Training Single Layer Neural Networks

Numerical Methods for Deep Learning

Overview

Given data $\mathbf{Y} \in \mathbb{R}^{n_f \times n}$ and class probabilities $\mathbf{C} \in \mathbb{R}^{n_c \times n}$ we aim to find the transformation parameters $\theta \in \mathbb{R}^p$ and classification weights (including potentially biases) $\mathbf{W} \in \mathbb{R}^{n_c \times m}$ by solving

$$\min_{\theta, \mathbf{W}} E(\mathbf{W}\sigma(\mathbf{K}(\theta)\mathbf{Y}), \mathbf{C}) + \lambda R(\boldsymbol{\theta}, \mathbf{W})$$

Today: Derive practical pipeline consisting of

- variable projection
- Stochastic Average Approximation (SAA)
- regularization

Variable Projection - 1

Idea: Treat learning problem as coupled optimization problem with blocks θ and $\mathbf{W}$.
Simple illustration for coupled least-squares problem [3, 2, 4]

$$\min_{\theta,\mathbf{w}} \phi(\theta,\mathbf{w}) = \frac{1}{2}\|\mathbf{A}(\theta)\mathbf{w}-\mathbf{c}\|^2 + \frac{\lambda}{2}\|\mathbf{L}\mathbf{w}\|^2 + \frac{\beta}{2}\|\mathbf{M}\theta\|^2$$

Note that for given θ the problem becomes a standard least-squares problem. Define:

$$\mathbf{w}(\theta) = \left(\mathbf{A}(\theta)^\top\mathbf{A}(\theta) + \lambda\mathbf{L}^\top\mathbf{L}\right)^{-1}\mathbf{A}(\theta)^\top\mathbf{c}$$

This gives optimization problem in θ only (aka *reduced/projected problem*)

$$\min_{\theta} \tilde{\phi}(\theta) = \frac{1}{2}\|\mathbf{A}(\theta)\mathbf{w}(\theta)-\mathbf{c}\|^2 + \frac{\lambda}{2}\|\mathbf{L}\mathbf{w}(\theta)\|^2 + \frac{\beta}{2}\|\mathbf{M}\theta\|^2$$

Variable Projection - 2

$$\min_\theta \tilde{\phi}(\theta) = \frac{1}{2}\|\mathbf{A}(\theta)\mathbf{w}(\theta) - \mathbf{c}\|^2 + \frac{\lambda}{2}\|\mathbf{L}\mathbf{w}(\theta)\|^2 + \frac{\beta}{2}\|\mathbf{M}\theta\|^2$$

Optimality condition:

$$\nabla\tilde{\phi}(\theta) = \nabla_\theta\phi(\theta, \mathbf{w}) + \nabla_\theta\mathbf{w}(\theta)\nabla_\mathbf{w}\phi(\theta, \mathbf{w}) \stackrel{!}{=} 0.$$

Less complicated than it seems since

$$\nabla_\mathbf{w}\phi(\theta, \mathbf{w}(\theta)) = \mathbf{A}(\theta)^\top(\mathbf{A}(\theta)\mathbf{w}(\theta) - \mathbf{c}) + \lambda\mathbf{L}^\top\mathbf{L}\mathbf{w}(\theta) = 0$$

Conclusion:

- ignore second term in gradient computation
- apply steepest descent or Gauss-Newton to minimize $\tilde{\phi}$
- need to solve least-squares problem in each evaluation of objective
- gradient is only correct if LS problem is solved exactly

Variable Projection for Single Layer

$$\min_{\theta,\mathbf{W}} E(\mathbf{W}\sigma(\mathbf{K}(\theta)\mathbf{Y}), \mathbf{C}) + \lambda R(\boldsymbol{\theta}, \mathbf{W})$$

Assume that the regularizer is separable, i.e.,

$$R(\theta, \mathbf{W}) = R_1(\theta) + R_2(\mathbf{W})$$

and that R_2 is convex and smooth. Hence, the projection requires solving the regularized classification problem

$$\mathbf{W}(\theta) = \arg\min_{\mathbf{W}} E(\mathbf{W}\sigma(\mathbf{K}(\theta)\mathbf{Y}), \mathbf{C}) + \lambda R_2(\mathbf{W})$$

practical considerations:

- solve for $\mathbf{W}(\theta)$ using Newton (need accuracy)
- need good solver to approximate gradient w.r.t. θ well
- use Gauss-Newton or steepest descent to solve for θ

Stochastic Optimization

Assume that each $\mathbf{y}_i$, $\mathbf{c}_i$ pair is drawn from some (unknown probability distribution).
Then, we can interpret the learning problem as minimizing the expected value of the cross entropy, e.g., in linear regression

$$E(\mathbf{W}) = \mathbb{E}\left(\frac{1}{2}\|\mathbf{y}^\top \mathbf{W} - \mathbf{c}\|^2\right)$$

This is a stochastic optimization problem [1].

Stochastic Optimization

Assume that each $\mathbf{y}_i$, $\mathbf{c}_i$ pair is drawn from some (unknown probability distribution).
Then, we can interpret the learning problem as minimizing the expected value of the cross entropy, e.g., in linear regression

$$E(\mathbf{W}) = \mathbb{E}\left(\frac{1}{2}\|\mathbf{y}^\top \mathbf{W} - \mathbf{c}\|^2\right)$$

This is a stochastic optimization problem [1]. One idea:
Stochastic Approximation: Design iteration $\mathbf{W}_k \to \mathbf{W}^*$ so that expected value decreases.
Example: Stochastic Gradient Descent, ADAM, ...
Pro: sample can be small (*mini batch*)
Con: how to monitor objective, linesearch, descent, ...

Stochastic Average Approximation

Alternative way to solve stochastic optimization problem

$$E(\mathbf{W}) = \mathbb{E}\left(\frac{1}{2}\|\mathbf{y}^\top \mathbf{W} - \mathbf{c}\|^2\right)$$

Pick relatively large sample $S \subset \{1, \ldots, n\}$ and use deterministic optimization method to solve

$$\min_{\mathbf{W}} \frac{1}{2|S|} \sum_{s \in S} \|\mathbf{y}_s \mathbf{W} - \mathbf{c}_s^\top\|^2.$$

Pro: use your favorite solver, linesearch, stopping...
Con: large batches needed

Note: Sample stays fixed during iteration.

A Practical Approach

Let's combine VarPro and SAA to come up with a simple yet powerful training algorithm.

A Practical Approach

Let's combine VarPro and SAA to come up with a simple yet powerful training algorithm.

Pick (θ_0) randomly and then do one or more steps of:

1. randomly select samples S (large enough)
2. do a few steps of variable projection to get θ_k.
   - inner solver: a few steps of Newton's method
3. check training error on current batch and validation error

A Practical Approach

Let's combine VarPro and SAA to come up with a simple yet powerful training algorithm.

Pick (θ_0) randomly and then do one or more steps of:

1. randomly select samples S (large enough)
2. do a few steps of variable projection to get θ_k.
   - inner solver: a few steps of Newton's method
3. check training error on current batch and validation error

Possible problems:

- $|S|$ too small $\rightarrow$ training error small but no convergence
- $|S|$ too large $\rightarrow$ slow, overfitting (use regularization)
- Too few Newton steps in classification $\rightarrow$ inaccurate gradients, line search fails, ...

Data Preprocessing

Some practical tips

- Remove the mean of the data
- Scale it to be "reasonable" scale
- Data augmentation
- Some other (domain specific) data transforms (optical flow for motion?)

Regularization for Network Weights

- Note that there are many more degrees of freedom.
- Need to add regularization for $\mathbf{K}$
- $\mathbf{K}$ Generally, $\mathbf{K}$ is not "physical" - difficult to choose reasonable regularization.

The obvious choice: Tikhonov

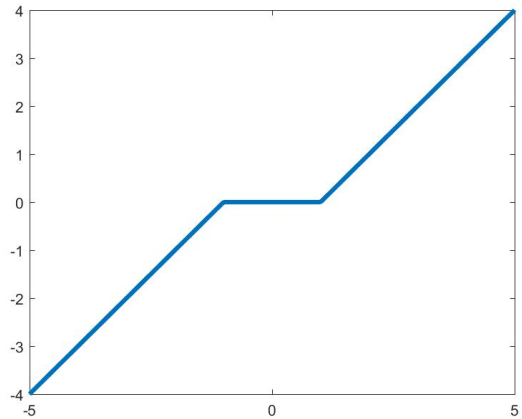
$$R(\mathbf{K}) = \frac{1}{2}\|\mathbf{K}\|_F^2$$

(also called weight decay)

Learning the weights - Regularization

More recent, demand that $\mathbf{K}$ is sparse

$$R(\mathbf{K}) = \|\mathrm{vec}(\mathbf{K})\|_1 = \sum_{ij} |\mathbf{K}_{ij}|$$



Implementation through soft-thresholding.
After each steepest descent iteration set

$$\mathbf{K} = \mathrm{softThresh}(\mathbf{K})$$

Obtain sparse matrices $\mathbf{K}$ that retain only necessary entries

Coding: Learning the weights

Class problem

Modify your steepest descent, nonlinear CG and SGD codes to work on single layer network with soft thresholding.
Test on Circle, peaks, spiral, MNIST and CIFAR10
Compare and report

References

[1] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization Methods for Large-Scale Machine Learning. *arXiv.org*, June 2016.

[2] G. Golub and V. Pereyra. Separable nonlinear least squares: the variable projection method and its applications. *Inverse Problems*, 19:R1–R26, 2003.

[3] G. H. Golub and V. Pereyra. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM Journal on Numerical Analysis*, 10(2):413–432, 1973.

[4] D. P. O'Leary and B. W. Rust. Variable projection for nonlinear least squares problems. *Computational Optimization and Applications. An International Journal*, 54(3):579–593, 2013.