Neural Networks and Deep Learning: Training Issues

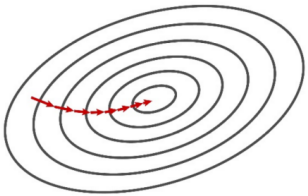
Nicolas Thome

Conservatoire National des Arts et Métiers (Cnam)
Département Informatique

Neural Network Training: Optimization Issues

- Classification loss over training set (vectorized $\mathbf{w}$, $\mathbf{b}$ ignored):

$$\mathcal{L}_{CE}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} \ell_{CE}(\hat{\mathbf{y}}_i, \mathbf{y}_i^*) = -\frac{1}{N}\sum_{i=1}^{N} log(\hat{y}_{c^*,i})$$



- Gradient descent optimization:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \frac{\partial \mathcal{L}_{CE}}{\partial \mathbf{w}}\left(\mathbf{w}^{(t)}\right) = \mathbf{w}^{(t)} - \eta \nabla_{\mathbf{w}}^{(t)}$$

- Gradient $\nabla_{\mathbf{w}}^{(t)} = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial \ell_{CE}(\hat{\mathbf{y}}_i, \mathbf{y}_i^*)}{\partial \mathbf{w}}\left(\mathbf{w}^{(t)}\right)$ linearly scales wrt:
  - $\mathbf{w}$ dimension
  - Training set size

$$\Rightarrow \textbf{Too slow even for moderate dimensionality \& dataset size!}$$

Stochastic Gradient Descent

- **Solution:** approximate $\nabla_{\mathbf{w}}^{(t)} = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial \ell_{CE}(\hat{\mathbf{y}}_i, \mathbf{y}_i^*)}{\partial \mathbf{w}} \left(\mathbf{w}^{(t)}\right)$ with subset of examples

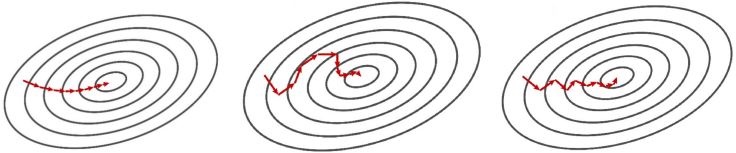
  $\Rightarrow$ **Stochastic Gradient Descent (SGD)**

  - Use a single example (online):

$$\nabla_{\mathbf{w}}^{(t)} \approx \frac{\partial \ell_{CE}(\hat{\mathbf{y}}_i, \mathbf{y}_i^*)}{\partial \mathbf{w}} \left(\mathbf{w}^{(t)}\right)$$

  - Mini-batch: use $B < N$ examples:

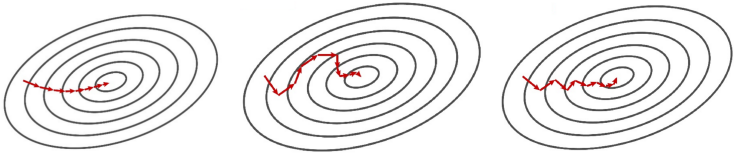
$$\nabla_{\mathbf{w}}^{(t)} \approx \frac{1}{B} \sum_{i=1}^{B} \frac{\partial \ell_{CE}(\hat{\mathbf{y}}_i, \mathbf{y}_i^*)}{\partial \mathbf{w}} \left(\mathbf{w}^{(t)}\right)$$



Full gradient SGD (online) SGD (mini-batch)

Stochastic Gradient Descent

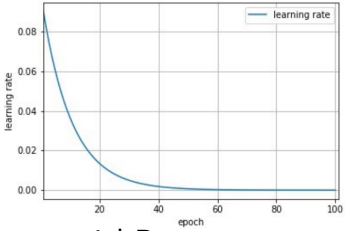
- **SGD: approximation of the true Gradient $\nabla_{\mathbf{w}}$!**
  - Noisy gradient can lead to bad direction, increase loss
  - **BUT:** much more parameter updates: online $\times N$, mini-batch $\times \frac{N}{B}$
  - **Faster convergence**, at the core of Deep Learning for large scale datasets



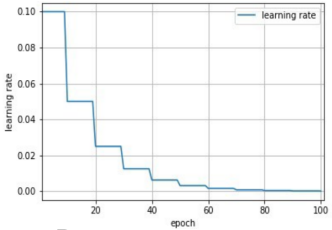
Full gradient SGD (online) SGD (mini-batch)

Optimization: Learning Rate Decay

- Gradient descent optimization: $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla_{\mathbf{w}}^{(t)}$
- η setup ? $\Rightarrow$ open question
- Learning Rate Decay: decrease η during training progress
  - Inverse (time-based) decay: $\eta_t = \frac{\eta_0}{1+r \cdot t}$, r decay rate
  - Exponential decay: $\eta_t = \eta_0 \cdot e^{-\lambda t}$
  - Step Decay $\eta_t = \eta_0 \cdot r^{\frac{t}{t_u}}$ ...



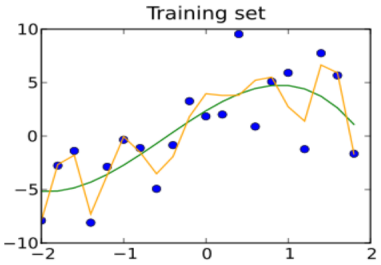
Exponential Decay ($\eta_0 = 0.1$, $\lambda = 0.1$s)

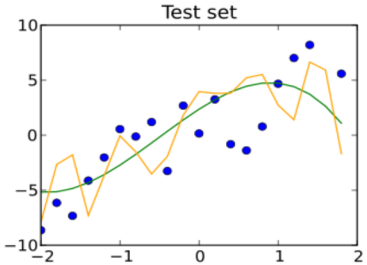


Step Decay ($\eta_0 = 0.1$, $r = 0.5$, $t_u = 10$)

Generalization and Overfitting

- **Learning:** minimizing classification loss $\mathcal{L}_{CE}$ over training set
  - Training set: sample representing data *vs* labels distributions
  - **Ultimate goal:** train a prediction function with low prediction error on the **true (unknown) data distribution**





$\mathcal{L}_{train} = 4$, $\mathcal{L}_{train} = 9$ $\mathcal{L}_{test} = 15$, $\mathcal{L}_{test} = 13$

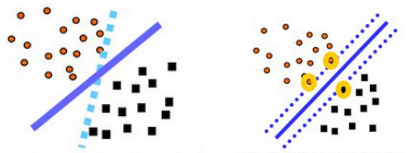
$\Rightarrow$ **Optimization $\neq$ Machine Learning!**
$\Rightarrow$ **Generalization / Overfitting!**

Regularization

- **Regularization:** improving generalization, *i.e.* test ($\neq$ *train*) performances
- Structural regularization: add ***Prior*** $R(\mathbf{w})$ in training objective:

$$\mathcal{L}(\mathbf{w}) = \mathcal{L}_{CE}(\mathbf{w}) + \alpha R(\mathbf{w})$$

- L^2 **regularization: weight decay**, $R(\mathbf{w}) = ||\mathbf{w}||^2$
  - Commonly used in neural networks
  - Theoretical justifications, generalization bounds (SVM)
- Other possible $R(\mathbf{w})$: L^1 regularization, dropout, *etc*

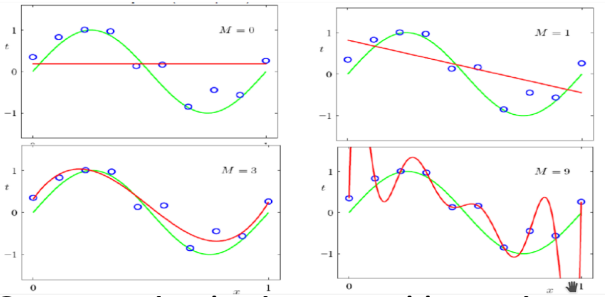


L^2 **regularization**: interpretation

- "Smooth" interpretation of L^2 **regularization**, Cauchy-Schwarz:

$$\langle \mathbf{w}, (\mathbf{x} - \mathbf{x}') \rangle \leq ||\mathbf{w}||^2 ||\mathbf{x} - \mathbf{x}'||^2$$

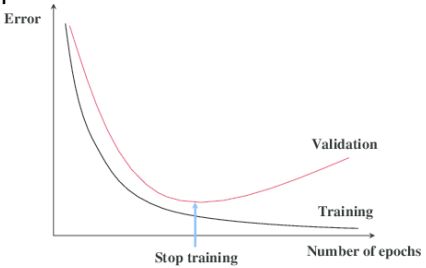
- Controlling L^2 norm $||\mathbf{w}||^2$: "small" variation between inputs $\mathbf{x}$ and $\mathbf{x}'$
  $\Rightarrow$ small variation in neuron prediction $\langle \mathbf{w}, \mathbf{x} \rangle$ and $\langle \mathbf{w}, \mathbf{x}' \rangle$



$\Rightarrow$ **Supports simple, *i.e.* smoothly varying prediction models**

Regularization and hyper-parameters

- **Neural networks:** hyper-parameters to tune:
  - **Training parameters:** learning rate, weight decay, learning rate decay, # epochs, *etc*
  - **Architectural parameters:** number of layers, number neurones, non-linearity type, *etc*
- **Hyper-parameters tuning:** $\Rightarrow$ improve generalization: estimate performances on a validation set



Neural networks: Conclusion

- Training issues at several levels: optimization, generalization, cross-validation
- **Limits of fully connected layers and Convolutional Neural Nets ? ⇒ following!**