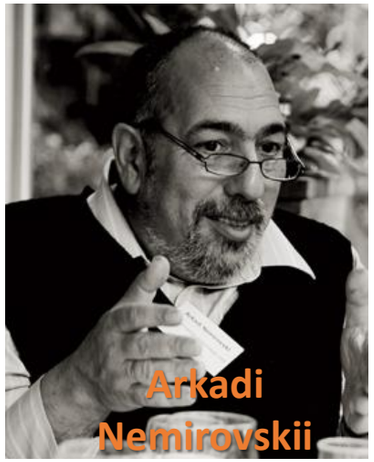
Stochastic Optimization

- Focus on sample size → didn't care about run time
- What can be done with a fixed number of samples?
  disregarding compute, $\arg\min L_S(h)$ is fine
- Abstract hypothesis classes
  - linear predictors, but also combinatorial hypothesis classes, eg decision trees, conjunctions, formulas
  - generic measures of complexity such as VC-dim, fat shattering, Radamacher
- Also non-convex classes and loss functions

↳ didn't really care about implementable loss fcns.

- Focus on computational efficiency → computationally feasible.
- Generally assumes unlimited sampling
  - as in monte-carlo methods for complicated objectives
- Optimization variable is a vector in a normed space
  - $\omega \in \mathbb{R}^d$, or perhaps infinite dimensional Banach space
  - complexity control through norm

  ↳ Need to be able to implement these methods
- Mostly convex objectives



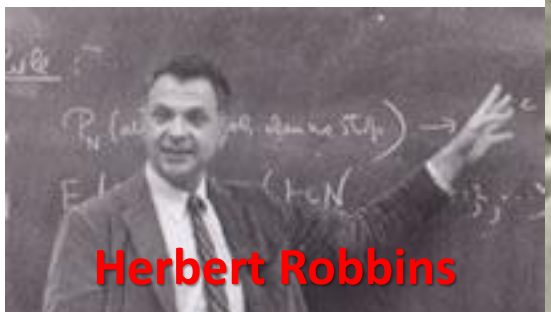
When SVMs were invented and norm was used for penalization in learning was the connection made. (20 yrs after these were invented) 3

Two Approaches to Stochastic Optimization / Learning

$$\min_{w \in \mathcal{W}} F(w) = \mathbb{E}_{z \sim \mathcal{D}}[f(w, z)]$$

- Empirical Risk Minimization (ERM) / Sample Average Approximation (SAA):
  - Collect sample $z_1, \ldots, z_m$
  - Minimize $F_S(w) = \frac{1}{m} \sum_i f(w, z_i)$ → reduce Stoch. opt. into deterministic optimization.
  - Analysis typically based on Uniform Concentration

- Stochastic Approximation (SA): [Robins Monro 1951]
  - Update $w^{(t)}$ based on z_t
    - E.g., based on $g^{(t)} = \nabla f(w, z_t)$
  - Simplest method: **stochastic gradient descent**



SGD for Machine Learning

$$\min_w L(w)$$

Direct SA (SGD) Approach:

Initialize $w^{(0)} = 0$

At iteration t:

- Draw $x_t, y_t \sim \mathcal{D}$ (Draw point inside.)
- If $y_t \langle w^{(t)}, \phi(x_t) \rangle < 1$,
  $w^{(t+1)} \leftarrow w^{(t)} + \eta_t y_t \phi(x_t)$
  else: $w^{(t+1)} \leftarrow w^{(t)}$

Return $\bar{w}^{(T)} = \frac{1}{T} \sum_{t=1}^{T} w^{(t)}$

- **Fresh sample at each iteration**, $m = T$
  ➔ one pass over the data
- No need to project nor require $\|w\| \leq B$
- Implicit regularization via early stopping

SGD on ERM:

$$\min_{\|w\|_2 \leq B} L_S(w)$$

Draw outside.

Draw $(x_1, y_1), \ldots, (x_m, y_m) \sim \mathcal{D}$

Initialize $w^{(0)} = 0$

At iteration t:

- Pick $i \in 1 \ldots m$ at random
- If $y_i \langle w^{(t)}, \phi(x_i) \rangle < 1$,
  $w^{(t+1)} \leftarrow w^{(t)} + \eta_t y_i \phi(x_i)$
  else: $w^{(t+1)} \leftarrow w^{(t)}$
- $w^{(t+1)} \leftarrow proj\ w^{(t+1)}\ to\ \|w\| \leq B$

Return $\bar{w}^{(T)} = \frac{1}{T} \sum_{t=1}^{T} w^{(t)}$

Need to construct hypothesis class. Since we are reusing the samples.

- Can have $T > m$ iterations (multiple passes)
- Need to project to $\|w\| \leq B$
- Explicit regularization via $\|w\|$

Note: for SGD on ERM, we reuse our samples, causing overfit, if we don't regularize.

- for SA, we do not reuse samples, so we don't get overfit, so we don't explicitly regularize. We implicitly regularize by early stopping.

SGD for Machine Learning

$$\min_w L(w)$$

Direct SA (SGD) Approach:

Initialize $w^{(0)} = 0$

At iteration t:

- Draw $x_t, y_t \sim \mathcal{D}$
- If $y_t \langle w^{(t)}, \phi(x_t) \rangle < 1$,
  $w^{(t+1)} \leftarrow w^{(t)} + \eta_t y_t \phi(x_t)$
  else: $w^{(t+1)} \leftarrow w^{(t)}$

Return $\bar{w}^{(T)} = \frac{1}{T} \sum_{t=1}^{T} w^{(t)}$

- **Fresh sample at each iteration**, $m = T$
  ➔ one pass over the data
- No need to project nor require $\|w\| \leq B$
- Implicit regularization via early stopping

SGD on RERM:

$$\min L_S(w) + \frac{\lambda}{2} \|w\|^2$$

Draw $(x_1, y_1), \ldots, (x_m, y_m) \sim \mathcal{D}$

Initialize $w^{(0)} = 0$

At iteration t:

- Pick $i \in 1 \ldots m$ at random
- If $y_i \langle w^{(t)}, \phi(x_i) \rangle < 1$,
  $w^{(t+1)} \leftarrow w^{(t)} + \eta_t y_i \phi(x_i)$
  else: $w^{(t+1)} \leftarrow w^{(t)}$
- $w^{(t+1)} \leftarrow w^{(t+1)} - \lambda \eta_t \, w^{(t)}$

Return $\bar{w}^{(T)} = \frac{1}{T} \sum_{t=1}^{T} w^{(t)}$

Same idea.

- Can have $T > m$ iterations (multiple passes)
- Need to shrink w
- Explicit regularization via $\|w\|$

SGD for Machine Learning

$$\min_w L(w)$$

Direct SA (SGD) Approach:

Initialize $w^{(0)} = 0$

At iteration t:

- Draw $x_t, y_t \sim \mathcal{D}$
- If $y_t \langle w^{(t)}, \phi(x_t) \rangle < 1$,
  $w^{(t+1)} \leftarrow w^{(t)} + \eta_t y_t \phi(x_t)$
  else: $w^{(t+1)} \leftarrow w^{(t)}$

$\eta_t = \sqrt{B^2/R^2 t}$

Return $\bar{w}^{(T)} = \frac{1}{T}\sum_{t=1}^{T} w^{(t)}$

$$L(\bar{w}^{(T)}) \leq L(w^*) + \sqrt{\frac{B^2R^2}{T}}$$

SGD on ERM:

$$\min_{\|w\|_2 \leq B} L_S(w)$$

Draw $(x_1, y_1), \ldots, (x_m, y_m) \sim \mathcal{D}$

Initialize $w^{(0)} = 0$

At iteration t:

- Pick $i \in 1 \ldots m$ at random
- If $y_i \langle w^{(t)}, \phi(x_i) \rangle < 1$,
  $w^{(t+1)} \leftarrow w^{(t)} + \eta_t y_i \phi(x_i)$
  else: $w^{(t+1)} \leftarrow w^{(t)}$
- $w^{(t+1)} \leftarrow proj\ w^{(t+1)}\ to\ \|w\| \leq B$

Return $\bar{w}^{(T)} = \frac{1}{T}\sum_{t=1}^{T} w^{(t)}$

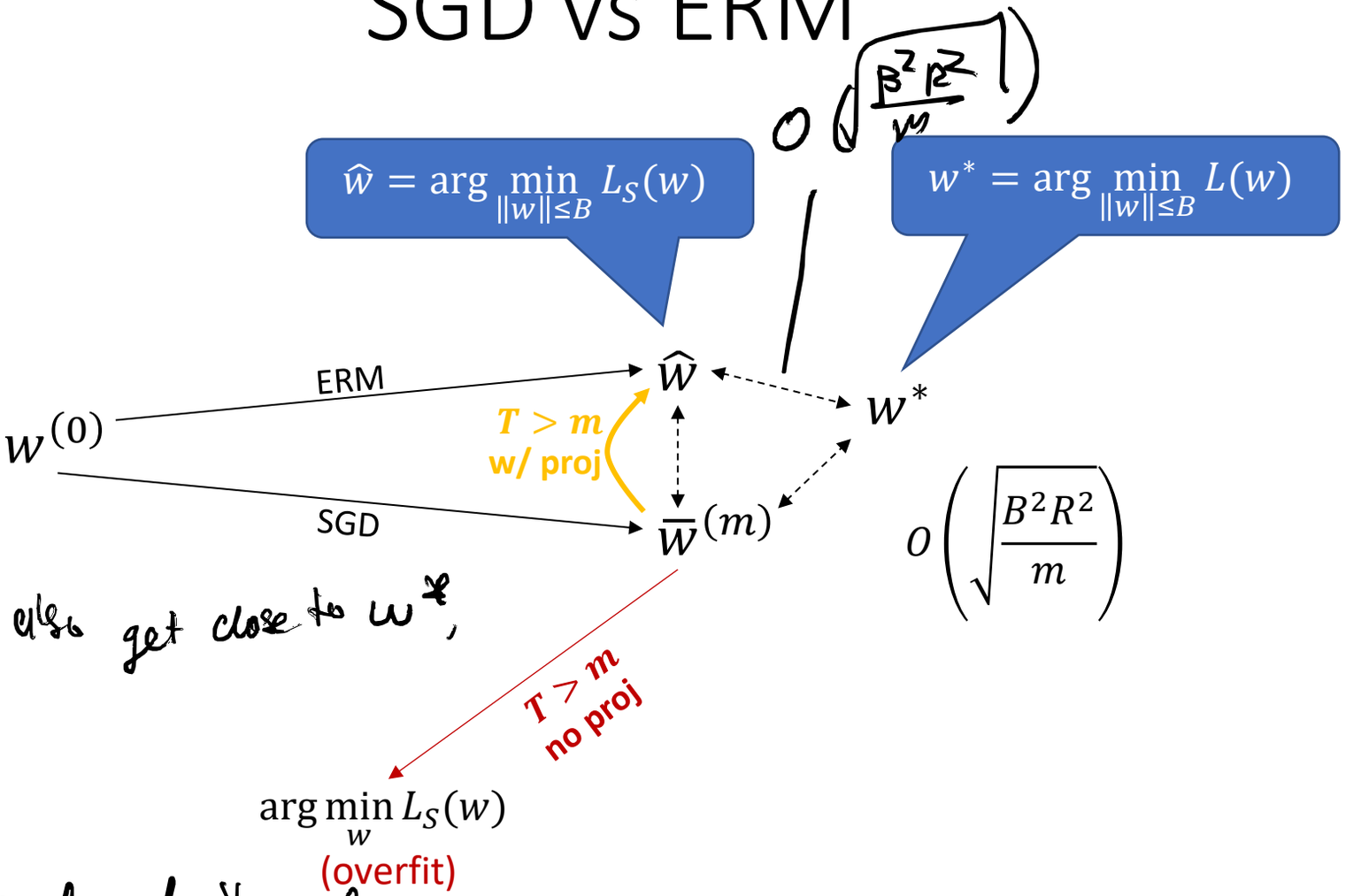
$$L(\bar{w}^{(T)}) \leq L(w^*) + 2\sqrt{\frac{B^2R^2}{m}} + \sqrt{\frac{B^2R^2}{T}}$$

$w^* = \arg\min_{\|w\| \leq B} L(w)$

Q: Where does B come from?

A: The learning rate. $\eta_t = \sqrt{B^2/R^2 t}$

SGD vs ERM



We run SGD and also get close to w^*,

But not at $\hat{w}$.

If we run SGD and don't draw new samples, we have two cases:

- w/ projection step $\bar{w}^{(T)}(m) \to \hat{w}$
- w/out projection $\bar{w}^{(T)}(m) \to$ overfit.

SGD for Machine Learning

$$\min_w L(w)$$

Direct SA (SGD) Approach:

Initialize $w^{(0)} = 0$

At iteration t:

- Draw $x_t, y_t \sim \mathcal{D}$
- If $y_t \langle w^{(t)}, \phi(x_t) \rangle < 1$,
  $w^{(t+1)} \leftarrow w^{(t)} + \eta_t y_t \phi(x_t)$
  else: $w^{(t+1)} \leftarrow w^{(t)}$

Return $\bar{w}^{(T)} = \frac{1}{T} \sum_{t=1}^{T} w^{(t)}$

$$L(\bar{w}^{(T)}) \leq L(w^*) + \sqrt{\frac{B^2 R^2}{T}}$$

SGD on ERM:

$$\min_{\|w\|_2 \leq B} L_S(w)$$

Draw $(x_1, y_1), \ldots, (x_m, y_m) \sim \mathcal{D}$

Initialize $w^{(0)} = 0$

At iteration t:

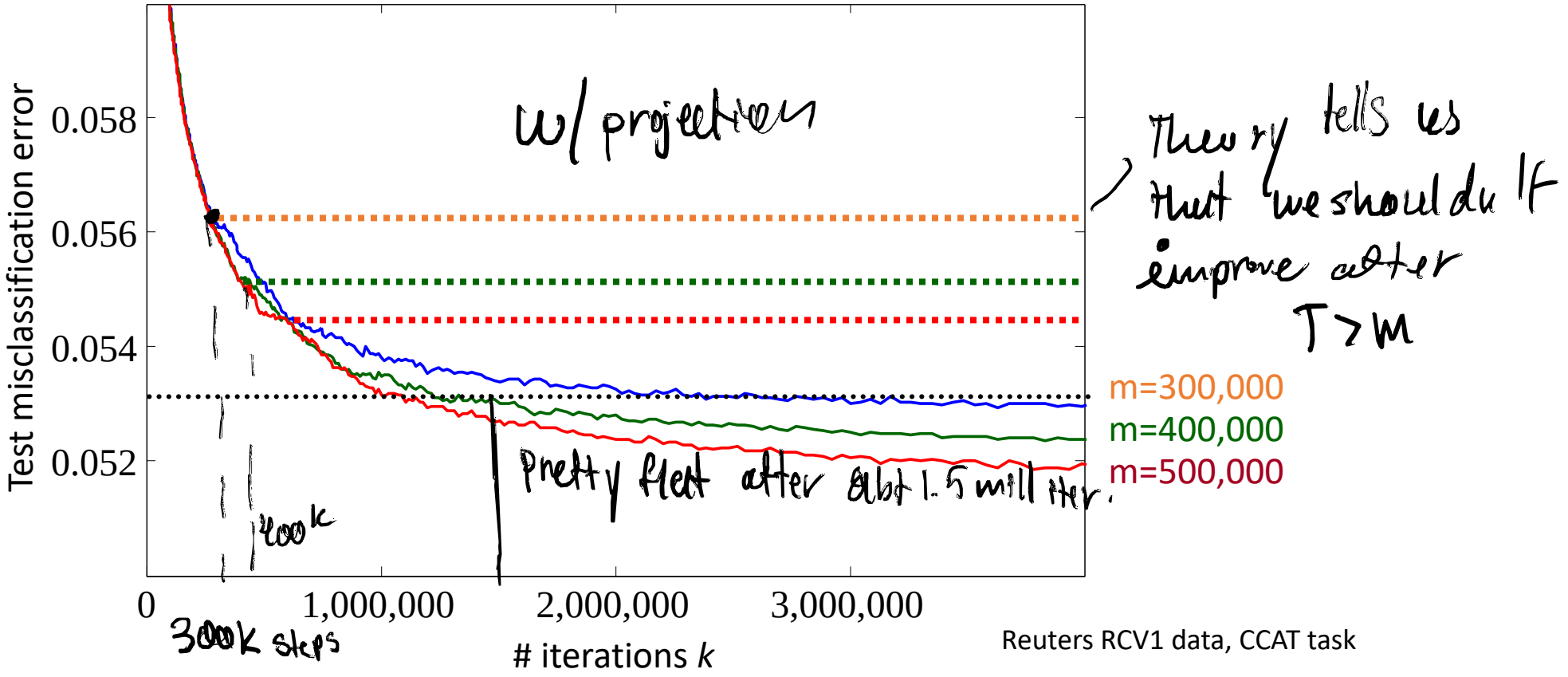
- Pick $i \in 1 \ldots m$ at random
- If $y_i \langle w^{(t)}, \phi(x_i) \rangle < 1$,
  $w^{(t+1)} \leftarrow w^{(t)} + \eta_t y_i \phi(x_i)$
  else: $w^{(t+1)} \leftarrow w^{(t)}$
- $w^{(t+1)} \leftarrow proj\ w^{(t+1)}\ to\ \|w\| \leq B$

Return $\bar{w}^{(T)} = \frac{1}{T} \sum_{t=1}^{T} w^{(t)}$

$$L(\bar{w}^{(T)}) \leq L(w^*) + 2\sqrt{\frac{B^2 R^2}{m}} + \sqrt{\frac{B^2 R^2}{T}}$$

$$w^* = \arg\min_{\|w\| \leq B} L(w)$$

Mixed Approach: SGD on ERM



Reuters RCV1 data, CCAT task

- The mixed approach (reusing examples) can make sense
- Still: fresh samples are better

  With a larger training set, can reduce generalization error faster

  Larger training set means *less* runtime to get target generalization error

Direct SA/SGD Approach

(Learning ***as*** Stochastic Optimization)

SGD on the objective $L_{\mathcal{D}}(w)$

SGD is the Learning Rule

One pass/epoch: "online", $T = m$
(processes each example once)

SGD on ERM

(Learning ***using*** Stochastic Optimization)

SGD on $L_S(w)$

Learning rule: $\mathrm{ERM}(S) = \widehat{w}_B = \arg\min_{\|w\|_2} L_S(w)$

or $RERM(S) = \widehat{w}_\lambda = \arg\min L_S(w) + \lambda\|w\|_2^2$

SGD as an Optimization Algorithm for min

Multiple passes/epochs, can have $T > m$
(can processes examples multiple times)

Explicit complexity control: $\|w\|_2 \leq B$ or $+\lambda\|w\|_2^2$

Generalization from explicit complexity control:

$$L_{\mathcal{D}}(\widehat{w}_B) \leq L_{\mathcal{D}}(w^*) + O\left(\sqrt{\frac{\|w^*\|_2^2\|\phi\|_2^2}{m}}\right)$$

Online learning:

At each iteration $t = 1,2, \ldots$

- Receive instance x_t
- Predict a label $\hat{y}_t = h^{(t)}(x_t)$
- Receive label y_t,
- Update $h^{(t+1)}$ based on (x_t, y_t)

Stochastic Approximation (e.g. SGD):

At each iteration $t = 1,2, \ldots$

receive (x_t, y_t)

update $h^{(t+1)}$ based on (x_t, y_t)

- Goal in realizable case ($\exists_{h^* \in \mathcal{H}} h^*(x_t) = y_t$): #mistakes (ie $h^{(t)}(x_t) \neq y_t$)

$$\frac{1}{m} \sum_t \ell^{01}\left(h^{(t)}(x_t), y_t\right) \leq \epsilon$$

#mistakes/m

$\epsilon \to 0$

- Goal in agnostic case: regret versus best $h^* \in \mathcal{H}$ in hindsight

$$\frac{1}{m} \sum_t \ell\left(h^{(t)}(x_t), y_t\right) \leq \inf_{h^* \in \mathcal{H}} \frac{1}{m} \sum_t \ell(h^*(x_t), y_t) + \epsilon$$

regret

Online regret guarantees beyond scope of course

Online Gradient Descent

Online learning:

At each iteration $t = 1,2, \ldots$

- Receive instance x_t
- Predict a label $\hat{y}_t = h_{w^{(t)}}(x_t)$
- Receive label y_t, suffer loss $\ell\left(h_{w^{(t)}}, y_t\right)$
- Update $w^{(t+1)}$ based on (x_t, y_t)

$$\begin{aligned} \boldsymbol{w^{(t+1)}} &\leftarrow \boldsymbol{w^{(t)}} - \boldsymbol{\eta_t \nabla_w \ell(h_{w^{(t)}}(x_t), y_t)} \\ &= w^{(t)} - \eta_t \nabla_w \ell(\langle \boldsymbol{w^{(t)}}, \boldsymbol{\phi(x_t)} \rangle, y_t) \\ &= w^{(t)} - \eta_t \ell'\left(\left\langle \boldsymbol{w^{(t)}}, \boldsymbol{\phi(x_t)} \right\rangle, \boldsymbol{y_t}\right) \boldsymbol{\phi(x_t)} \end{aligned}$$

For linear pred $h_w(x) = \langle w, \phi(x) \rangle$

- If $\ell(h_w(x), y)$ is convex and ρ-Lipschitz in w

$$\frac{1}{m} \sum_t \ell\left(h_{w^{(t)}}, y_t\right) \leq \inf_{\|w\|_2 \leq \boldsymbol{B}} \frac{1}{m} \sum_t \ell(h_w(x_t), y_t) + \sqrt{\frac{B^2 \rho^2}{m}}$$

- If $h_w(x) = \langle w, \phi(x) \rangle$, $\|\phi(x)\|_2 \leq R$ and $\ell(z, y)$ is 1-Lipschitz in z:

$$\frac{1}{m} \sum_t \ell\left(\left\langle w^{(t)}, \phi(x_t) \right\rangle, y_t\right) \leq \inf_{\|w\|_2 \leq \boldsymbol{B}} \frac{1}{m} \sum_t \ell(\langle w, \phi(x_t) \rangle, y_t) + \sqrt{\frac{B^2 R^2}{m}}$$

Online regret guarantees beyond scope of course

Perceptron as OGD

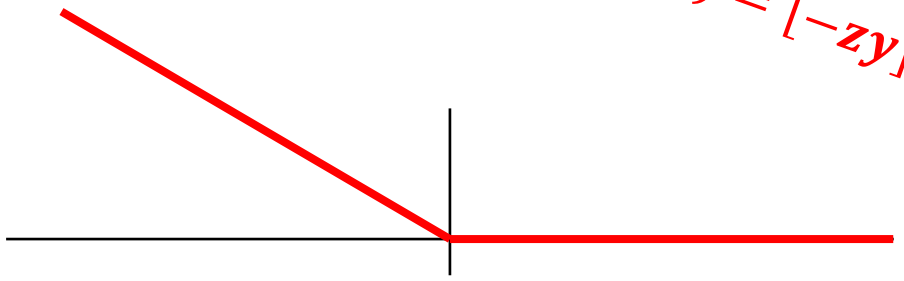
Online learning:

At each iteration $t = 1,2, \ldots$

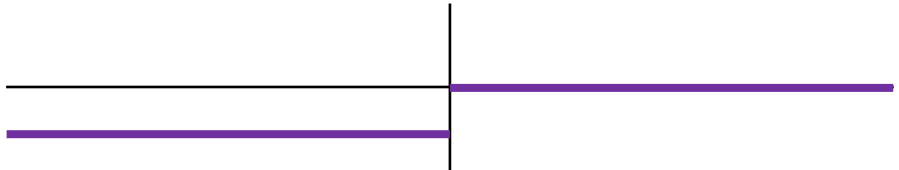
- Receive instance x_t
- Predict a label $\hat{y}_t = h_{w^{(t)}}(x_t)$
- Receive label y_t, suffer loss $\ell\left(h_{w^{(t)}}, y_t\right)$
- Update $w^{(t+1)}$ based on (x_t, y_t)

$$\boldsymbol{w^{(t+1)}} \leftarrow \boldsymbol{w^{(t)}} - \boldsymbol{\eta_t \nabla_w \ell(h_{w^{(t)}}(x_t), y_t)}$$
$$= w^{(t)} - \eta_t \nabla_w \ell(\langle \boldsymbol{w^{(t)}}, \boldsymbol{\phi(x_t)} \rangle, y_t)$$
$$= w^{(t)} - \eta_t \ell'\left(\left\langle \boldsymbol{w^{(t)}}, \boldsymbol{\phi(x_t)} \right\rangle, \boldsymbol{y_t}\right) \boldsymbol{\phi(x_t)}$$

$\ell(z, y) = [-zy]_+$



$$\ell'(\quad z \quad, y) = \begin{cases} -1, & y \neq sign(z) \text{ i.e., } yz < 0 \\ 0, & y = sign(z) \text{ i.e., } yz > 0 \end{cases}$$



At iteration t:

- Receive x_t
- Predict $\hat{y}_t = sign\left(\langle w^{(t)}, \phi(x_t) \rangle\right)$
- Receive y_t
- If $\boldsymbol{y_t \neq \hat{y}_t}$,
  $\boldsymbol{w^{(t+1)} \leftarrow w^{(t)} + y_t \phi(x_t)}$
  else: $\boldsymbol{w^{(t+1)} \leftarrow w^{(t)}}$

$$\frac{1}{m}\sum_t \ell\left(h_{w^{(t)}}, y_t\right) \le \inf_w \frac{1}{m}\sum_t \ell(h_w(x_t), y_t) \; + Regret$$

Online algorithm A

e.g. Online Gradient Descent:

$$w^{(t+1)} \leftarrow w^{(t)} - \eta \nabla_w \left(h_{w^{(t)}}(x_t, y_t)\right)$$

or online Perceptron

Realizable Online-to-Batch

(if $\exists w^* \; L_S(w^*) = 0$)

Input: $S = (x_1, y_1)..(x_m, y_m) \sim \mathcal{D}^m$

While $y_i w^{(t)}(x_i) < 0$,

feed (x_i, y_i) into A to get $w^{(t+1)}$

Output $w^{(T)}$

Empirical Optimization: $L_S\left(w^{(T)}\right) = 0$

Generalization: $\mathbb{E}\left[L_{\mathcal{D}}\left(w^{(T)}\right)\right] \le \frac{\#\text{mistakes}}{m} = Regret$

One-Pass Online-to-Batch

Input: $S = (x_1, y_1)..(x_m, y_m) \sim \mathcal{D}^m$

For $t = 1 \ldots \boldsymbol{m}$,

feed (x_t, y_t) into A to get $w^{(t+1)}$

Output $\bar{w} = \frac{1}{m}\sum w^{(t)}$

Generalization:

$$\mathbb{E}[L_{\mathcal{D}}(\bar{w})] \le \inf_{w^*} L_{\mathcal{D}}(w^*) + Regret$$

Onlined Gradient Descent **[Zinkevich 03]** → online2stochastic **[Cesa-Binachi et al 04]** → Stochastic Gradient Descent **[Nemirovski Yudin 78]**

Online Learning vs Stochastic Approximation

- In both Online Setting and Stochastic Approximation
  - Receive samples sequentially
  - Update predictor after each sample

- But, in Online Setting:
  - Objective is empirical regret, i.e. behavior on observed instances
  - Every point is both a training point and a test point
  - (x_t, y_t) chosen arbitrarily (no distribution involved), could be non stationary, non independent, adapt based on predictor, anything goes
- Whereas in Stochastic Approximation:
  - Objective is $L(h) = \mathbb{E}_{x,y}[loss(h(x), y)]$, i.e. behavior on "future" samples $(x, y) \sim \mathcal{D}$
  - i.i.d. *training* samples $(x_t, y_t) \sim \mathcal{D}$
  - Have same source distribution $\mathcal{D}$ for train and test crucial

- Stochastic Approximation is a computational approach, Online Learning is an analysis setup
  - E.g. "Majority" is a valid online algorithm and makes sense to analyze as such

Direct SA/SGD Approach

(Learning ***as*** Stochastic Optimization)

SGD on the objective $L_{\mathcal{D}}(w)$

SGD as a Learning Rule

One pass/epoch: "online", $T = m$
(processes each example once)

Generalization from SGD regret guarantee

$$L_{\mathcal{D}}\left(\overline{w}^{T}\right) \leq L_{\mathcal{D}}(w^*) + O\left(\sqrt{\frac{\|w^*\|_2^2 \|\phi\|_2^2}{T}}\right)$$

What is the inductive bias?

How is it specified or used in SGD?

SGD on ERM

(Learning ***using*** Stochastic Optimization)

SGD on $L_S(w)$

Learning rule: $\mathrm{ERM}(S) = \widehat{w}_B = \arg\min_{\|w\|_2} L_S(w)$

or $RERM(S) = \widehat{w}_\lambda = \arg\min L_S(w) + \lambda\|w\|_2^2$

SGD as an Optimization Algorithm for min

Multiple passes/epochs, can have $T > m$
(can processes examples multiple times)

Explicit complexity control: $\|w\|_2 \leq B$ or $+\lambda\|w\|_2^2$

Generalization from explicit complexity control:

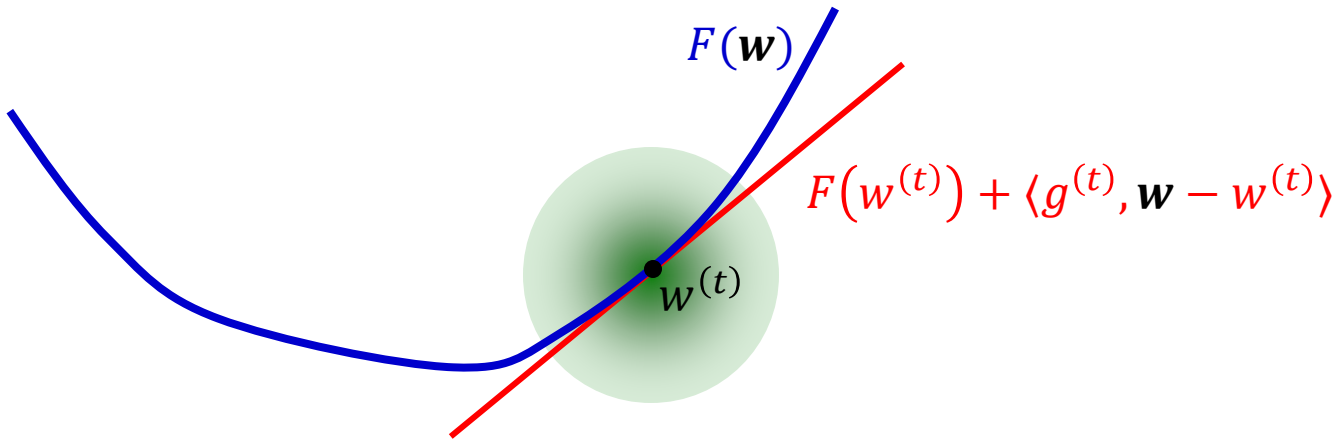
$$L_{\mathcal{D}}(\widehat{w}_B) \leq L_{\mathcal{D}}(w^*) + O\left(\sqrt{\frac{\|w^*\|_2^2 \|\phi\|_2^2}{m}}\right)$$

Explicit inductive bias: $\|w\|_2$

Where's the Regularization

- Gradient Descent seems to be regularizing with $\|w\|_2$. How?

$$w^{(t+1)} \leftarrow \arg\min_w \underbrace{F(w^{(t)}) + \langle g^{(t)}, w - w^{(t)} \rangle}_{\text{1st order model of F(w) around } \mathbf{w}^{(t)}, \text{ based on } \mathbf{g}^{(t)}} + \underbrace{\frac{1}{2\eta} \|w - w^{(t)}\|_2}_{\text{only valid near } w^{(t)}, \text{ so don't go too far. And stochastic, so don't trust it too much}}$$

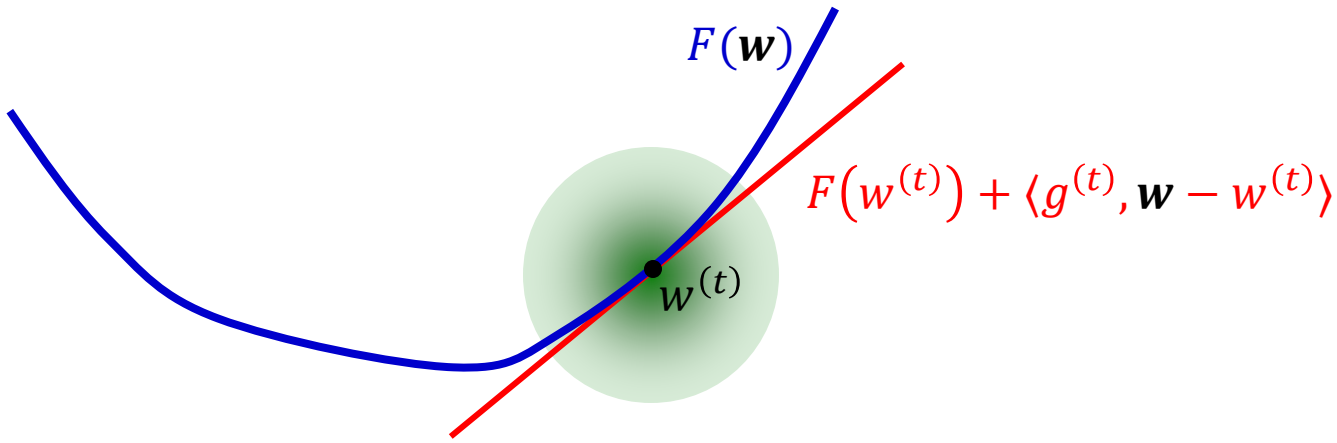


Where's the Regularization

- Gradient Descent seems to be regularizing with $\|w\|_2$. How?

$$w^{(t+1)} \leftarrow \arg\min_w \underbrace{F\left(w^{(t)}\right) + \langle g^{(t)}, w - w^{(t)} \rangle}_{\text{1st order model of F(w) around } \mathbf{w}^{(t)}, \text{ based on } \mathbf{g}^{(t)}} + \frac{1}{2\eta}\left\|w - w^{(t)}\right\|_2$$

$$= w^{(t)} - \eta g^{(t)}$$



- SGD (at least on convex problems) implicitly regularizes using $\|w\|_2$
  - #iterations $T \approx$ sample complexity $m \propto \|w\|_2^2$
  - Generalization/suboptimality controlled in terms of $\|w\|_2$ ➔ this is the inductive bias
  - Alternative to $\|w\|_2 \leq B$ or adding $\lambda\|w\|_2$ for injecting $\|w\|_2$ inductive bias (same guarantee)

- What about other regularizers $R(w)$ / inductive biases??
  - Can apply SGD to regularized or constrained ERM:

$$\min_{R(w)\leq B} L_S(w) \quad \text{or} \quad \min L_S(w) + \lambda R(w)$$

Sample complexity m controlled by $R(w^*)$,

…but #iterations T controlled by $\|w^*\|_2$

  - Other optimization methods related to other regularizers / inductive biases

(generic answer for convex $R(w)$ and convex (ie linear) learning problems: Stochastic Mirror Descent with potential function corresponding to $R(w)$—beyond scope of this course)

- Stochastic Gradient Descent as a Learning Algorithm:
  - One pass over the data!

- What if we do multiple passes over the data?
- Or what about batch gradient descent?

Can Batch Gradient Descent also help generalization (inject inductive bias)?

$\min_w L_S(w)$ using $w^{(t+1)} \leftarrow w^{(t)} - \eta_t \nabla L_S\left(w^{(t)}\right)$

$w^{(t)} \xrightarrow{t\to\infty} \arg\min L_S(w)$, but which minimizer??

- Consider $h_w(x) = \langle w, \phi(x)\rangle, \phi(x) \in \mathbb{R}^D, \quad D \gg m, \quad \ell(h_w(x), y) = |h_w(x) - y|$
- If data in ``general position'': $\exists w\ L_S(w) = 0$, in fact an entire $D - m$ dim space of minimizers!

Claim: starting from $w^{(0)} = 0$, $w^{(t)} \xrightarrow{t\to\infty} \arg\min \|w\|_2$ s.t. $L_S(w) = 0$

Proof:

(1) $w^{(t)} \in span\left(\phi(x_1), \dots, \phi(x_m)\right)$

$$\nabla L_S(w) = \sum \ell'(\dots)\phi(x_i) \in span\left(\phi(x_1), \dots, \phi(x_m)\right)$$

$$w^{(t)} = -\sum \eta_t \nabla L_S\left(w^{(j)}\right) \in span\left(\nabla L_S\left(w^{(j)}\right)\right) \subseteq span\left(\phi(x_1), \dots, \phi(x_m)\right)$$

(2) If $w \in span\left(\phi(x_1), \dots, \phi(x_m)\right)$ and $\langle w, \phi(x_i)\rangle = y_i$, then it's the min norm solution

consider $w + w_\parallel + w_\perp$. Any $w_\perp \neq 0$ would violate constraints, and any $w_\parallel \neq 0$ would increase norm

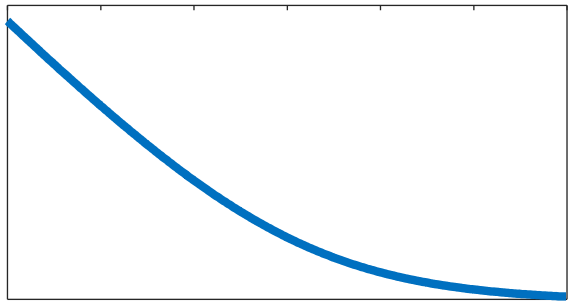
Can Batch Gradient Descent also help generalization (inject inductive bias)?

$\min_w L_S(w)$ using $w^{(t+1)} \leftarrow w^{(t)} - \eta_t \nabla L_S^{lgstc}\left(w^{(t)}\right)$

$w^{(t)} \xrightarrow{t\to\infty} \arg\min L_S(w)$, but which minimizer??

- Consider $h_w(x) = \langle w, \phi(x)\rangle, \phi(x) \in \mathbb{R}^D, \quad D \gg m, \quad \ell^{lgstc}(h_w(x), y) = \log\left(1 + e^{-yh_w(x)}\right)$
- Data linear separable: $\exists w\ \forall_i y_i \langle w, \phi(x_i)\rangle > 0$

$L_S^{lgstc}\left(w^{(t)}\right) \to 0$



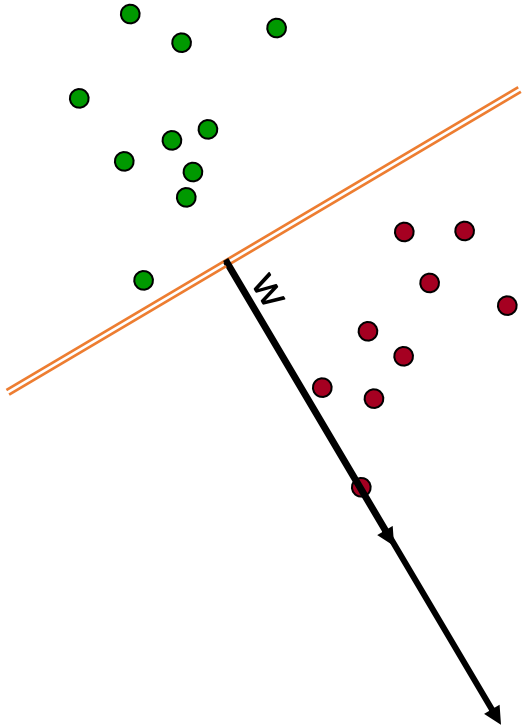
$w^{(t)} \xrightarrow{t\to\infty}$??

$w^{(t)} \to \infty$!

But in what direction?

$sign\left(\left\langle w^{(t)}, \phi(x)\right\rangle\right) \to$??

$\frac{w^{(t)}}{\|w^{(t)}\|} \to$??



Can Batch Gradient Descent also help generalization (inject inductive bias)?

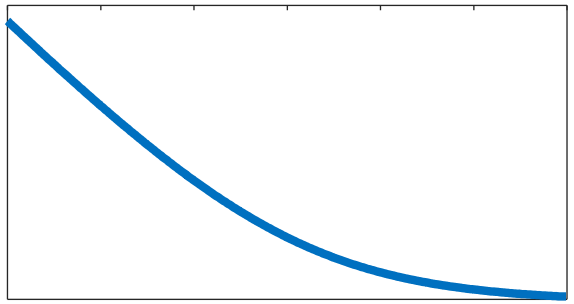
$\min_w L_S(w)$ using $w^{(t+1)} \leftarrow w^{(t)} - \eta_t \nabla L_S^{lgstc}(w^{(t)})$

$w^{(t)} \xrightarrow{t\to\infty} \arg\min L_S(w)$, but which minimizer??

- Consider $h_w(x) = \langle w, \phi(x)\rangle$, $\phi(x) \in \mathbb{R}^D$, $D \gg m$, $\ell^{lgstc}(h_w(x), y) = \log\left(1 + e^{-y h_w(x)}\right)$
- Data linear separable: $\exists w\ \forall_i y_i \langle w, \phi(x_i)\rangle > 0$

$L_S^{lgstc}(w^{(t)}) \to 0$ $\quad w^{(t)} \xrightarrow{t\to\infty}$??

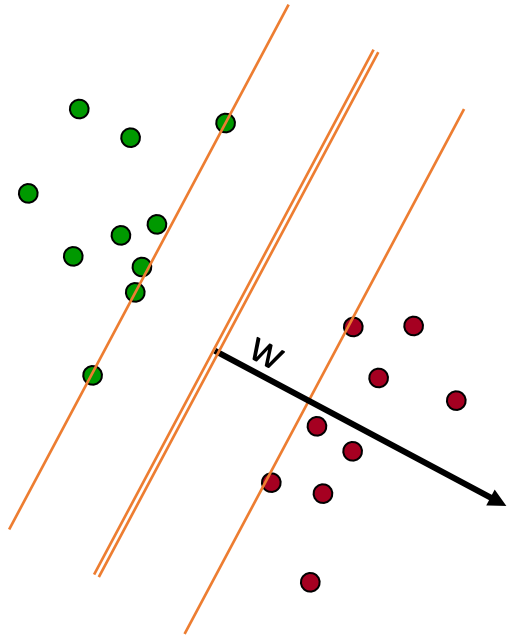
$w^{(t)} \to \infty$!



But in what direction?

$sign(\langle w^{(t)}, \phi(x)\rangle) \to$??

$\frac{w^{(t)}}{\|w^{(t)}\|} \to$??



Claim: $\frac{w(t)}{\|w(t)\|_2} \xrightarrow{t\to\infty} \frac{\hat{w}}{\|\hat{w}\|_2}$ $\quad \hat{w} = \arg\min \|w\|_2 \ s.t. \forall_i y_i \langle w, x_i\rangle \geq 1$

- **Gradient Descent (or Multi-Pass SGD) on $L_S(w)$ converges to $\arg\min \|w\|_2 \; s.t. \; L_s(w) = 0$**

  or $\propto \arg\min \|w\|_2 \; s.t. \; L_s^{\text{margin}}(w) = 0$ (with ℓ^{lgstc})

  (with $\ell^{abs}(h_w(x), y) = |h_w(x) - y|$ or $\ell^{sq}(h_w(x), y) = (h_w(x) - y)^2$)

  $\equiv$ **MDL for $\|w\|_2$**

- Gradient Descent or Multi-Pass SGD with Early Stopping

  provides complexity control related to $\|w\|_2$

  generalization properties similar to RERM, $\arg\min L_S(w) + \lambda\|w\|_2$

  tradeoff controlled by **stepsize *and* stopping time (#iterations)**

- **One-Pass ("Online") Stochastic Gradient Descent**

  Learning with $\|w\|_2$ inductive bias

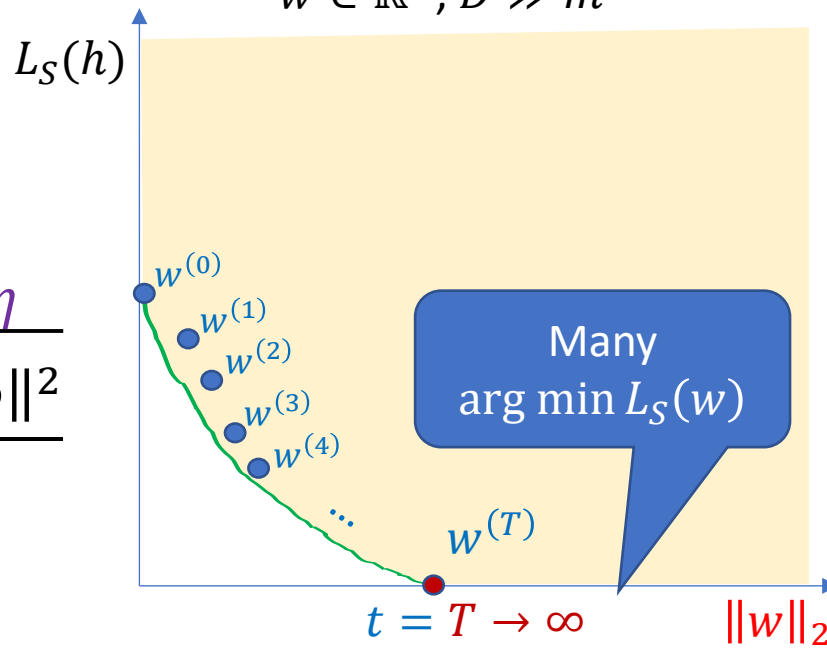
  complexity/fit tradeoff controlled by stepsize ("learning rate") η

$$L(w) \leq \inf_{\|w^*\|_2 \leq \eta\|\phi\| m} L(w^*) + \eta\|\phi\|^2 \leq \inf_{\|w^*\|_2 \leq B} L(w^*) + \sqrt{\frac{B^2\|\phi\|^2}{m}}$$

with $\eta = \frac{B}{\|\phi\|\sqrt{m}}$

$h_w(x) = \langle w, \phi(x) \rangle$

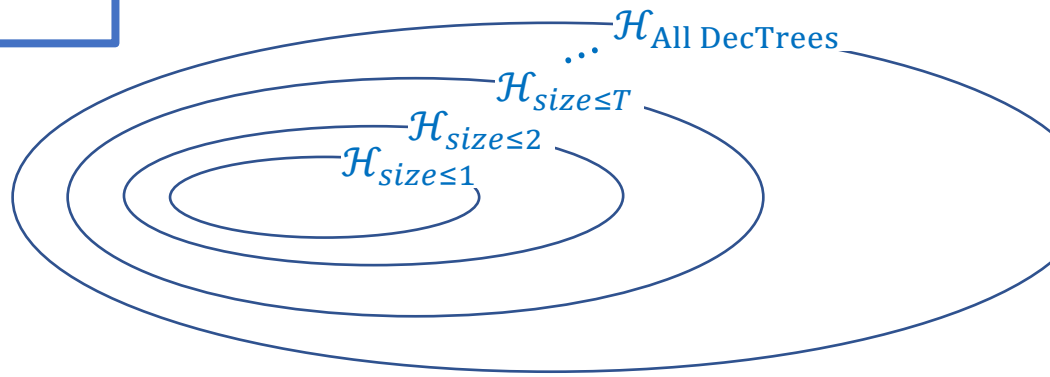
$w \in \mathbb{R}^D, D \gg m$



Greedy Decision Tree Construction, minimizing $L_S(h)$

```
Init empty decision tree h_0
While some nodes in h_t are impure (have ≥ 1 train label):
    Pick node v and feature that maxs train error reduction
    Split v according to predicate to obtain h_{t+1}
```

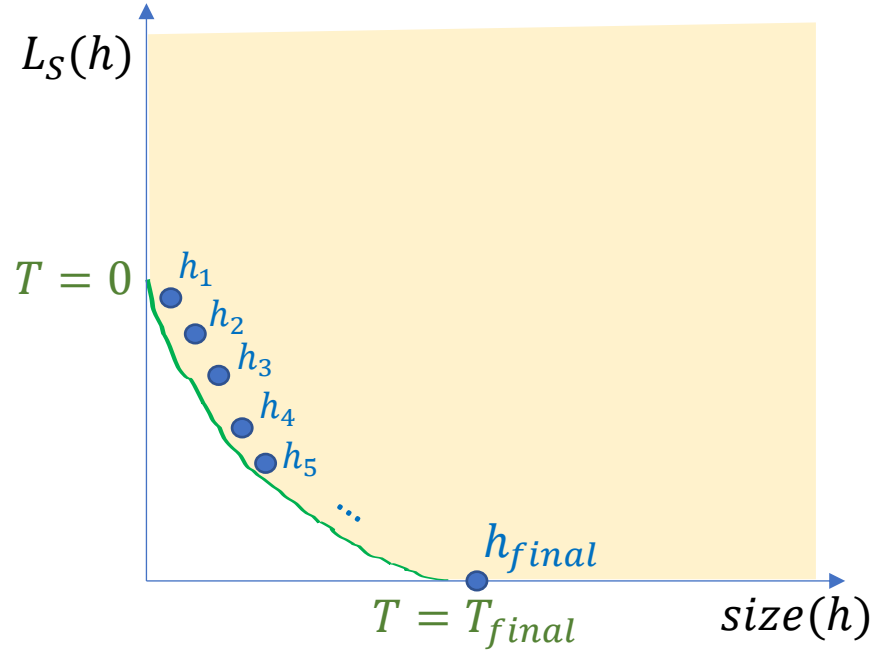
$$h_{final} \approx \arg \min_{L_S(h)=0} size(h_T)$$



- But early stopping after T iterations: $size(h_T) \le T$

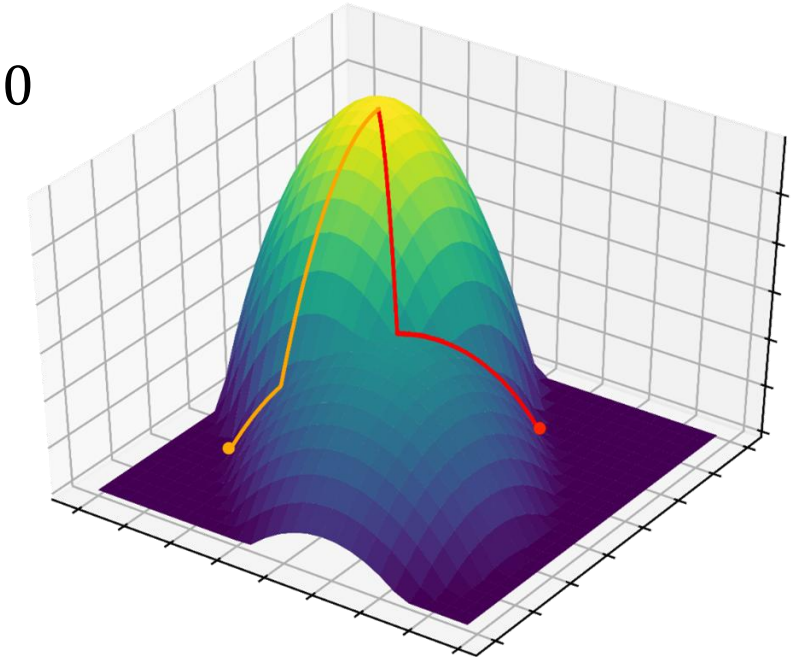
- Early stopping corresponds to controlling the inductive bias "decision tree size"

- How early we step ≡ balance between fit and complexity ≡ where we are on regularization path



Grad Descent to convergence ➔ $\min \|w\|_2 \ \ s.t. L_S(w) = 0$

Early Stopping / online ➔ $\approx \arg\min \ \ \|w\|_2, \ \ \ L_S(w)$



$\|w\|_2$ bias is specific to the optimization method!

Instead, **Coordinate Descent:**

$$i^{(t)} = \arg\max \left|\partial_i L_S\left(w^{(t)}\right)\right|$$

$$w^{(t+1)} = \arg\min L(w) \ \ w = w^{(t)} + \eta e_i$$

Bias towards sparser solutions!

Or $\approx \arg\min \|w\|_1, L_S(w)$

$h_w(x) = \langle w, \phi(x) \rangle$

$w \in \mathbb{R}^D, D \gg m$