Recitation - 8, Part 2
10605/10805

October 28, 2022

Slides credits: Justin Johnson, Sebastian Ruder, Yuanzhi Li, Qinxin Wang

Plan

- Stochastic Gradient Descent
- Learning Rate Tuning
- Different learning rate decays
  - Step
  - Linear
  - Cosine
- Underfitting/Overfitting
- Tensorflow example

Stochastic Gradient Descent Recap

Stochastic Gradient Descent

- Update parameter based on gradient information from a single samples in the training set

for i in range(n):

$$w_{t+1} = w_t - \alpha * \frac{\partial F_i}{\partial w_t}$$

Gradient Descent

- Update parameter based on gradient information from all samples in the training set

$$w_{t+1} = w_t - \alpha * \frac{\partial F}{\partial w_t}$$

Stochastic Gradient Descent Recap

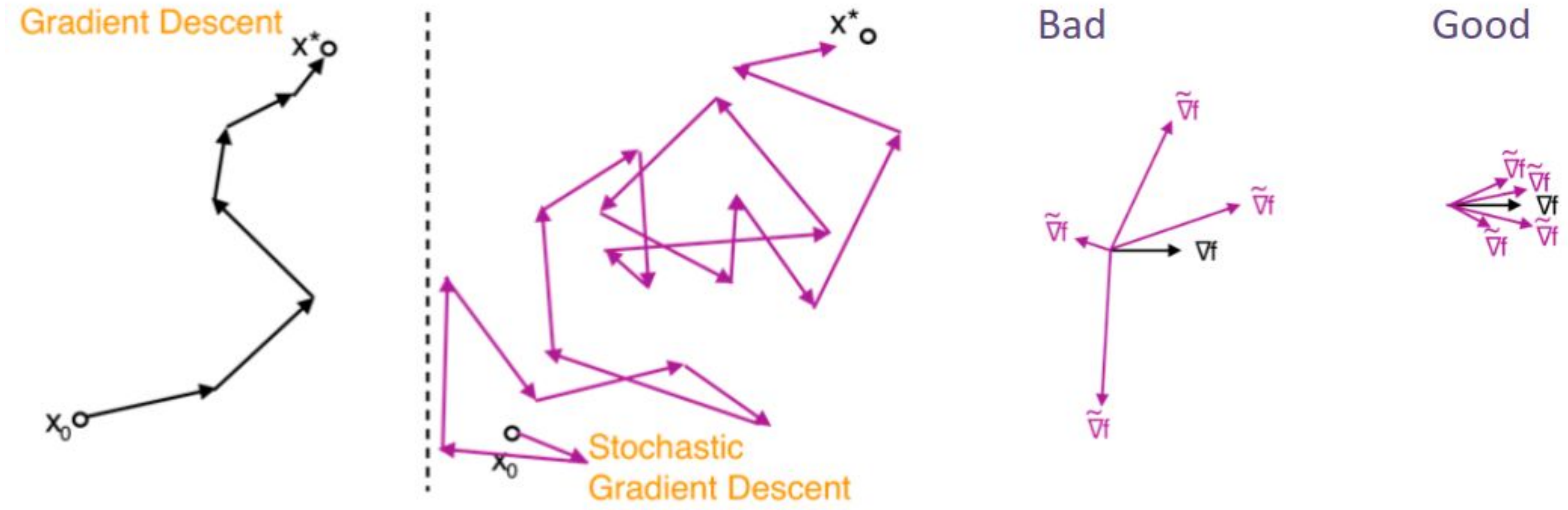
Stochastic Gradient Descent

- Computationally cheap for one step
- Takes more steps to converge
- High variance

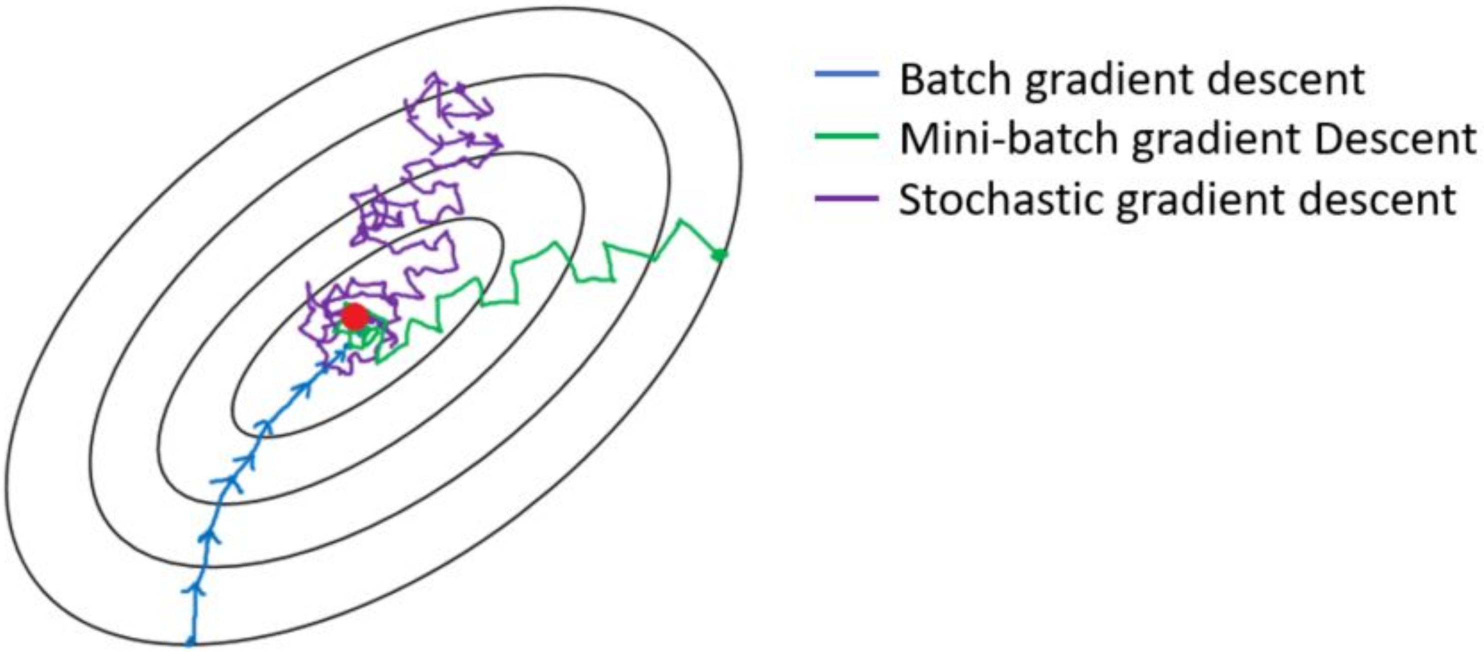
Gradient Descent

- Computationally expensive for one step
- Takes lesser steps to converge
- Low variance

Stochastic Gradient Descent Recap



$$E\left[\left|\left|\nabla F(w_j)\right|\right|_2^2\right] \textit{ is known as the variance}$$



Taking step decision based on single sample (SDG) vs based on entire training data (GD)

SGD with momentum

- Idea: Use the concept of exponentially weighted averages (concept to reduce noise and smoothen time series data) to reduce the oscillatory behaviour of SGD and make convergence faster

$$w_t = w_{t-1} - \eta \frac{\partial L}{\partial w_{t-1}}$$

$$b_t = b_{t-1} - \eta \frac{\partial L}{\partial b_{t-1}}$$

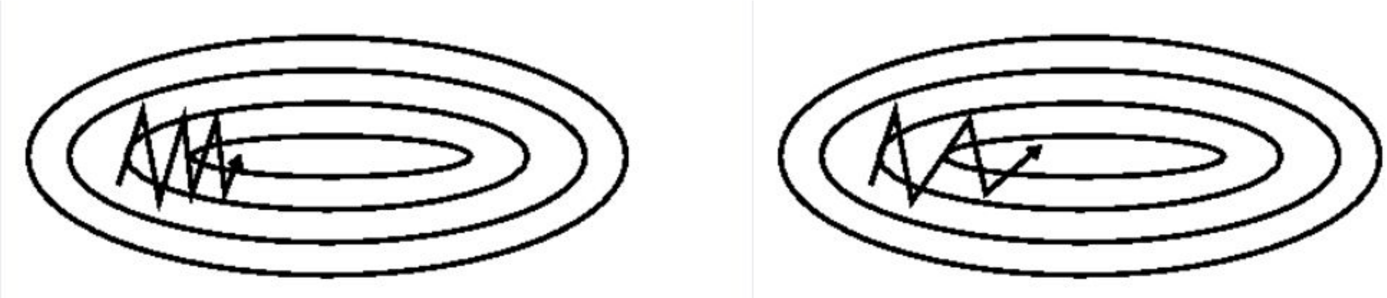
$\longrightarrow$

$$w_t = w_{t-1} - \eta V_{dw_t}$$

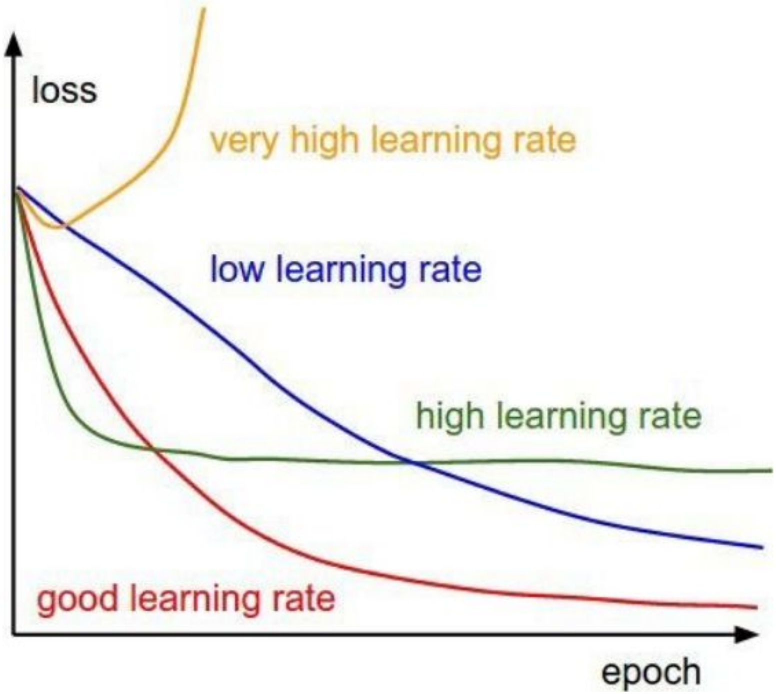
$$\text{where } V_{dw_t} = \beta V_{dw_{t-1}} + (1-\beta)\frac{\partial L}{\partial w_{t-1}}$$

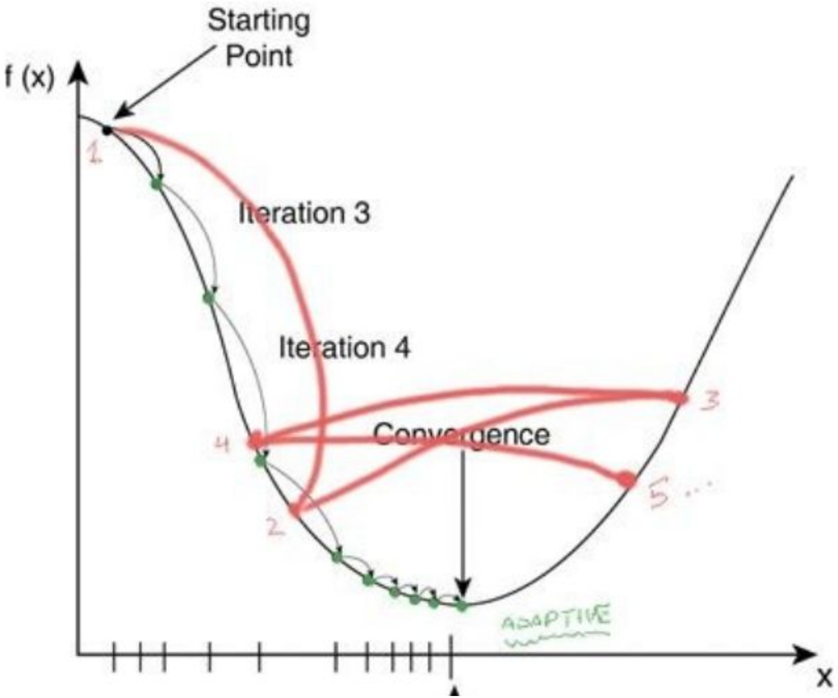
$$b_t = b_{t-1} - \eta V_{db_t}$$

$$\text{where } V_{db_t} = \beta V_{db_{t-1}} + (1-\beta)\frac{\partial L}{\partial b_{t-1}}$$



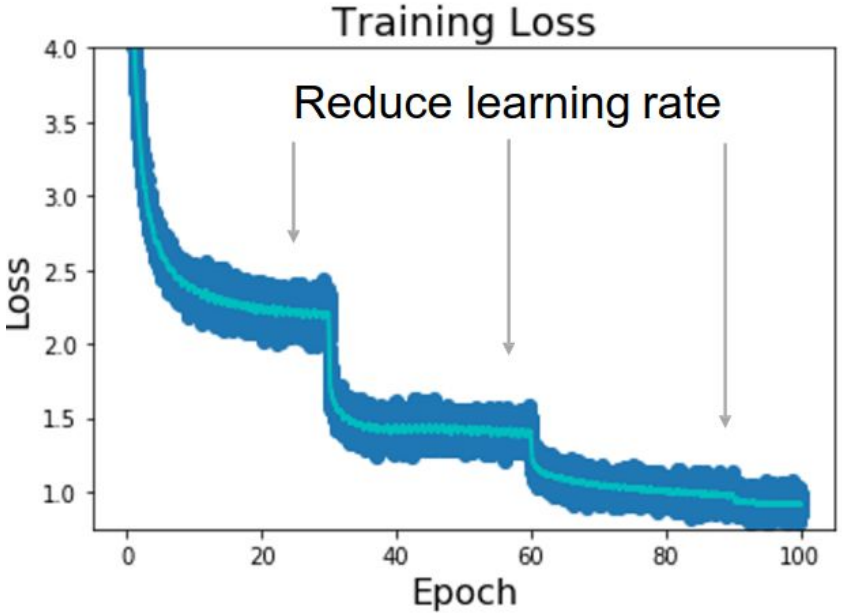
Learning Rate tuning



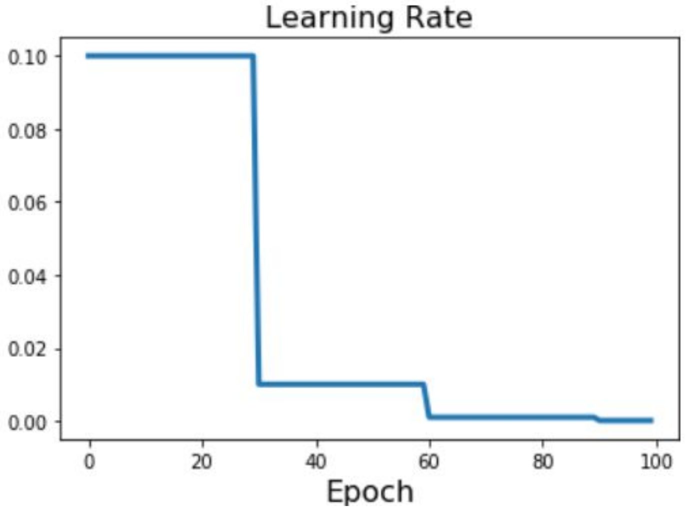


Good idea to change the lr over time, as the training progresses

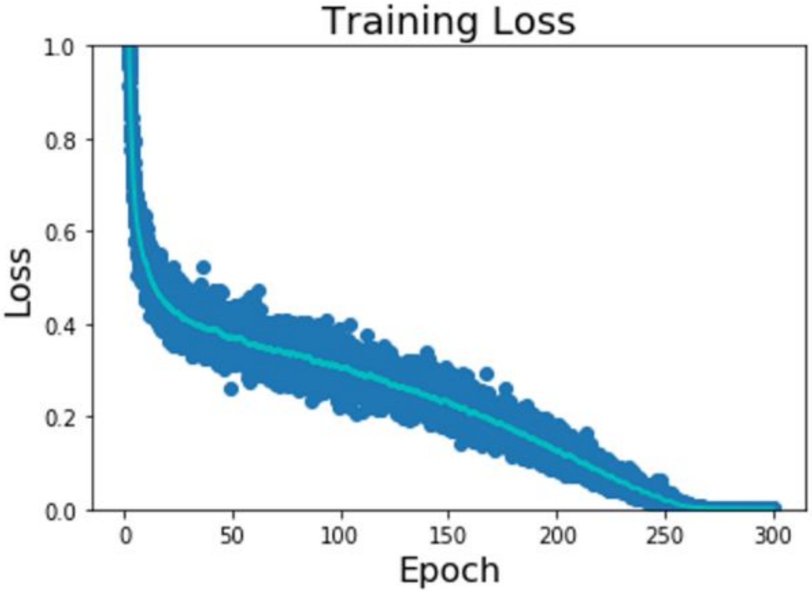
Learning Rate Decay: Step



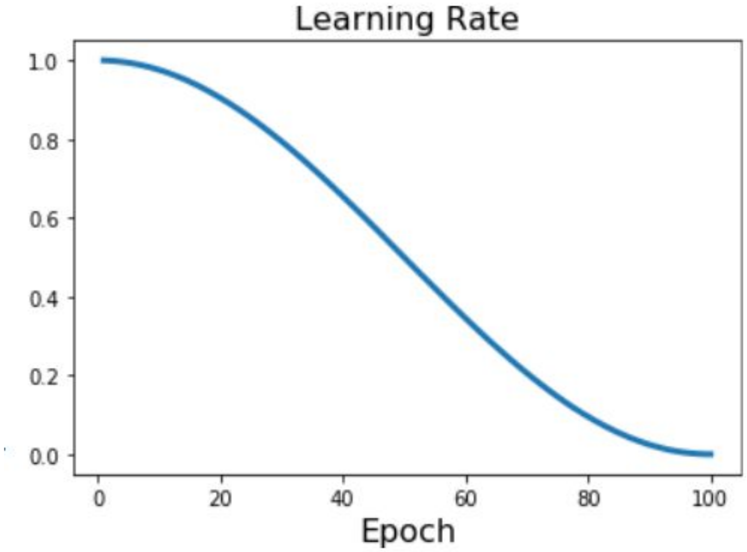
Step: Reduce learning rate at a few fixed points. E.g. for ResNets, multiply LR by 0.1 after epochs 30, 60, and 90.



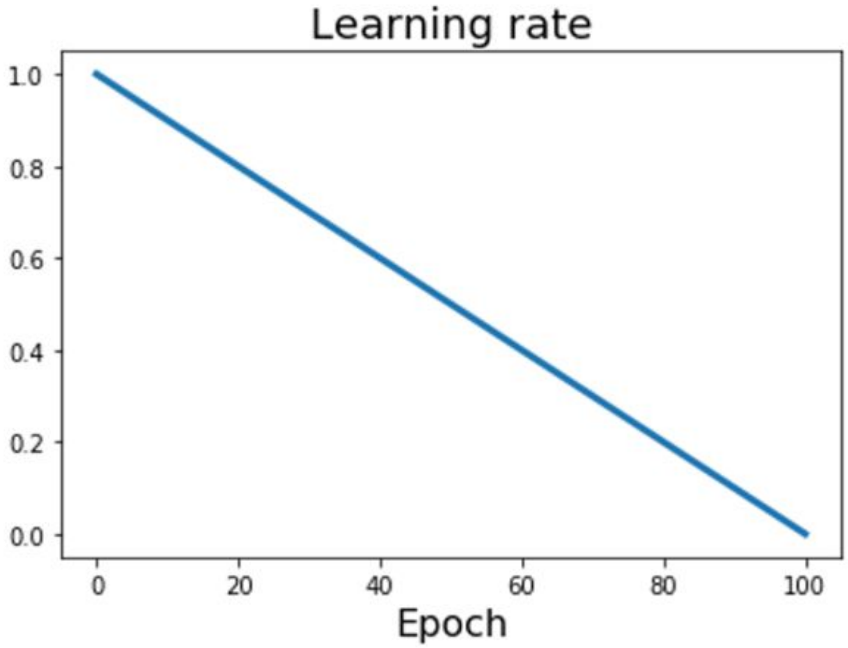
Learning Rate Decay: Cosine



Cosine: $\alpha_t = \frac{1}{2}\alpha_0\left(1 + \cos\left(\frac{t\pi}{T}\right)\right)$

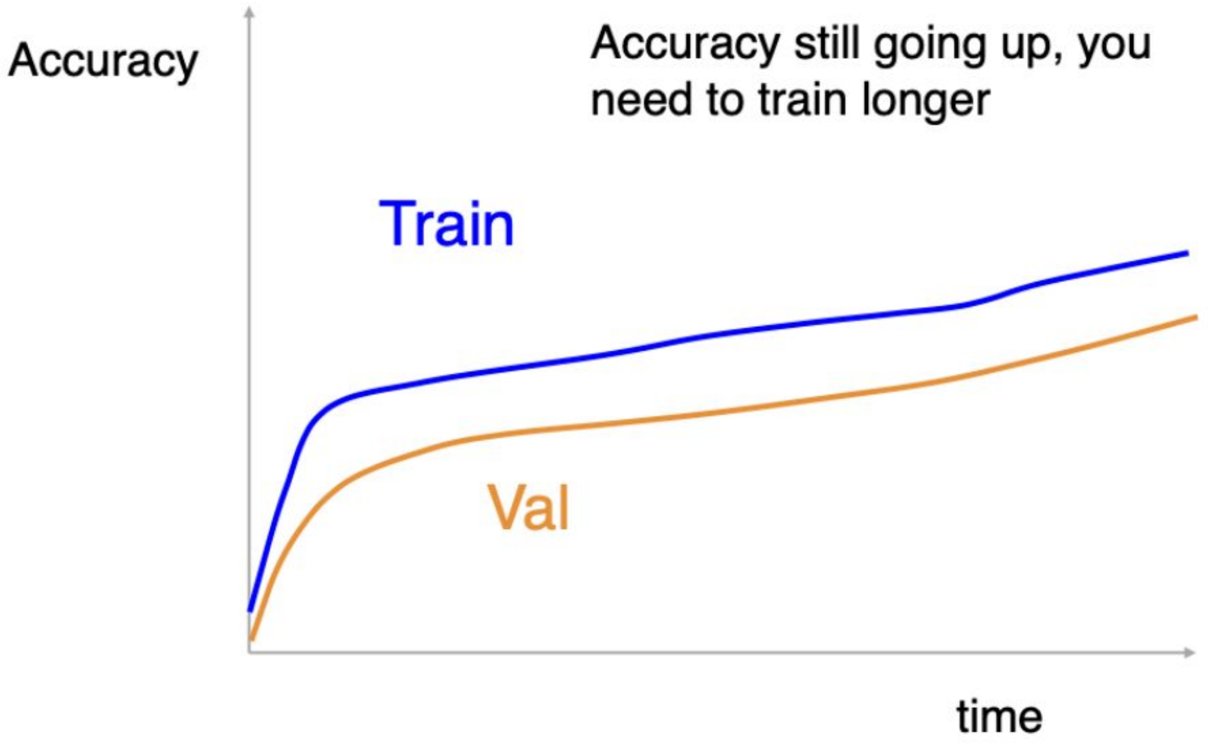


Learning Rate Decay: Linear

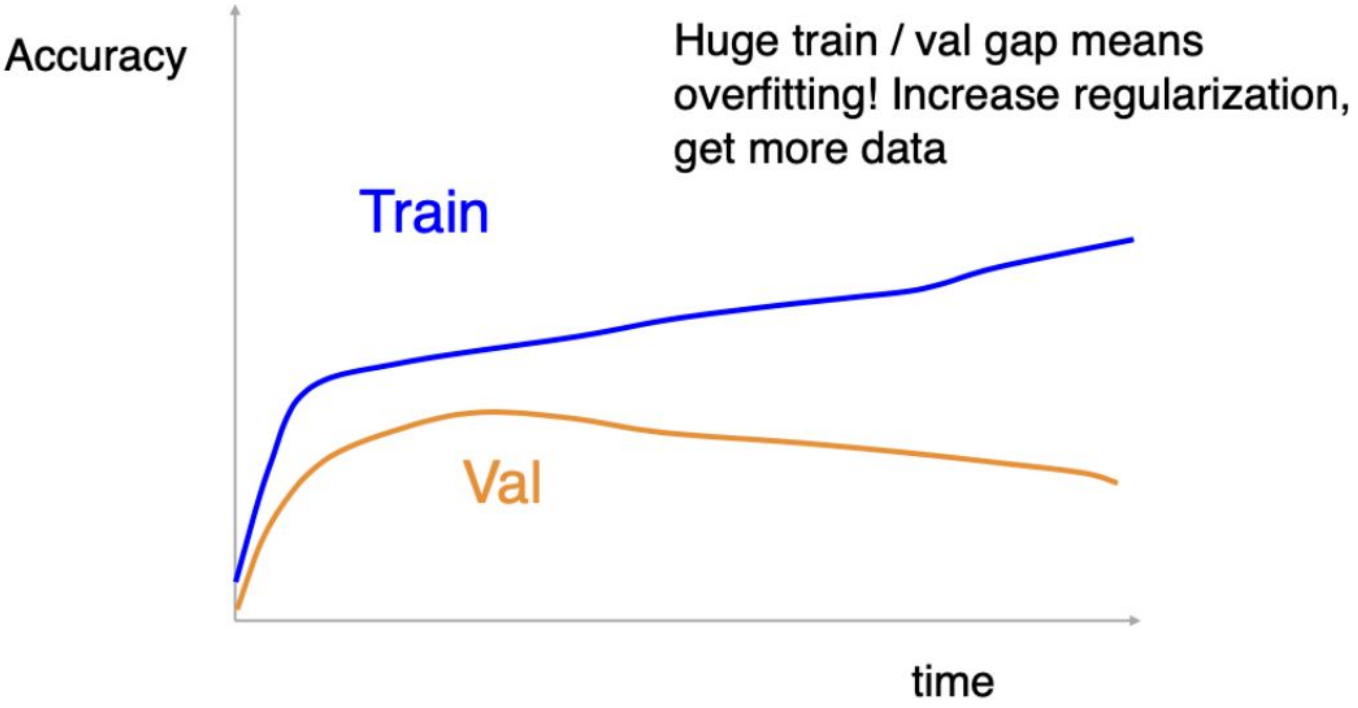


Linear: $\alpha_t = \alpha_0 \left(1 - \frac{t}{T}\right)$

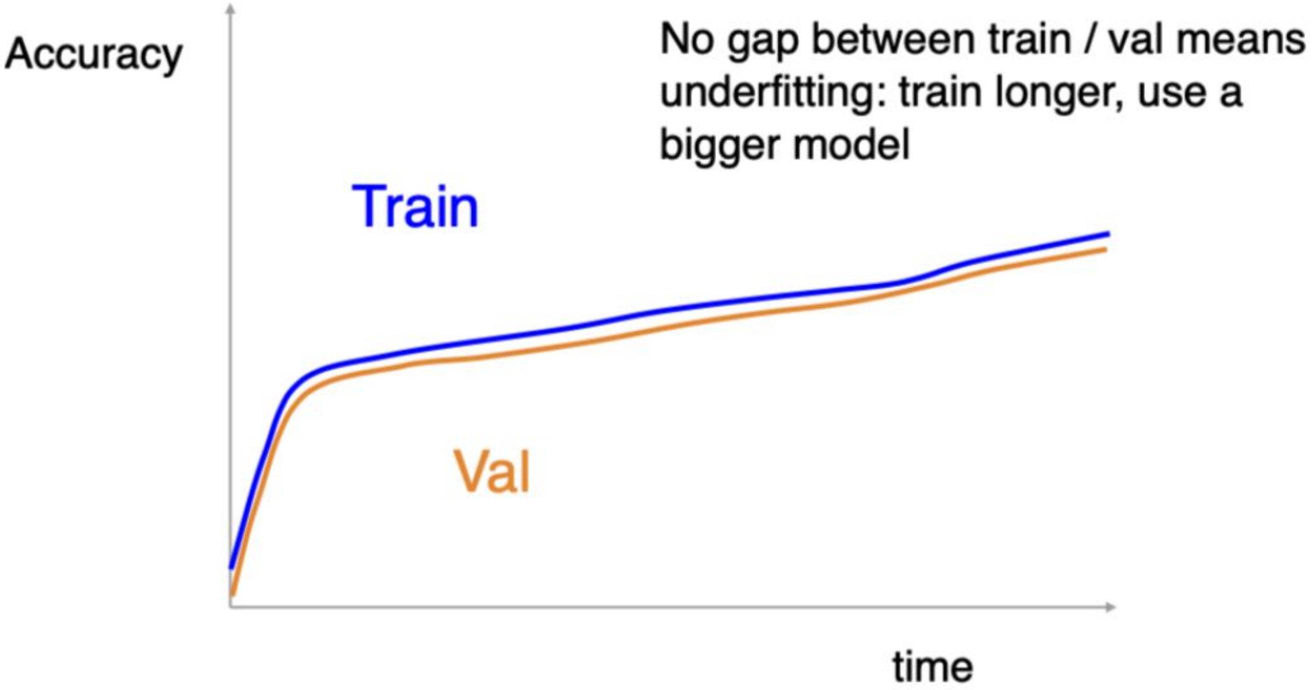
Model Learning Situations - 1



Model Learning Situations - 2



Model Learning Situations - 3



Choosing Hyperparameters and debugging NN models

1. Check initial loss
2. Overfit on a small sample of the data
3. Experiment with different learning rates, check that the training loss decreases over time. If you are using Adam optimizer, can start with lr of 1e-3 or 1e-4.
4. After you have performed the sanity check, you can now use the entire training data with the same model, choose a small weight decay and start training.