Learning rate in traditional optimization

\[ w^{t+1} \leftarrow w^t - \eta \cdot \nabla L(w^t) \]

Traditional: Start with some LR; decay over time.

(Extensive literature in optimization justifying this)

Confusingly, exotic LR schedules also reported to work: triangular [Smith, 2015], cosine [Loshchilov & Hutter, 2016] etc.; no justification in theory.
This work: Exponential LR schedules

Result 1 (empirical): Possible to train today’s deep architectures, while growing LR exponentially (i.e., at each iteration multiply by \((1 + c)\) for some \(c > 0\))

Result 2 (theory): Mathematical proof that exponential LR schedule can yield (in function space*) every net produced by existing training schedules.

Raises questions for theory; highlights importance of trajectory analysis (vs landscape)

(* In all nets that use batch norm [Ioffe-Szegedy’13] or any other normalization scheme.)
Main Thm (original schedule has fixed LR)

\[ \theta_t = \theta_{t-1} - \eta v_t \]
\[ v_t = \gamma v_{t-1} + \nabla_{\theta} \left( L_t(\theta_{t-1}) + \frac{\lambda}{2} \| \theta_{t-1} \|^2 \right) \]

Learning Rate
Momentum
\( \ell_2 \) regularizer (aka Weight decay)
Stochastic Loss at round \( t \)

"General training algorithm; fixed LR"

**Thm:** For nets w/ batch norm or layer norm, following is equivalent to above: weight decay 0, momentum \( \gamma \), and LR schedule \( \eta_t = \eta \alpha^{-2t-1} \) where \( \alpha (\alpha < 1) \) is nonzero root of

\[ x^2 - (1 + \gamma - \lambda \eta)x + \gamma = 0, \]

(proof shows nets in the new trajectory are equivalent in function space to nets in original trajectory)
More general (original schedule has varying LR)

Original schedule (Step Decay): K phases. In phase $I$, iteration $[T_I, T_{I+1} - 1]$, use LR $\eta^*_I$.

Thm: Step Decay can be realized with tapered exponential LR schedule

Tapered Exponential LR schedule (TEXP):
Exponential growing LR in each phase + when entering a new phase:
• Switching to a slower exponential growing rate;
• Dividing the current LR by some constant.

(See details in the paper)
Key concept in proof: Scale-invariant training loss

\[ L(c \cdot \theta) = L(\theta) \text{ for every vector } \theta \text{ of net parameters and } c > 0 \]

Observation: Batch norm + fixed final layer \(\rightarrow\) Loss is scale invariant
(True for Feed-forward nets, Resnets, DenseNets, etc.; see our appendix)

Scale-invariant loss fn sufficient for state of the art deep learning! [Hoffer et al, 18]

Lemma: Gradient for such losses satisfies

1. \( \langle \nabla_\theta L, \theta \rangle = 0 \);
2. \( \nabla_\theta L \big|_{\theta = \theta_0} = c \nabla_\theta L \big|_{\theta = c\theta_0} \), for any \( c > 0 \)
Proof sketch when momentum=0

\( \eta' = c^2 \eta \)

\[ \eta' = \frac{1}{c} \nabla L(\theta_t) \]

\[ \theta'_t = c \theta_t \]

\[ \theta'_t + 1 = \theta'_t - \eta' \nabla L(\theta'_t) \]

\[ (\theta'_t, \eta') \rightarrow (c \theta_t, c^2 \eta) \]

\[ (\theta'_t + 1, \eta') \rightarrow (c \theta_t + 1, c^2 \eta) \]
Warm-up: Equivalence of momentum-free case

\[
(\theta', \eta') \rightarrow (c\theta_t, c^2\eta)
\]

\[
\theta_{t+1}' = \theta_t' - \eta' \nabla L(\theta_t')
\]

\[
(\theta_t', \eta') \rightarrow (c\theta_{t+1}, c^2\eta)
\]

\[
\theta_{t+1} = \theta_{t+1} - \eta \nabla L(\theta_t)
\]

1. Run GD for a step:
\[
GD_t(\theta, \eta) = (\theta - \eta \nabla L_t(\theta), \eta);
\]

2. Scale the parameter \( \theta \):
\[
\Pi_1^c(\theta, \eta) = (c\theta, \eta);
\]

3. Scale the LR \( \eta \):
\[
\Pi_2^c(\theta, \eta) = (\theta, c\eta).
\]
Warm-up: Equivalence of momentum-free case

1. Run GD for a step: \[ GD_t(\theta, \eta) = (\theta - \eta \nabla L_t(\theta), \eta); \]
2. Scale the parameter \( \theta \): \[ \Pi_1^c(\theta, \eta) = (c\theta, \eta); \]
3. Scale the LR \( \eta \): \[ \Pi_2^c(\theta, \eta) = (\theta, c\eta). \]
Run GD with WD for a step: \( \text{GD}_t^\rho(\theta, \eta) = (\rho \theta - \eta \nabla L_t(\theta), \eta); \quad (\rho = 1 - \lambda \eta) \)

**Lemma 2.2.** \( \text{GD}_t^\rho = \Pi_2^\rho \circ \Pi_1^\rho \circ \text{GD}_t \circ \Pi_2^{-1} \).
Run GD with WD for a step: \[ GD_t^\rho(\theta, \eta) = (\rho \theta - \eta \nabla L_t(\theta), \eta); \quad (\rho = 1 - \lambda \eta) \]

\[ GD_t^\rho = \Pi_2^\rho \circ \Pi_1^\rho \circ GD_t \circ \Pi_2^{\rho^{-1}} = \]

\[ \Pi_2^{\rho^{-1}} \quad GD_t \quad \Pi_2^\rho \circ \Pi_1^\rho \]

**Theorem:** GD + WD + constant LR = GD + Exp LR.

\[ \Pi_1^{\rho^{-t}} \circ \Pi_2^{\rho^{-2t}} \circ GD_{t-1}^\rho \circ \cdots \circ GD_0^\rho = \Pi_2^{\rho^{-1}} \circ GD_{t-1} \circ \Pi_2^{\rho^{-2}} \circ \cdots \circ GD_1 \circ \Pi_2^{\rho^{-2}} \circ GD_0 \circ \Pi_2^{\rho^{-1}} \]

**Proof:**

\[ \theta_0 = \theta_0' \]

\[ \Pi_2^{\rho^{-1}} \circ \Pi_1^{\rho^{-2}} \]

\[ \theta' \]

\[ \theta' \]

\[ \theta' \]
Conclusions

• Scale-Invariance (provided by BN) makes the training procedure incredibly robust to LR schedules, even to exponentially growing schedules.

• Space of good LR schedules in current architectures is vast (hopeless to search for best schedule??)

• Current ways of thinking about training/optimization should be rethought;
  • should focus on trajectory, not landscape.