Practical Issues in Deep Learning

Vineeth N Balasubramanian
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

CS5480: Deep Learning

29th Jan 2018

Challenges in GD: How to address?

Algorithmic Approaches

- Batch GD, SGD, Mini-batch SGD
- Momentum, Nesterov Momentum
- Adagrad, Adadelta, RMSProp, Adam
- Advanced Optimization Methods

Last session

Practical Tricks

- Regularization Methods (including DropOut)
- Data Manipulation Methods
- Parameter Choices/Initialization Methods (Activation Functions, Loss Functions, Weights)

Now

Outline

1. Regularization Methods
2. Data Manipulation Methods
3. Parameter Choices/Initialization Methods
4. Takeaways and Readings

Difference between Machine Learning and Optimization

- Thoughts?

Difference between Machine Learning and Optimization

- Thoughts? Generalization!
- In mainstream optimization, minimizing J is itself the goal; whereas in deep learning, minimizing J so as to minimize a generalizable out-of-sample performance measure is the goal
- Empirical Risk Minimization (ERM):

$$\mathbb{E}_{\mathbf{x},y\approx\hat{p}_{data}(\mathbf{x},y)}(J(\theta;\mathbf{x},y)) = \frac{1}{m}\sum_{i=1}^{m} J(\theta;\mathbf{x}_i,y_i)$$

- However, ERM can lead to overfitting. Avoiding overfitting is regularization.

Learning and Generalization

What's my rule?

- 1 2 3 $\implies$ satisfies rule
- 4 5 6 $\implies$ satisfies rule
- 7 8 9 $\implies$ satisfies rule
- 9 2 31 $\implies$ does not satisfy rule

Learning and Generalization

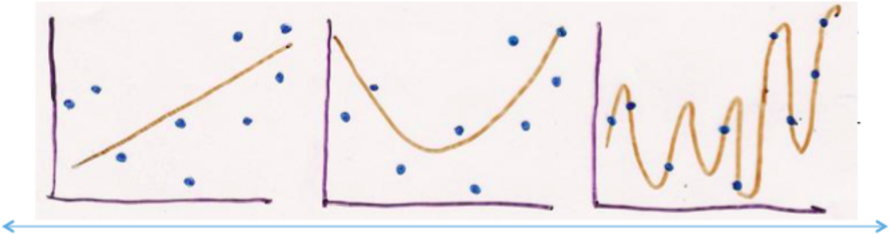
What's my rule?

- 1 2 3 $\Longrightarrow$ satisfies rule
- 4 5 6 $\Longrightarrow$ satisfies rule
- 7 8 9 $\Longrightarrow$ satisfies rule
- 9 2 31 $\Longrightarrow$ does not satisfy rule

Plausible rules

- 3 consecutive single digits
- 3 consecutive integers
- 3 numbers in ascending order
- 3 numbers whose sum is less than 25
- 3 numbers < 10
- 1, 4, or 6 in first column
- "yes" to first 3 sequences, "no" to all others

Overfitting

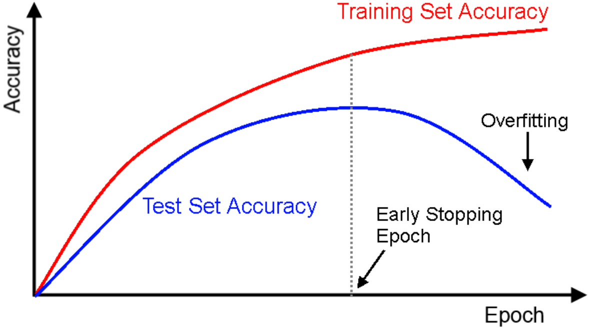


- simple model
- constrained
- small capacity may prevent it from representing all structure in data

- complex model
- unconstrained
- large capacity may allow it to memorize data and fail to capture regularities

Early Stopping

- Simple idea to keep monitoring the cost function, and not let it become too consistently low; stop at an earlier iteration



Early Stopping

When to stop?

- Train n epochs; lower learning rate; train m epochs $\rightarrow$ Bad idea: can't assume one-size-fits-all approach
- *Error-change criterion:*
  - Stop when error isn't dropping over a window of, say, 10 epochs
  - Train for a fixed number of epochs after criterion is reached (possibly with lower learning rate)
- *Weight-change criterion:*
  - Compare weights at epochs t-10 and t and test: $\max_i \|w_i^t - w_i^{t-10}\| < \rho$
  - Don't base on length of overall weight change vector
  - Possibly express as a percentage of the weight

Weight Decay

- We have already seen a regularization method: weight decay in gradient descent

L2-Weight Decay Term

$$J(\theta) = \left[\frac{1}{m} \sum_{i=1}^{m} \left(\frac{1}{2} \left\| h_\theta(x^{(i)}) - y^{(i)} \right\|^2 \right) \right] + \boxed{\frac{\lambda}{2} \sum_{l=1}^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} \left(W_{ji}^{(l)} \right)^2}$$

L1-Weight Decay Term

$$J(\theta) = \left[\frac{1}{m} \sum_{i=1}^{m} \left(\frac{1}{2} \left\| h_\theta(x^{(i)}) - y^{(i)} \right\|^2 \right) \right] + \boxed{\lambda \sum_{l=1}^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} \left| W_{ji}^{(l)} \right|}$$

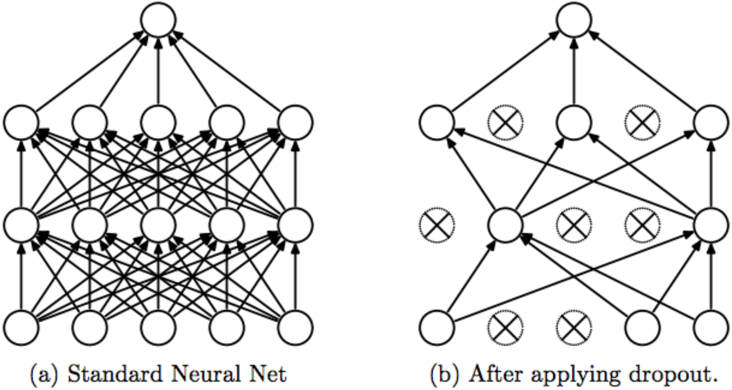
DropOut

- Another standard approach to regularization in ML: *Model Averaging*
- DropOut $\rightarrow$ a very interesting way to perform model averaging in deep learning

DropOut

- Another standard approach to regularization in ML: *Model Averaging*
- DropOut $\rightarrow$ a very interesting way to perform model averaging in deep learning
- Training Phase: For each hidden layer, for each training sample, for each iteration, ignore (zero out) a random fraction, p, of nodes (and corresponding activations)
- Test Phase: Use all activations, but reduce them by a factor p (to account for the missing activations during training)

DropOut



[1]

[1] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting", JMLR 2014

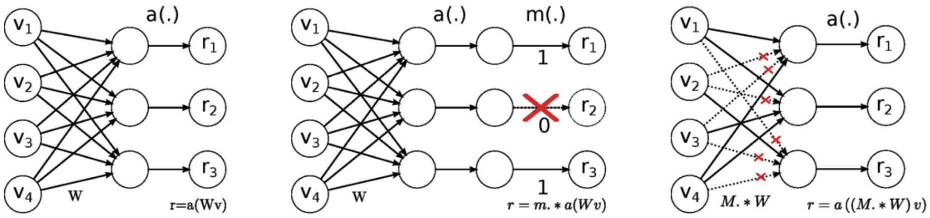
DropOut

- With H hidden units, each of which can be dropped, we have 2^H possible models
- Each of the 2^{H-1} models that include hidden unit h must share the same weights for the unit
  - serves as a form of regularization
  - makes the models cooperate
- Including all hidden units at test with a scaling of 0.5 is equivalent to computing the geometric mean of all 2^H models [2] [3]

[2] Hinton et al, Improving neural networks by preventing co-adaptation of feature detectors, 2012

[3] Warde-Farley et al, An empirical analysis of dropout in piecewise linear networks, 2014

DropConnect: An Extension



No-Drop Network DropOut Network DropConnect Network

a = activation function; m = dropping rate; M = binary mask matrix

[4] Wan, Li, et al. "Regularization of neural networks using dropconnect." ICML 2013

Noise in Data, Label and Gradient

Using noise is another form of regularization; has shown some impressive results recently. Could be:

- Data Noise
  - Has been there for a while: add noise to data while training
  - Minimization of sum-of-squares error with zero-mean gaussian noise(added to training data) is equivalent to minimization of sum-of-squares error without noise with an added regularized term [5]
  - Very similar to data augmentation that we will see later
- Label Noise
- Gradient Noise

[5] Bishop. Training with noise is equivalent to Tikhonov regularization. Neural Computation, 1995.

Regularization through Label Noise[6]

- Disturb each training sample with the probability α.
- For each disturbed sample, label is randomly drawn from a uniform distribution over $\{1, 2, \cdots, C\}$, regardless of the true label.

Algorithm 1 DisturbLabel

```
Input: D = {(x_n, y_n)}_{n=1}^N, noise rate α.
Initialization: a network model M: f(x; θ_0) ∈ R^C;
for each mini-batch D_t = {(x_m, y_m)}_{m=1}^M do
    for each sample (x_m, y_m) do
        Generate a disturbed label ỹ_m with Eqn (2);
    end for
    Update the parameters θ_t with Eqn (1);
end for
Output: the trained model M′: f(x; θ_T) ∈ R^C.
```

$$\begin{cases} \widetilde{c} & \sim & \mathcal{P}(\alpha), \\ \widetilde{y}_{\widetilde{c}} & = & 1, \\ \widetilde{y}_{i} & = & 0, \qquad \forall i \neq \widetilde{c}. \end{cases} \qquad (2)$$

[6] [illegible]

Regularization through Gradient Noise[7]

- Simple idea: add noise to gradient

$$g_t \leftarrow g_t + N(0, \sigma_t^2)$$

- Annealed Gaussian noise by decaying the variance

$$\sigma_t^2 = \frac{\eta}{(1+t)^\gamma}$$

- Showed significant improvement in performance

[7] Neelakantan, Arvind, et al. "Adding gradient noise improves learning for very deep networks." arXiv preprint arXiv:1511.06807 (2015).

Outline

1. Regularization Methods
2. Data Manipulation Methods
3. Parameter Choices/Initialization Methods
4. Takeaways and Readings

Data Transformation

- Normalize/standardize the inputs
  - Convergence is faster if average input over the training set is close to zero. Why? [8]
- Scaled to have the same covariance - speeds learning
  - Why?
  - Ideally, value of covariance should be matched with output of activation function (e.g. sigmoid)
  - Is this always necessary?

[8] Le Cun et al, Efficient Backprop, 1998

Data Transformation

- Normalize/standardize the inputs
  - Convergence is faster if average input over the training set is close to zero. Why? [8]
- Scaled to have the same covariance - speeds learning
  - Why?
  - Ideally, value of covariance should be matched with output of activation function (e.g. sigmoid)
  - Is this always necessary?

[8] Le Cun et al, Efficient Backprop, 1998

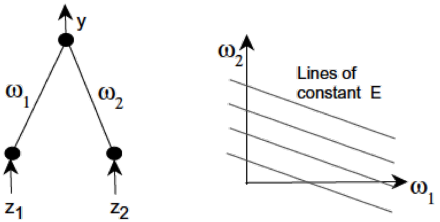
Data Transformation

- Normalize/standardize the inputs
  - Convergence is faster if average input over the training set is close to zero. Why? [8]
- Scaled to have the same covariance - speeds learning
  - Why?
  - Ideally, value of covariance should be matched with output of activation function (e.g. sigmoid)
  - Is this always necessary?

[8] Le Cun et al, Efficient Backprop, 1998

Data Transformation

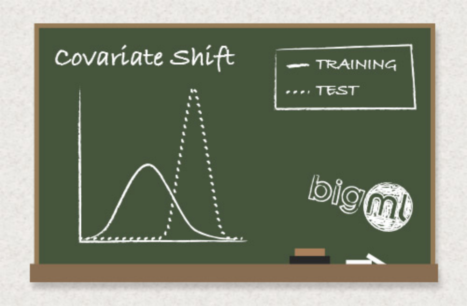
- Decorrelate the inputs
  - Why? Imagine one input is always twice the other, i.e. $z_2 = 2z_1$. Output y will be constant on lines $w_2 + \frac{1}{2}w_1 = \text{const.}$ No use making weight changes on these lines.
  - How? PCA!



Batch Normalization

Covariate Shift

- Change in distributions of data inputs is a problem because the model needs to continuously adapt to the new distribution → called covariate shift
- This is typically handled using domain adaptation



Batch normalization[9]

- What if this happens in a subnetwork in DL? $\rightarrow$ called internal covariate shift. How to handle?

[9] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint 2015

Batch normalization[9]

- What if this happens in a subnetwork in DL? → called internal covariate shift. How to handle?
- Whiten every layer's inputs → helps obtain a fixed distribution of inputs into each layer

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

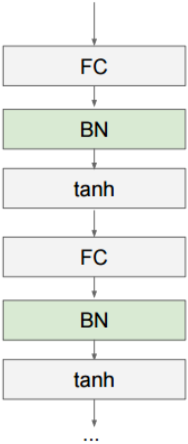
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

[9] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint 2015

Batch normalization



- BN layer usually inserted before non-linearity layer (after FC or convolutional layer)
- Allows higher learning rates
- Reduces the strong dependence on initialization
- Acts as a form of regularization too

How do we handle test time? Evaluate a mini-batch at a time?

Shuffling Inputs[10]

- Choose examples with maximum information content
  - Shuffle the training set so that successive training examples never (rarely) belong to the same class.
- Present input examples that produce a large error more frequently than examples that produce a small error. Why?

[10] LeCun, Yann A., et al. "Efficient backprop." Neural networks: Tricks of the Trade. Springer Berlin Heidelberg, 2012. 9-48.

Shuffling Inputs[10]

- Choose examples with maximum information content
  - Shuffle the training set so that successive training examples never (rarely) belong to the same class.
- Present input examples that produce a large error more frequently than examples that produce a small error. Why? Helps take large steps in the gradient descent
- Do you see any problems?

[10] LeCun, Yann A., et al. "Efficient backprop." Neural networks: Tricks of the Trade. Springer Berlin Heidelberg, 2012. 9-48.

Shuffling Inputs[10]

- Choose examples with maximum information content
  - Shuffle the training set so that successive training examples never (rarely) belong to the same class.
- Present input examples that produce a large error more frequently than examples that produce a small error. Why? Helps take large steps in the gradient descent
- Do you see any problems? What if the data sample is an outlier?

[10] LeCun, Yann A., et al. "Efficient backprop." Neural networks: Tricks of the Trade. Springer Berlin Heidelberg, 2012. 9-48.

Shuffling Inputs[10]

- Choose examples with maximum information content
  - Shuffle the training set so that successive training examples never (rarely) belong to the same class.
- Present input examples that produce a large error more frequently than examples that produce a small error. Why? Helps take large steps in the gradient descent
- Do you see any problems? What if the data sample is an outlier?
- Is this relevant for Batch GD?

[10] LeCun, Yann A., et al. "Efficient backprop." Neural networks: Tricks of the Trade. Springer Berlin Heidelberg, 2012. 9-48.

Curriculum Learning[11]

- Old idea, proposed by Elman in 1993
- Humans and animals learn much better when examples are not randomly presented but organized in a meaningful order which illustrates gradually more concepts, and gradually more complex ones.
- Start small, learn easier aspects of the task or easier sub-tasks, and then gradually increase the difficulty level
- By choosing examples and their order, one can guide training and remarkably increase learning speed
- Introduces the concept of a teacher who:
  - has prior knowledge about the training data to decide on a sequence of concepts that can more easily be learned when presented in that order
  - monitoring 'learner's progress to decide when to move on to new material from the curriculum

[11] Bengio, Yoshua, et al. "Curriculum learning." ICML 2009.

Curriculum Learning[11]

- Old idea, proposed by Elman in 1993
- Humans and animals learn much better when examples are not randomly presented but organized in a meaningful order which illustrates gradually more concepts, and gradually more complex ones.
- Start small, learn easier aspects of the task or easier sub-tasks, and then gradually increase the difficulty level
- By choosing examples and their order, one can guide training and remarkably increase learning speed
- Introduces the concept of a teacher who:
    - has prior knowledge about the training data to decide on a sequence of concepts that can more easily be learned when presented in that order
    - monitoring ‘learner’s progress to decide when to move on to new material from the curriculum

[11] Bengio, Yoshua, et al. "Curriculum learning." ICML 2009.

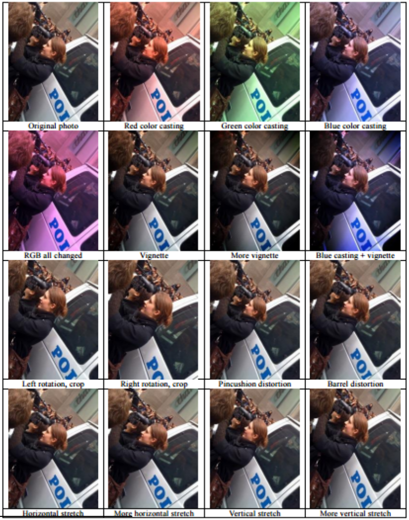
Data Augmentation

Methods

- Data jittering (E.g. Distortion and blurring of images)
- Rotations
- Color changes
- Noise injection
- Mirroring

- Helps increase data; is useful when training data provided is less (CNNs lead large amounts of training data to work!)
- Also acts as a regularizer (by avoiding overfitting to provided data)

Data Augmentation: Example [12]



[12] Wu, Ren, et al. "Deep image: Scaling up image recognition." arXiv 2015

Outline

1 Regularization Methods

2 Data Manipulation Methods

3 Parameter Choices/Initialization Methods

4 Takeaways and Readings

Parameter Choices

- Activation Functions: We discussed this earlier
- Loss Functions: We discussed this earlier
- Learning Rates: We discussed this earlier
  - All of them decrease it when weight vector "oscillates", and increase it when the weight vector follows a relatively steady direction
  - Worthwhile picking a different learning rate for each weight (e.g. based on curvature)

Choosing Target Values

- Assuming a binary classification problem, what do you choose the target labels to be? $+1$ and -1?

Choosing Target Values

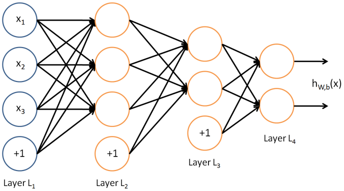
- Assuming a binary classification problem, what do you choose the target labels to be? $+1$ and -1?
- What if these are the sigmoid's asymptotes?
  - Weights will be increased continuously to very high values to match the target
  - Weights multiplied by small sigmoid derivative $\rightarrow$ small weight updates $\rightarrow$ Stuck!

Choosing Target Values

- Assuming a binary classification problem, what do you choose the target labels to be? $+1$ and -1?
- What if these are the sigmoid's asymptotes?
  - Weights will be increased continuously to very high values to match the target
  - Weights multiplied by small sigmoid derivative $\rightarrow$ small weight updates $\rightarrow$ Stuck!
- Choose target values at the point of the maximum second derivative on the sigmoid so as to avoid saturating the output units.

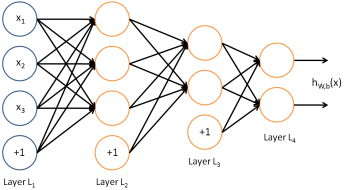
Weight Initialization

- What do you think? What if we started weights with zeroes?



Weight Initialization

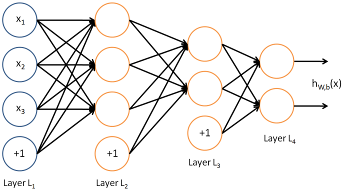
- What do you think? What if we started weights with zeroes?



- To be chosen randomly, but in such a way that the activation function is in its linear region
  - Both large and small weights can cause very low gradients (in case of sigmoid activation)

Weight Initialization

- What do you think? What if we started weights with zeroes?



- To be chosen randomly, but in such a way that the activation function is in its linear region
  - Both large and small weights can cause very low gradients (in case of sigmoid activation)
- Assuming inputs to a unit are uncorrelated with variance 1, standard deviation of units weighted sum is: $\sigma_{y_i} = (\Sigma_j w_{ij}^2)^{\frac{1}{2}}$
- Then weights should be randomly drawn from a distribution with mean zero and a standard deviation given by: $\sigma_w = m^{-\frac{1}{2}}$, where m is the node's fan-in.

Weight Initialization

Most recommended today (removed the need for unsupervised pre-training):

- Xavier's initialization[13]: $\texttt{uniform}(-\frac{\sqrt{6}}{\sqrt{fan_{in}+fan_{out}}}, \frac{\sqrt{6}}{\sqrt{fan_{in}+fan_{out}}})$
- Caffe implements a simpler version of Xavier's initialization as: $\texttt{uniform}(-\frac{2}{fan_{in}+fan_{out}}, \frac{2}{fan_{in}+fan_{out}})$
- He's initialization[14]: $\texttt{uniform}(-\frac{4}{fan_{in}+fan_{out}}, \frac{4}{fan_{in}+fan_{out}})$

[13] Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." AISTATS 2010

[14] He, Kaiming, et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." CVPR 2015

Weight Initialization

Still an active area of research...

- Understanding the difficulty o training deep feedforward neural networks by Glorot and Bengio, 2010
- Exact solutions to the nonlinear dynamics of learning in deep linear neural networks by Saxe et al, 2013
- Random walk initialization for training very deep feedforward networks by Sussillo and Abbott, 2014
- Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification by He et al., 2015
- Data-dependent Initializations of Convolutional Neural Networks by Krähenbühl et al., 2015
- All you need is a good init, Mishkin and Matas, 2015

Outline

1 Regularization Methods

2 Data Manipulation Methods

3 Parameter Choices/Initialization Methods

4 Takeaways and Readings

Takeaways

- Some standard choices for training deep networks: SGD + Nesterov momentum, SGD with Adagrad/RMSProp/Adam
- ReLUs, Leaky ReLUs and MaxOut are the best bets for activation functions
- Batch Normalization layers are here to stay (at least, for now)
- DropOut is an excellent regularizer
- Data Augmentation is a must in vision applications
- Weight Initialization is very important while training a new network

Readings

- Deep Learning book, Sections 7.1-7.5, 7.8, 7.12: http://www.deeplearningbook.org/contents/regularization.html
- Efficient Backprop by Yann Le Cun, 1998: http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf