TTIC 31230, Fundamentals of Deep Learning

David McAllester, Autumn 2020

Regularization: Early Stopping and Shrinkage

Fitting Finite Training Data

We take a training sample **Train** with N elements.

$$\begin{aligned}\Phi^*(\mathbf{Train}) &= \operatorname*{argmin}_{\Phi} E_{(x,y)\sim\mathbf{Train}}\, \mathcal{L}_\Phi(x,y) \\ &= \operatorname*{argmin}_{\Phi} \frac{1}{N} \sum_{(x,y)\in\mathbf{Train}} \mathcal{L}(x,y)\end{aligned}$$

The training loss is typically less than the test loss.

$$E_{(x,y)\sim\mathbf{Train}}\, \mathcal{L}_{\Phi^*(\mathbf{Train})}(x,y) < E_{(x,y)\sim\mathrm{Pop}}\, \mathcal{L}_{\Phi^*(\mathbf{Train})}(x,y)$$

Fitting Finite Training Data

An nth order polynomial can fit any n (pure noise) data points.

Cross Entropy Loss Vs. Task Loss

While SGD is generally done on cross entropy loss, one often wants minimum error rate or some other leader board measure (task loss).

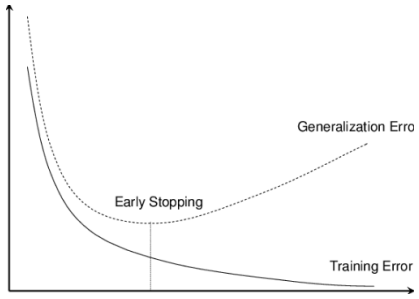
The term "loss" often refers to cross entropy loss as opposed to task loss (leader-board metric)

SGD typically optimizes cross-entropy loss because task loss is typically not differentiable.

Some systems attempt to directly optimize task loss.

But training on (cross entropy) loss is generally effective for minimizing task loss.

Early Stopping



During SGD one tracks validation loss and validation error.

One stops training when the validation error stops improving.

Empirically, loss reaches a minimum sooner than error.

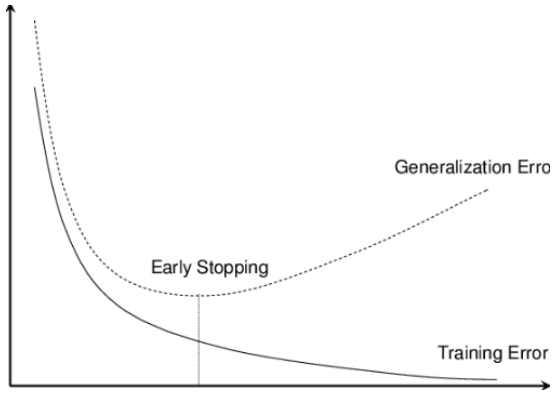
Training Data, Validation Data and Test Data

In general one designs algorithms and tunes hyper-parameters by training on training data and evaluating on validation data.

But it is possible to over-fit the validation data (validation loss becomes smaller than test loss).

Kaggle withholds test data until the final contest evaluation.

Over Confidence



Validation error is larger than training error when we stop.

The model probabilities are tuned on training data statistics.

The probabilities are tuned to an unrealistically low error rate and are therefore over-confident.

This over-confidence occurs before the stopping point and damages validation loss (as opposed to validation error).

Regularization

There is never harm in doing early stopping — one should always do early stopping.

Regularization is a modification to the training algorithm designed to reduce the training-validation gap and, in this way, improve overall performance.

Shrinkage: L_2 regularization

We first give a Bayesian derivation. We put a prior probability on Φ and maximize the a-posteriori probability (MAP).

$$\begin{aligned}
\Phi^* &= \operatorname*{argmax}_{\Phi} \; p(\Phi | \langle x_1, y_1 \rangle, \ldots, \langle x_n, y_n \rangle) \\
&= \operatorname*{argmax}_{\Phi} \; \frac{p(\Phi, \; \langle x_1, y_1 \rangle, \ldots, \langle x_n, y_n \rangle)}{p(\langle x_1, y_1 \rangle, \ldots, \langle x_n, y_n \rangle)} \\
&= \operatorname*{argmax}_{\Phi} \; p(\Phi, \; \langle x_1, y_1 \rangle, \ldots, \langle x_n, y_n \rangle)
\end{aligned}$$

Shrinkage: L_2 regularization

$$
\begin{aligned}
\Phi^* &= \operatorname*{argmax}_{\Phi}\ p(\Phi,\ \langle x_1, y_1\rangle, \ldots, \langle x_n, y_n\rangle) \\
&= \operatorname*{argmax}_{\Phi}\ p(\Phi)\ \prod_i \mathrm{Pop}(x_i) P_\Phi(y_i|x_i) \\
&= \left(\prod_i p(x_i)\right) \operatorname*{argmax}_{\Phi}\ p(\Phi)\ \prod_i P_\Phi(y_i|x_i) \\
&= \operatorname*{argmax}_{\Phi}\ p(\Phi)\ \prod_i P_\Phi(y_i|x_i)
\end{aligned}
$$

Shrinkage: L_2 Regularization

$$\begin{aligned}\Phi^* &= \operatorname*{argmax}_{\Phi} \; p(\Phi) \prod_i P_\Phi(y_i|x_i) \\ &= \operatorname*{argmin}_{\Phi} \sum_i -\ln P_\Phi(y_i|x_i) - \ln p(\Phi)\end{aligned}$$

We now take a Gaussian prior

$$p(\Phi) \propto \exp\left(-\frac{||\Phi||^2}{2\sigma^2}\right)$$

Shrinkage: L_2 Regularization

$$\Phi^* = \underset{\Phi}{\operatorname{argmin}} \sum_{i=1}^{n} -\ln P_\Phi(y_i|x_i) + \frac{||\Phi||^2}{2\sigma^2}$$

$$= \underset{\Phi}{\operatorname{argmin}} \frac{1}{N} \left(\sum_{i=1}^{n} -\ln P_\Phi(y_i|x_i) + \frac{||\Phi||^2}{2\sigma^2} \right)$$

$$= \underset{\Phi}{\operatorname{argmin}} \left(E_{\langle x,\, y\rangle \sim \text{Train}} \; -\ln P_\Phi(y|x) \right) + \frac{1}{2N\sigma^2}||\Phi||^2$$

Shrinkage: L_2 Regularization

$$\nabla_\Phi \, E_{(x,y)\sim\text{Train}} \left(\mathcal{L}(\Phi, x, y) + \frac{||\Phi||^2}{2N\sigma^2} \right)$$

$$= E_{(x,y)\sim\text{Train}} \left(g(\Phi, x, y) + \frac{\Phi}{N\sigma^2} \right)$$

$$\Phi_{i+1} = \Phi_i - \eta \left(\hat{g}_i - \frac{1}{N\sigma^2}\Phi_i \right)$$

The last term in the update equation is called "shrinkage".

Robust Shrinkage

The PyTorch parameters are the learning rate η and the **weight decay** γ:

$$\Phi_{i+1} = \Phi_i - \eta\left(\hat{g} + \frac{1}{N\sigma^2}\Phi_i\right) \qquad = \Phi_i - \eta(\hat{g} - \gamma\Phi_i)$$

To make SGD with shrinkage robust to changes in training size, batch size, learning rate (temperature) and momentum we can use

$$\eta = (1-\mu)B\eta_0 \qquad \gamma = \frac{1}{N_{\text{Train}}\sigma^2}$$

where η_0 is the robust temperature parameter and σ^2 is the robust shrinkage parameter.

END