Machine Learning (part II)

Optimization Strategies And Meta-Algorithms

Angelo Ciaramella

Introduction

- Many optimization techniques
  - General templates
  - Subroutines that can be incorporated into many different algorithms
- Methodologies
  - Batch Normalization
  - Coordinate descent
  - Polyak Averaging
  - Supervaised Pretraining
  - Design Models to Aid Optimization
  - Curriculum learning

Normalization and Standardization

- Input data
  - Normalization
  - Standardization

- Normalization
  - Values in the range [0, 1]

$$\tilde{\mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|_2}$$

$$\tilde{\mathbf{x}} = \frac{\mathbf{x} - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})}$$

Normalization and Standardization

- Standardization
  - zero mean
  - unit standard deviation

$$\mu_i = \frac{\mathbf{1}}{N}\sum_{i=1}^{N} x_i^n \qquad \sigma_i^2 = \frac{\mathbf{1}}{N-1}\sum_{i=1}^{N}(x_i^n - \mu_i)^2$$

$$\tilde{x}_i^n = \frac{x_i^n - \mu_i}{\sigma_i}$$

Normalization and Standardization

- Linear rescaling
  - Correlations amongst the variables

$$\mathbf{x} \;= (x_1, x_2, \dots, x_d)^T$$

$$\bar{\mathbf{x}} = \frac{\mathbf{1}}{N}\sum_{i=1}^{N} \mathbf{x}^n \qquad\qquad \Sigma \;= \frac{\mathbf{1}}{N-1}\sum_{i=1}^{N} (\mathbf{x}^n \;- \bar{\mathbf{x}}) \;\; (\mathbf{x}^n \;- \bar{\mathbf{x}})^T$$

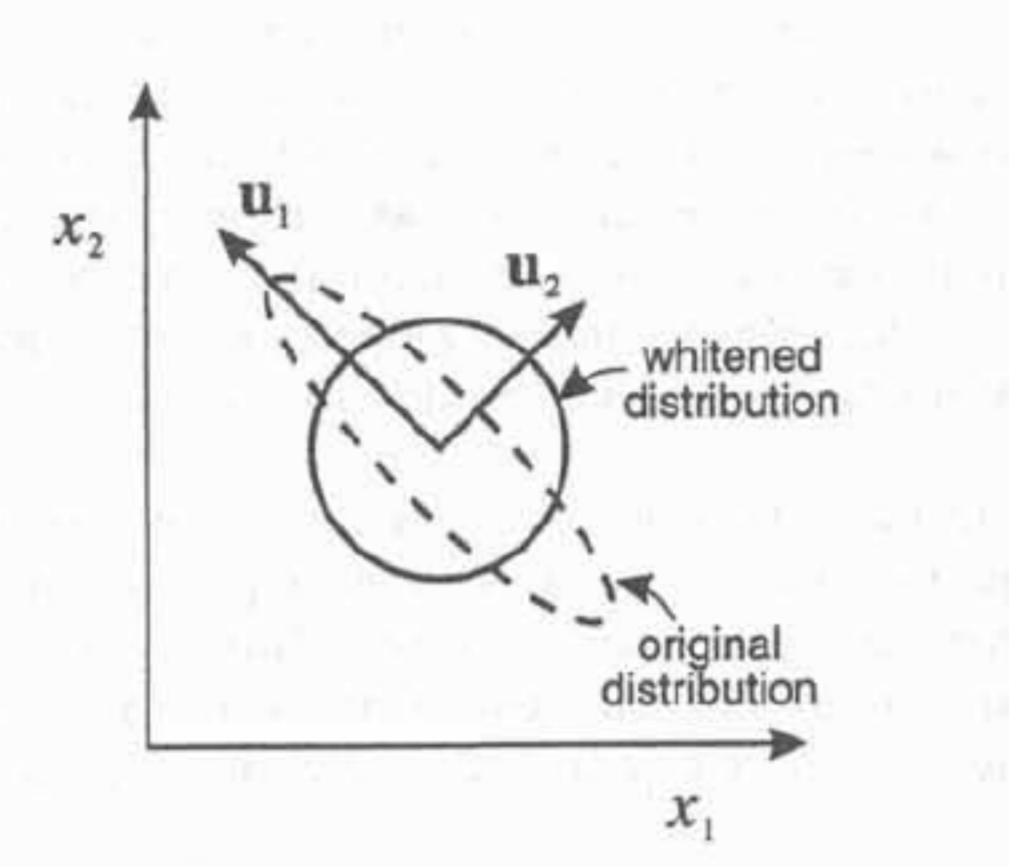
$$\Sigma \mathbf{u}_j = \lambda_j \mathbf{u}_j$$

$$\mathbf{U} \;= (u_1, u_2, \dots, u_d)$$

$$\tilde{\mathbf{x}}^n = \mathbf{\Lambda}^{-\frac{1}{2}} \mathbf{U}^T (\mathbf{x}^n \;- \bar{\mathbf{x}})$$

$$\mathbf{\Lambda} = diag(\lambda_1, \lambda_2, \dots, \lambda_d)$$

Whitening



Use of the eigenvectors of the covariance matrix of a distribution so that its Covariance matrix becomes the unit matrix

Batch Normalization

- Batch normalization
  - adaptive reparametrization
  - gradient update each parameter
    - all layers simultaneously

- DNN
  - Only one unit per layer
  - No activation functions

Batch Normalization

- DNN
  - Output

$$\hat{y} = x w_1 w_2 w_3 \ldots w_l$$

  - Output layer *i*

$$h_i = h_{i-1} w_i$$

  - Back-propagation algorithm

$$\boldsymbol{g} = \nabla_{\boldsymbol{w}} \hat{y} \qquad \boldsymbol{w} \leftarrow \boldsymbol{w} - \epsilon \boldsymbol{g}$$

  - New value

$$x(w_1 - \epsilon g_1)(w_2 - \epsilon g_2) \ldots (w_l - \epsilon g_l)$$

Batch Normalization

- Second order series approximation

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}^{(0)}) + (\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{g} + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}^{(0)})$$

$\boldsymbol{x}^{(0)} - \epsilon \boldsymbol{g}$ new point x

$$f(\boldsymbol{x}^{(0)} - \epsilon \boldsymbol{g}) \approx f(\boldsymbol{x}^{(0)}) - \epsilon \boldsymbol{g}^\top \boldsymbol{g} + \frac{1}{2}\epsilon^2 \boldsymbol{g}^\top \boldsymbol{H} \boldsymbol{g}$$

Batch Normalization

- Second-order term arising from this update

$$\epsilon^2 g_1 g_2 \prod_{i=3}^{l} w_i$$ can be large

- very hard to choose an appropriate learning rate

- Batch normalization
  - elegant way of reparametrizing almost any deep network
  - Reduces the problem of coordinating updates across many layers
  - applied to any input or hidden layer in a network

Batch Normalization

- Let H be a minibatch of activations of the layer to normalize

$$\boldsymbol{H}' = \frac{\boldsymbol{H} - \boldsymbol{\mu}}{\boldsymbol{\sigma}}$$

- Broadcasting the vector μ and the vector σ to be applied to every row of the matrix ***H***

- At training time

10^{-8}

$$\boldsymbol{\mu} = \frac{1}{m}\sum_i \boldsymbol{H}_{i,:} \qquad \boldsymbol{\sigma} = \sqrt{\delta + \frac{1}{m}\sum_i (\boldsymbol{H} - \boldsymbol{\mu})_i^2}$$

Batch Normalization

- At test time
  - μ and σ may be replaced by running averages that were collected during training time

- In order to maintain the expressive power of the network

$$\gamma H' + \beta$$

learned variables

Coordinate descent

- Goal
  - minimize $f(x)$ with respect to a single variable x_i
    - successivelly, minimize it with respect to another variable x_j and so on
    - repeatedly cycling through all variables
    - we are guaranteed to arrive at a (local) minimum

- Block coordinate descent
  - minimizing with respect to a subset of the variables simultaneously

Coordinate descent

- e.g., sparse coding

$$J(\boldsymbol{H}, \boldsymbol{W}) = \sum_{i,j} |H_{i,j}| + \sum_{i,j} \left(\boldsymbol{X} - \boldsymbol{W}^\top \boldsymbol{H}\right)^2_{i,j}$$

function J is not convex

- training algorithm into two sets
  - dictionary parameters W
  - code representations H
- Minimizing the objective function with respect to either one of these sets of variables is a convex problem
  - optimizing W with H fixed, then optimizing H with W fixed

Polyak Averaging

- Goal
  - averaging several points in the trajectory through parameter space visited by an optimization algorithm
  - t iterations of gradient descent visit points θ(1), . . . , θ(t)

$$\hat{\boldsymbol{\theta}}^{(t)} = \frac{1}{t}\sum_i \boldsymbol{\theta}^{(i)}$$

  - For non-convex problems

$$\hat{\boldsymbol{\theta}}^{(t)} = \alpha\hat{\boldsymbol{\theta}}^{(t-1)} + (1-\alpha)\boldsymbol{\theta}^{(t)}$$

Supervised pretraining

- Goal
  - train a simpler model to solve the task, then make the model more complex

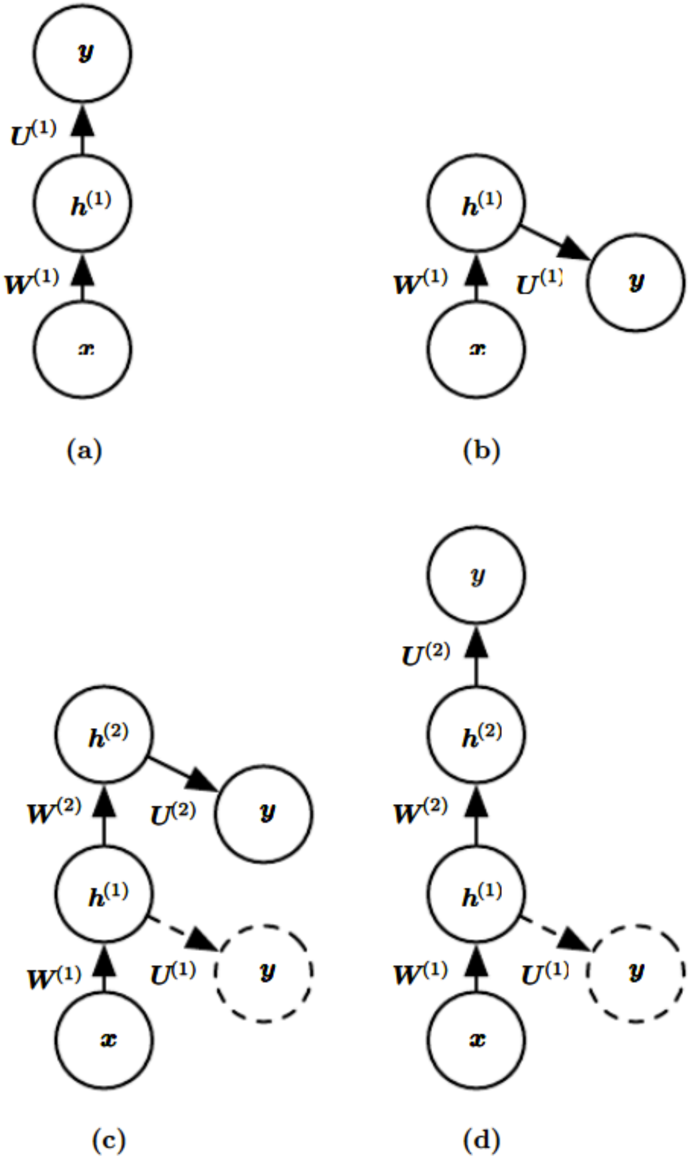
- Greedy algorithms
  - break a problem into many components
  - solve for the optimal version of each component in isolation
  - combining is not guaranteed to yield an optimal complete solution
  - followed by a fine-tuning stage
    - speed it up and improve the quality of the solution it
    - finds

Supervised pretraining

- Greedy supervised pretraining
  - supervised learning training task involving only a subset of the layers in the final neural network
  - each added hidden layer is pretrained as part of a shallow supervised MLP

- e.g., deep convolutional network (eleven weight layers)
  - Use the first four and last three layers from this network to initialize even deeper networks
    - with up to nineteen layers of weights
  - The middle layers of the new deep network are initialized randomly

Supervised pretraining



Greedy pretraining

Supervised pretraining

- FitNets
  - Teacher - training a network that has low enough depth and great enough width (number of units per layer) to be easy to train
  - Student - much deeper and thinner (eleven to nineteen layers) and would be difficult to train with SGD under normal circumstances

- Training
  - predict the output for the original task
  - predict the value of the middle layer of the teacher network

Continuation methods

- Goal
  - choosing initial points to ensure that local optimization spends most of its time in well-behaved regions of space
  - construct a series of objective functions over the same parameters
  - “blurring” the original cost function

$$J^{(i)}(\boldsymbol{\theta}) = \mathbb{E}_{\theta' \sim \mathcal{N}(\boldsymbol{\theta}';\boldsymbol{\theta},\sigma^{(i)2})} J(\boldsymbol{\theta}')$$

Continuation methods

- Curriculum learning (or shaping)
  - learning process to begin by learning simple concepts
  - progress to learning more complex concepts that depend on these simpler concepts
- stochastic curriculum
  - random mix of easy and difficult examples is always presented to the learner
  - the average proportion of the more difficult examples is gradually increased