Other optimization tips and tricks

- **Momentum:** use exponentially weighted sum of previous gradients

$$\overline{\nabla}_{\boldsymbol{\theta}}^{(t)} = \nabla_{\boldsymbol{\theta}} l(\mathbf{f}(\mathbf{x}^{(t)}), y^{(t)}) + \beta \overline{\nabla}_{\boldsymbol{\theta}}^{(t-1)}$$

can get pass plateaus more quickly, by "gaining momentum"

- **Initialization:** cannot initialize to same value, all units in a hidden layer will behave same; randomly initialize unif[-b,b]

- **Adaptive learning rates:** one learning rate per parameter

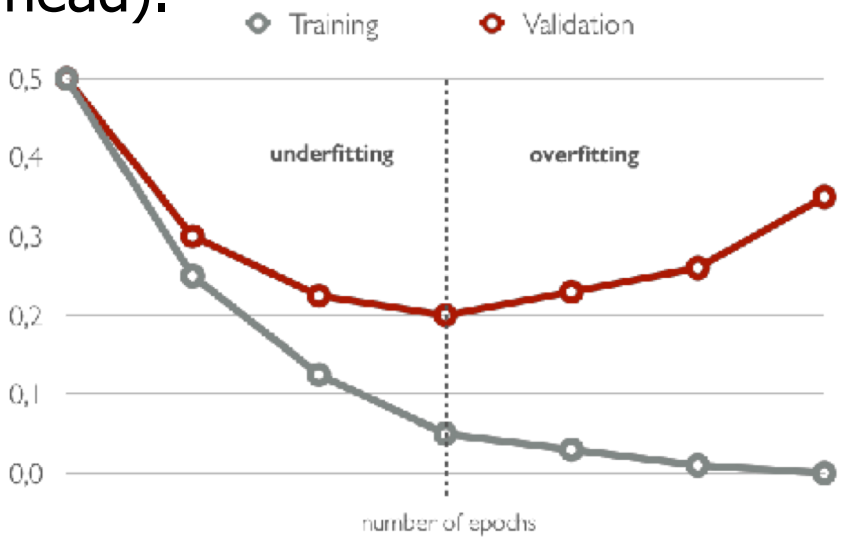
e.g. RMSProp uses exponentially weighted average of squared gradients

$$\gamma^{(t)} = \beta \gamma^{(t-1)} + (1-\beta) \left(\nabla_\theta l(\mathbf{f}(\mathbf{x}^{(t)}), y^{(t)}) \right)^2 \qquad \overline{\nabla}_\theta^{(t)} = \frac{\nabla_\theta l(\mathbf{f}(\mathbf{x}^{(t)}), y^{(t)})}{\sqrt{\gamma^{(t)} + \epsilon}}$$

Adam combines RMSProp with momentum

Tips and tricks for preventing overfitting

- **Dropout**
- **Data augmentation**
- **Early stopping:** stop training when validation set error increases (with some look ahead).



- **Neural Architecture search:** tune number of layers and neurons per layer using grid search or clever optimization

Tips and Tricks for training deep NNs

- First hypothesis (underfitting): better optimize
  - ➢ Increase the capacity of the neural network
  - ➢ Check initialization
  - ➢ Check gradients (saturating units and vanishing gradients)
  - ➢ Tune learning rate
- Second hypothesis (overfitting): use better regularization
  - ➢ Dropout
  - ➢ Data augmentation
  - ➢ Early stopping
  - ➢ Architecture search
- For many large-scale practical problems, you will need to use both: better optimization and better regularization!