ECBM E4040 Neural Networks and Deep Learning

Regularization of Deep or Distributed Models

Instructor Zoran Kostic

Department of Electrical Engineering
Columbia University

Oct. 1, 2016

Outline

1. Part I: Review of Previous Lecture
    - Summary of Feedforward Deep Networks
2. Regularization from a Bayesian Perspective
3. Classical Regularization: Parameter Norm Penalty
    - L^2 Parameter Regularization
    - L^1 Parameter Regularization
4. Regularizations that can be approximated by Penalty Regularizations
    - Regularization via Injecting Noise
    - Early Stopping
5. Other techniques for regularization
    - Dataset Augmentation
    - Adversarial Training
    - Parameter Sharing
    - Multi-Task Learning
    - Bootstrap Aggregating (Bagging)
    - Dropout

Outline

1. Part I: Review of Previous Lecture
   - Summary of Feedforward Deep Networks
2. Regularization from a Bayesian Perspective
3. Classical Regularization: Parameter Norm Penalty
   - L^2 Parameter Regularization
   - L^1 Parameter Regularization
4. Regularizations that can be approximated by Penalty Regularizations
   - Regularization via Injecting Noise
   - Early Stopping
5. Other techniques for regularization
   - Dataset Augmentation
   - Adversarial Training
   - Parameter Sharing
   - Multi-Task Learning
   - Bootstrap Aggregating (Bagging)
   - Dropout

Outline

1. Part I: Review of Previous Lecture
   - Summary of Feedforward Deep Networks
2. Regularization from a Bayesian Perspective
3. Classical Regularization: Parameter Norm Penalty
   - L^2 Parameter Regularization
   - L^1 Parameter Regularization
4. Regularizations that can be approximated by Penalty Regularizations
   - Regularization via Injecting Noise
   - Early Stopping
5. Other techniques for regularization
   - Dataset Augmentation
   - Adversarial Training
   - Parameter Sharing
   - Multi-Task Learning
   - Bootstrap Aggregating (Bagging)
   - Dropout

Outline

1. Part I: Review of Previous Lecture
   - Summary of Feedforward Deep Networks
2. Regularization from a Bayesian Perspective
3. Classical Regularization: Parameter Norm Penalty
   - L^2 Parameter Regularization
   - L^1 Parameter Regularization
4. Regularizations that can be approximated by Penalty Regularizations
   - Regularization via Injecting Noise
   - Early Stopping
5. Other techniques for regularization
   - Dataset Augmentation
   - Adversarial Training
   - Parameter Sharing
   - Multi-Task Learning
   - Bootstrap Aggregating (Bagging)
   - Dropout

Outline

1. Part I: Review of Previous Lecture
   - Summary of Feedforward Deep Networks
2. Regularization from a Bayesian Perspective
3. Classical Regularization: Parameter Norm Penalty
   - L^2 Parameter Regularization
   - L^1 Parameter Regularization
4. Regularizations that can be approximated by Penalty Regularizations
   - Regularization via Injecting Noise
   - Early Stopping
5. Other techniques for regularization
   - Dataset Augmentation
   - Adversarial Training
   - Parameter Sharing
   - Multi-Task Learning
   - Bootstrap Aggregating (Bagging)
   - Dropout

Outline

1. Part I: Review of Previous Lecture
   - Summary of Feedforward Deep Networks
2. Regularization from a Bayesian Perspective
3. Classical Regularization: Parameter Norm Penalty
   - L^2 Parameter Regularization
   - L^1 Parameter Regularization
4. Regularizations that can be approximated by Penalty Regularizations
   - Regularization via Injecting Noise
   - Early Stopping
5. Other techniques for regularization
   - Dataset Augmentation
   - Adversarial Training
   - Parameter Sharing
   - Multi-Task Learning
   - Bootstrap Aggregating (Bagging)
   - Dropout

Feedforward Deep Networks

- MLPs are realizations/implementations of compositions of multi-input and multi-output parametric functions.
- Examples of non-linear activation functions: sigmoid, tanh, softmax, radial basis functions, maxout, etc.
- Backpropagation: a method for computing gradients for multi-layer neural networks.
- The basic idea of the back-propagation algorithm is that the partial derivative of the cost J with respect to parameters $\boldsymbol{\theta}$ can be decomposed recursively by taking into consideration the composition of functions that relate $\boldsymbol{\theta}$ to J, via intermediate quantities that mediate that influence, e.g., the activations of hidden units in a deep neural network.

Feedforward Deep Networks

- MLPs are realizations/implementations of compositions of multi-input and multi-output parametric functions.
- Examples of non-linear activation functions: sigmoid, tanh, softmax, radial basis functions, maxout, etc.
- Backpropagation: a method for computing gradients for multi-layer neural networks.
- The basic idea of the back-propagation algorithm is that the partial derivative of the cost J with respect to parameters $\boldsymbol{\theta}$ can be decomposed recursively by taking into consideration the composition of functions that relate $\boldsymbol{\theta}$ to J, via intermediate quantities that mediate that influence, e.g., the activations of hidden units in a deep neural network.

Feedforward Deep Networks

- MLPs are realizations/implementations of compositions of multi-input and multi-output parametric functions.
- Examples of non-linear activation functions: sigmoid, tanh, softmax, radial basis functions, maxout, etc.
- Backpropagation: a method for computing gradients for multi-layer neural networks.
- The basic idea of the back-propagation algorithm is that the partial derivative of the cost J with respect to parameters $\boldsymbol{\theta}$ can be decomposed recursively by taking into consideration the composition of functions that relate $\boldsymbol{\theta}$ to J, via intermediate quantities that mediate that influence, e.g., the activations of hidden units in a deep neural network.

Feedforward Deep Networks

- MLPs are realizations/implementations of compositions of multi-input and multi-output parametric functions.
- Examples of non-linear activation functions: sigmoid, tanh, softmax, radial basis functions, maxout, etc.
- Backpropagation: a method for computing gradients for multi-layer neural networks.
- The basic idea of the back-propagation algorithm is that the partial derivative of the cost J with respect to parameters $\boldsymbol{\theta}$ can be decomposed recursively by taking into consideration the composition of functions that relate $\boldsymbol{\theta}$ to J, via intermediate quantities that mediate that influence, e.g., the activations of hidden units in a deep neural network.

Outline

1. Part I: Review of Previous Lecture
   - Summary of Feedforward Deep Networks
2. Regularization from a Bayesian Perspective
3. Classical Regularization: Parameter Norm Penalty
   - L^2 Parameter Regularization
   - L^1 Parameter Regularization
4. Regularizations that can be approximated by Penalty Regularizations
   - Regularization via Injecting Noise
   - Early Stopping
5. Other techniques for regularization
   - Dataset Augmentation
   - Adversarial Training
   - Parameter Sharing
   - Multi-Task Learning
   - Bootstrap Aggregating (Bagging)
   - Dropout

Regularization from a Bayesian Perspective

There is a deep connection between the Bayesian perspective on estimation and the process of regularization. Many forms of regularization can be given a Bayesian interpretation.

Given a dataset $(\mathbf{x}^1, \ldots, \mathbf{x}^m)$, the posterior $p(\boldsymbol{\theta}|\mathbf{x}^1, \ldots, \mathbf{x}^m)$ can be obtained by combining the data likelihood $p(\mathbf{x}^1, \ldots, \mathbf{x}^m|\boldsymbol{\theta})$ with the prior belief on the parameter encapsulated by $p(\boldsymbol{\theta})$:

$$\log p(\boldsymbol{\theta}|\mathbf{x}^1, \ldots, \mathbf{x}^m) = \log p(\boldsymbol{\theta}) + \sum_i \log p(\mathbf{x}^i|\boldsymbol{\theta}) + \text{constant}$$

where the constant is $-\log Z$, with the normalization constant Z that depends only on the data.

When maximizing over $\boldsymbol{\theta}$, this constant does not matter.

Regularization from a Bayesian Perspective

There is a deep connection between the Bayesian perspective on estimation and the process of regularization. Many forms of regularization can be given a Bayesian interpretation.

Given a dataset $(\mathbf{x}^1, \ldots, \mathbf{x}^m)$, the posterior $p(\boldsymbol{\theta}|\mathbf{x}^1, \ldots, \mathbf{x}^m)$ can be obtained by combining the data likelihood $p(\mathbf{x}^1, \ldots, \mathbf{x}^m|\boldsymbol{\theta})$ with the prior belief on the parameter encapsulated by $p(\boldsymbol{\theta})$:

$$\log p(\boldsymbol{\theta}|\mathbf{x}^1, \ldots, \mathbf{x}^m) = \log p(\boldsymbol{\theta}) + \sum_i \log p(\mathbf{x}^i|\boldsymbol{\theta}) + \text{constant}$$

where the constant is $-\log Z$, with the normalization constant Z that depends only on the data.

When maximizing over $\boldsymbol{\theta}$, this constant does not matter.

Regularization from a Bayesian Perspective

There is a deep connection between the Bayesian perspective on estimation and the process of regularization. Many forms of regularization can be given a Bayesian interpretation.

Given a dataset $(\mathbf{x}^1, \ldots, \mathbf{x}^m)$, the posterior $p(\boldsymbol{\theta}|\mathbf{x}^1, \ldots, \mathbf{x}^m)$ can be obtained by combining the data likelihood $p(\mathbf{x}^1, \ldots, \mathbf{x}^m|\boldsymbol{\theta})$ with the prior belief on the parameter encapsulated by $p(\boldsymbol{\theta})$:

$$\log p(\boldsymbol{\theta}|\mathbf{x}^1, \ldots, \mathbf{x}^m) = \log p(\boldsymbol{\theta}) + \sum_i \log p(\mathbf{x}^i|\boldsymbol{\theta}) + \text{constant}$$

where the constant is $-\log Z$, with the normalization constant Z that depends only on the data.

When maximizing over $\boldsymbol{\theta}$, this constant does not matter.

Regularization from a Bayesian Perspective (cont'd)

In the context of maximum likelihood learning, the introduction of the prior distribution plays the same role as a regularizer in that it can be seen as a term (the first one below)

$$\log p(\boldsymbol{\theta}|\mathbf{x}^1, \ldots, \mathbf{x}^m) = \log p(\boldsymbol{\theta}) + \sum_i \log p(\mathbf{x}^i|\boldsymbol{\theta}) + \text{constant}$$

added to the objective function that is added (to the second term, the log-likelihood) in hopes of achieving better generalization, despite of its detrimental effect on the likelihood of the training data.

Regularization from a Bayesian Perspective (cont'd)

In the context of maximum likelihood learning, the introduction of the prior distribution plays the same role as a regularizer in that it can be seen as a term (the first one below)

$$\log p(\boldsymbol{\theta}|\mathbf{x}^1, \ldots, \mathbf{x}^m) = \log p(\boldsymbol{\theta}) + \sum_i \log p(\mathbf{x}^i|\boldsymbol{\theta}) + \text{constant}$$

added to the objective function that is added (to the second term, the log-likelihood) in hopes of achieving better generalization, despite of its detrimental effect on the likelihood of the training data.

Regularization from a Bayesian Perspective (cont'd)

In the context of maximum likelihood learning, the introduction of the prior distribution plays the same role as a regularizer in that it can be seen as a term (the first one below)

$$\log p(\boldsymbol{\theta}|\mathbf{x}^1, \ldots, \mathbf{x}^m) = \log p(\boldsymbol{\theta}) + \sum_i \log p(\mathbf{x}^i|\boldsymbol{\theta}) + \text{constant}$$

added to the objective function that is added (to the second term, the log-likelihood) in hopes of achieving better generalization, despite of its detrimental effect on the likelihood of the training data.

Outline

1. Part I: Review of Previous Lecture
   - Summary of Feedforward Deep Networks
2. Regularization from a Bayesian Perspective
3. Classical Regularization: Parameter Norm Penalty
   - L^2 Parameter Regularization
   - L^1 Parameter Regularization
4. Regularizations that can be approximated by Penalty Regularizations
   - Regularization via Injecting Noise
   - Early Stopping
5. Other techniques for regularization
   - Dataset Augmentation
   - Adversarial Training
   - Parameter Sharing
   - Multi-Task Learning
   - Bootstrap Aggregating (Bagging)
   - Dropout

Classical Regularization

We denote the regularized objective function by $\tilde{J}$:

$$\tilde{J}(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) = J(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) + \alpha\Omega(\boldsymbol{\theta}),$$

where α is a hyperparameter that weights the relative contribution of the norm penalty term, Ω, relative to the standard objective function $J(\mathbf{x}; \boldsymbol{\theta})$.

The hyperparameter α is a non-negative real number. Setting $\alpha = 0$ results in no regularization. Larger values of α correspond to more regularization.

Note that for neural networks, we typically use a parameter norm penalty Ω that only penalizes the interaction weights. The offsets are left unregularized. The offsets typically require less data to fit accurately than the weights.

Classical Regularization

We denote the regularized objective function by $\tilde{J}$:

$$\tilde{J}(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) = J(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) + \alpha \Omega(\boldsymbol{\theta}),$$

where α is a hyperparameter that weights the relative contribution of the norm penalty term, Ω, relative to the standard objective function $J(\mathbf{x}; \boldsymbol{\theta})$.

The hyperparameter α is a non-negative real number. Setting $\alpha = 0$ results in no regularization. Larger values of α correspond to more regularization.

Note that for neural networks, we typically use a parameter norm penalty Ω that only penalizes the interaction weights. The offsets are left unregularized. The offsets typically require less data to fit accurately than the weights.

Classical Regularization

We denote the regularized objective function by $\tilde{J}$:

$$\tilde{J}(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) = J(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) + \alpha\Omega(\boldsymbol{\theta}),$$

where α is a hyperparameter that weights the relative contribution of the norm penalty term, Ω, relative to the standard objective function $J(\mathbf{x}; \boldsymbol{\theta})$.

The hyperparameter α is a non-negative real number. Setting $\alpha = 0$ results in no regularization. Larger values of α correspond to more regularization.

Note that for neural networks, we typically use a parameter norm penalty Ω that only penalizes the interaction weights. The offsets are left unregularized. The offsets typically require less data to fit accurately than the weights.

L^2 Parameter Regularization

The L^2 parameter norm penalty commonly known as weight decay is given by $\Omega(\boldsymbol{\theta}) = \frac{1}{2}||\mathbf{w}||_2^2$.

As we will see, the L^2 regularization strategy drives the parameters closer to the origin. To simplify the presentation, we assume that $\boldsymbol{\theta} = \mathbf{w}$ (no offset term). The gradient of the total objective function amounts to

$$\nabla_{\mathrm{w}}\tilde{J}(\mathbf{w};\mathbf{X},\mathbf{y}) = \alpha\mathbf{w} + \nabla_{\mathrm{w}}J(\mathbf{w};\mathbf{X},\mathbf{y}).$$

A single gradient step for updating the weights is

$$\mathbf{w} \leftarrow \mathbf{w} - \epsilon(\alpha\mathbf{w} + \nabla_{\mathrm{w}}J(\mathbf{w};\mathbf{X},\mathbf{y})),$$

or

$$\mathbf{w} \leftarrow (1-\epsilon\alpha)\mathbf{w} - \epsilon\nabla_{\mathrm{w}}J(\mathbf{w};\mathbf{X},\mathbf{y})).$$

L^2 Parameter Regularization

The L^2 parameter norm penalty commonly known as weight decay is given by $\Omega(\boldsymbol{\theta}) = \frac{1}{2}||\mathbf{w}||_2^2$.

As we will see, the L^2 regularization strategy drives the parameters closer to the origin. To simplify the presentation, we assume that $\boldsymbol{\theta} = \mathbf{w}$ (no offset term). The gradient of the total objective function amounts to

$$\nabla_{\mathrm{w}}\tilde{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \alpha\mathbf{w} + \nabla_{\mathrm{w}}J(\mathbf{w}; \mathbf{X}, \mathbf{y}).$$

A single gradient step for updating the weights is

$$\mathbf{w} \leftarrow \mathbf{w} - \epsilon(\alpha\mathbf{w} + \nabla_{\mathrm{w}}J(\mathbf{w}; \mathbf{X}, \mathbf{y})),$$

or

$$\mathbf{w} \leftarrow (1 - \epsilon\alpha)\mathbf{w} - \epsilon\nabla_{\mathrm{w}}J(\mathbf{w}; \mathbf{X}, \mathbf{y})).$$

L^2 Parameter Regularization

The L^2 parameter norm penalty commonly known as weight decay is given by $\Omega(\boldsymbol{\theta}) = \frac{1}{2}||\mathbf{w}||_2^2$.

As we will see, the L^2 regularization strategy drives the parameters closer to the origin. To simplify the presentation, we assume that $\boldsymbol{\theta} = \mathbf{w}$ (no offset term). The gradient of the total objective function amounts to

$$\nabla_{\mathrm{w}} \tilde{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \alpha \mathbf{w} + \nabla_{\mathrm{w}} J(\mathbf{w}; \mathbf{X}, \mathbf{y}).$$

A single gradient step for updating the weights is

$$\mathbf{w} \leftarrow \mathbf{w} - \epsilon(\alpha \mathbf{w} + \nabla_{\mathrm{w}} J(\mathbf{w}; \mathbf{X}, \mathbf{y})),$$

or

$$\mathbf{w} \leftarrow (1 - \epsilon\alpha)\mathbf{w} - \epsilon \nabla_{\mathrm{w}} J(\mathbf{w}; \mathbf{X}, \mathbf{y})).$$

L^2 Parameter Regularization

The L^2 parameter norm penalty commonly known as weight decay is given by $\Omega(\boldsymbol{\theta}) = \frac{1}{2}||\mathbf{w}||_2^2$.

As we will see, the L^2 regularization strategy drives the parameters closer to the origin. To simplify the presentation, we assume that $\boldsymbol{\theta} = \mathbf{w}$ (no offset term). The gradient of the total objective function amounts to

$$\nabla_{\mathrm{w}}\tilde{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \alpha\mathbf{w} + \nabla_{\mathrm{w}}J(\mathbf{w}; \mathbf{X}, \mathbf{y}).$$

A single gradient step for updating the weights is

$$\mathbf{w} \leftarrow \mathbf{w} - \epsilon(\alpha\mathbf{w} + \nabla_{\mathrm{w}}J(\mathbf{w}; \mathbf{X}, \mathbf{y})),$$

or

$$\mathbf{w} \leftarrow (1 - \epsilon\alpha)\mathbf{w} - \epsilon\nabla_{\mathrm{w}}J(\mathbf{w}; \mathbf{X}, \mathbf{y})).$$

L^2 Parameter Regularization

The L^2 parameter norm penalty commonly known as weight decay is given by $\Omega(\boldsymbol{\theta}) = \frac{1}{2}||\mathbf{w}||_2^2$.

As we will see, the L^2 regularization strategy drives the parameters closer to the origin. To simplify the presentation, we assume that $\boldsymbol{\theta} = \mathbf{w}$ (no offset term). The gradient of the total objective function amounts to

$$\nabla_{\mathrm{w}} \tilde{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \alpha \mathbf{w} + \nabla_{\mathrm{w}} J(\mathbf{w}; \mathbf{X}, \mathbf{y}).$$

A single gradient step for updating the weights is

$$\mathbf{w} \leftarrow \mathbf{w} - \epsilon(\alpha \mathbf{w} + \nabla_{\mathrm{w}} J(\mathbf{w}; \mathbf{X}, \mathbf{y})),$$

or

$$\mathbf{w} \leftarrow (1 - \epsilon\alpha)\mathbf{w} - \epsilon \nabla_{\mathrm{w}} J(\mathbf{w}; \mathbf{X}, \mathbf{y})).$$

L^2 Parameter Regularization (cont'd)

To simplify the analysis, we consider a quadratic approximation to the objective function in the neighborhood of the empirically optimal value of the weights $\mathbf{w}^*$.

$$\hat{J}(\mathbf{w}) = J(\mathbf{w}^*) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^T \mathbf{H}(\mathbf{w} - \mathbf{w}^*)$$

where $\mathbf{H}$ of J with respect to $\mathbf{w}$ evaluated at $\mathbf{w}^*$.

There is no first order term in this quadratic approximation, because $\mathbf{w}^*$ is defined to be a minimum, where the gradient vanishes.

Likewise, because $\mathbf{w}^*$ is a minimum, we can conclude that $\mathbf{H}$ is positive semi-definite and

$$\nabla_{\mathrm{w}} \hat{J}(\mathbf{w}) = \mathbf{H}(\mathbf{w} - \mathbf{w}^*).$$

L^2 Parameter Regularization (cont'd)

To simplify the analysis, we consider a quadratic approximation to the objective function in the neighborhood of the empirically optimal value of the weights $\mathbf{w}^*$.

$$\hat{J}(\mathbf{w}) = J(\mathbf{w}^*) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^T \mathbf{H}(\mathbf{w} - \mathbf{w}^*)$$

where $\mathbf{H}$ of J with respect to $\mathbf{w}$ evaluated at $\mathbf{w}^*$.

There is no first order term in this quadratic approximation, because $\mathbf{w}^*$ is defined to be a minimum, where the gradient vanishes.

Likewise, because $\mathbf{w}^*$ is a minimum, we can conclude that $\mathbf{H}$ is positive semi-definite and

$$\nabla_{\mathbf{w}} \hat{J}(\mathbf{w}) = \mathbf{H}(\mathbf{w} - \mathbf{w}^*).$$

L^2 Parameter Regularization (cont'd)

To simplify the analysis, we consider a quadratic approximation to the objective function in the neighborhood of the empirically optimal value of the weights $\mathbf{w}^*$.

$$\hat{J}(\mathbf{w}) = J(\mathbf{w}^*) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^T \mathbf{H}(\mathbf{w} - \mathbf{w}^*)$$

where $\mathbf{H}$ of J with respect to $\mathbf{w}$ evaluated at $\mathbf{w}^*$.

There is no first order term in this quadratic approximation, because $\mathbf{w}^*$ is defined to be a minimum, where the gradient vanishes.

Likewise, because $\mathbf{w}^*$ is a minimum, we can conclude that $\mathbf{H}$ is positive semi-definite and

$$\nabla_{\mathbf{w}} \hat{J}(\mathbf{w}) = \mathbf{H}(\mathbf{w} - \mathbf{w}^*).$$

L^2 Parameter Regularization (cont'd)

To simplify the analysis, we consider a quadratic approximation to the objective function in the neighborhood of the empirically optimal value of the weights $\mathbf{w}^*$.

$$\hat{J}(\mathbf{w}) = J(\mathbf{w}^*) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^T \mathbf{H}(\mathbf{w} - \mathbf{w}^*)$$

where $\mathbf{H}$ of J with respect to $\mathbf{w}$ evaluated at $\mathbf{w}^*$.

There is no first order term in this quadratic approximation, because $\mathbf{w}^*$ is defined to be a minimum, where the gradient vanishes.

Likewise, because $\mathbf{w}^*$ is a minimum, we can conclude that $\mathbf{H}$ is positive semi-definite and

$$\nabla_{\mathrm{w}} \hat{J}(\mathbf{w}) = \mathbf{H}(\mathbf{w} - \mathbf{w}^*).$$

L^2 Parameter Regularization (cont'd)

The location of the minimum of the regularized objective function is given by

$$\alpha \mathbf{w} + \mathbf{H}(\mathbf{w} - \mathbf{w}^*) = 0,$$

or

$$(\mathbf{H} + \alpha \mathbf{I})\mathbf{w} = \mathbf{H}\mathbf{w}^*,$$

or finally,

$$\tilde{\mathbf{w}} = (\mathbf{H} + \alpha \mathbf{I})^{-1}\mathbf{H}\mathbf{w}^*.$$

The regularization term moves the optimum from $\mathbf{w}^*$ to $\tilde{\mathbf{w}}$. As α approaches 0, $\tilde{\mathbf{w}}$ approaches $\mathbf{w}^*$.

L^2 Parameter Regularization (cont'd)

The location of the minimum of the regularized objective function is given by

$$\alpha \mathbf{w} + \mathbf{H}(\mathbf{w} - \mathbf{w}^*) = 0,$$

or

$$(\mathbf{H} + \alpha \mathbf{I})\mathbf{w} = \mathbf{H}\mathbf{w}^*,$$

or finally,

$$\tilde{\mathbf{w}} = (\mathbf{H} + \alpha \mathbf{I})^{-1}\mathbf{H}\mathbf{w}^*.$$

The regularization term moves the optimum from $\mathbf{w}^*$ to $\tilde{\mathbf{w}}$. As α approaches 0, $\tilde{\mathbf{w}}$ approaches $\mathbf{w}^*$.

L^2 Parameter Regularization (cont'd)

The location of the minimum of the regularized objective function is given by

$$\alpha \mathbf{w} + \mathbf{H}(\mathbf{w} - \mathbf{w}^*) = 0,$$

or

$$(\mathbf{H} + \alpha \mathbf{I})\mathbf{w} = \mathbf{H}\mathbf{w}^*,$$

or finally,

$$\tilde{\mathbf{w}} = (\mathbf{H} + \alpha \mathbf{I})^{-1}\mathbf{H}\mathbf{w}^*.$$

The regularization term moves the optimum from $\mathbf{w}^*$ to $\tilde{\mathbf{w}}$. As α approaches 0, $\tilde{\mathbf{w}}$ approaches $\mathbf{w}^*$.

L^2 Parameter Regularization (cont'd)

The location of the minimum of the regularized objective function is given by

$$\alpha \mathbf{w} + \mathbf{H}(\mathbf{w} - \mathbf{w}^*) = 0,$$

or

$$(\mathbf{H} + \alpha \mathbf{I})\mathbf{w} = \mathbf{H}\mathbf{w}^*,$$

or finally,

$$\tilde{\mathbf{w}} = (\mathbf{H} + \alpha \mathbf{I})^{-1}\mathbf{H}\mathbf{w}^*.$$

The regularization term moves the optimum from $\mathbf{w}^*$ to $\tilde{\mathbf{w}}$. As α approaches 0, $\tilde{\mathbf{w}}$ approaches $\mathbf{w}^*$.

L^2 Parameter Regularization (cont'd)

Here we investigate the case when α grows.
Because $\mathbf{H}$ is real and symmetric, we can decompose it into a diagonal matrix

$$\mathbf{H} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$$

with $\mathbf{\Lambda}$ having real eigenvalues, $\mathbf{Q}$ orthonormal eigenvectors and $\mathbf{Q}^{-1} = \mathbf{Q}^T$.
We have

$$\begin{aligned}\tilde{\mathbf{w}} &= (\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T + \alpha\mathbf{I})^{-1}\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T\mathbf{w}^* \\ &= [\mathbf{Q}(\mathbf{\Lambda} + \alpha\mathbf{I})\mathbf{Q}^T]^{-1}\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T\mathbf{w}^* \\ &= \mathbf{Q}(\mathbf{\Lambda} + \alpha\mathbf{I})^{-1}\mathbf{\Lambda}\mathbf{Q}^T\mathbf{w}^*,\end{aligned}$$

and finally

$$\mathbf{Q}^T\tilde{\mathbf{w}} = (\mathbf{\Lambda} + \alpha\mathbf{I})^{-1}\mathbf{\Lambda}\mathbf{Q}^T\mathbf{w}^*.$$

L^2 Parameter Regularization (cont'd)

Here we investigate the case when α grows.
Because $\mathbf{H}$ is real and symmetric, we can decompose it into a diagonal matrix

$$\mathbf{H} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T$$

with $\boldsymbol{\Lambda}$ having real eigenvalues, $\mathbf{Q}$ orthonormal eigenvectors and $\mathbf{Q}^{-1} = \mathbf{Q}^T$.
We have

$$\begin{aligned}\tilde{\mathbf{w}} &= (\mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T + \alpha\mathbf{I})^{-1}\mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T\mathbf{w}^* \\ &= [\mathbf{Q}(\boldsymbol{\Lambda} + \alpha\mathbf{I})\mathbf{Q}^T]^{-1}\mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T\mathbf{w}^* \\ &= \mathbf{Q}(\boldsymbol{\Lambda} + \alpha\mathbf{I})^{-1}\boldsymbol{\Lambda}\mathbf{Q}^T\mathbf{w}^*,\end{aligned}$$

and finally

$$\mathbf{Q}^T\tilde{\mathbf{w}} = (\boldsymbol{\Lambda} + \alpha\mathbf{I})^{-1}\boldsymbol{\Lambda}\mathbf{Q}^T\mathbf{w}^*.$$

L^2 Parameter Regularization (cont'd)

Here we investigate the case when α grows.
Because $\mathbf{H}$ is real and symmetric, we can decompose it into a diagonal matrix

$$\mathbf{H} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T$$

with $\boldsymbol{\Lambda}$ having real eigenvalues, $\mathbf{Q}$ orthonormal eigenvectors and $\mathbf{Q}^{-1} = \mathbf{Q}^T$.
We have

$$\begin{aligned}\tilde{\mathbf{w}} &= (\mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T + \alpha\mathbf{I})^{-1}\mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T\mathbf{w}^* \\ &= [\mathbf{Q}(\boldsymbol{\Lambda} + \alpha\mathbf{I})\mathbf{Q}^T]^{-1}\mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T\mathbf{w}^* \\ &= \mathbf{Q}(\boldsymbol{\Lambda} + \alpha\mathbf{I})^{-1}\boldsymbol{\Lambda}\mathbf{Q}^T\mathbf{w}^*,\end{aligned}$$

and finally

$$\mathbf{Q}^T\tilde{\mathbf{w}} = (\boldsymbol{\Lambda} + \alpha\mathbf{I})^{-1}\boldsymbol{\Lambda}\mathbf{Q}^T\mathbf{w}^*.$$

L^2 Parameter Regularization (cont'd)

$\mathbf{Q}^T\tilde{\mathbf{w}}$ is rotating our solution parameters $\tilde{\mathbf{w}}$ into the basis defined by the eigenvectors of $\mathbf{Q}$ of $\mathbf{H}$.

Consequently, the effect of weight decay is to rescale the coefficients of the eigenvectors. The i'th component is rescaled by a factor of $\frac{\lambda_i}{\lambda_i+\alpha}$.

Along the directions where the eigenvalues of $\mathbf{H}$ are relatively large, for example, where $\lambda_i >> \alpha$, the effect of regularization is relatively small. However, components with $\lambda_i << \alpha$ will be shrunk to have nearly zero magnitude.

L^2 Parameter Regularization (cont'd)

$\mathbf{Q}^T\tilde{\mathbf{w}}$ is rotating our solution parameters $\tilde{\mathbf{w}}$ into the basis defined by the eigenvectors of $\mathbf{Q}$ of $\mathbf{H}$.

Consequently, the effect of weight decay is to rescale the coefficients of the eigenvectors. The i'th component is rescaled by a factor of $\frac{\lambda_i}{\lambda_i+\alpha}$.

Along the directions where the eigenvalues of $\mathbf{H}$ are relatively large, for example, where $\lambda_i >> \alpha$, the effect of regularization is relatively small. However, components with $\lambda_i << \alpha$ will be shrunk to have nearly zero magnitude.

L^2 Parameter Regularization (cont'd)

$\mathbf{Q}^T\tilde{\mathbf{w}}$ is rotating our solution parameters $\tilde{\mathbf{w}}$ into the basis defined by the eigenvectors of $\mathbf{Q}$ of $\mathbf{H}$.

Consequently, the effect of weight decay is to rescale the coefficients of the eigenvectors. The i'th component is rescaled by a factor of $\frac{\lambda_i}{\lambda_i+\alpha}$.

Along the directions where the eigenvalues of $\mathbf{H}$ are relatively large, for example, where $\lambda_i >> \alpha$, the effect of regularization is relatively small. However, components with $\lambda_i << \alpha$ will be shrunk to have nearly zero magnitude.

L^1 Parameter Regularization

L^1 regularization on the model parameter $\mathbf{w}$ is defined by

$$\Omega(\boldsymbol{\theta}) = ||\mathbf{w}||_1 = \sum_i |\mathbf{w}_i|,$$

that is, as the sum of absolute values of the individual parameters.

The gradient of the regularized objective function is

$$\nabla_{\mathbf{w}} \tilde{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \beta \text{sign}(\mathbf{w}) + \nabla_{\mathbf{w}} J(\mathbf{w}; \mathbf{X}, \mathbf{y})$$

where $\text{sign}(\mathbf{w})$ is the sign of $\mathbf{w}$ applied element-wise.

Not that the regularization contribution to the gradient no longer scales linearly with $\mathbf{w}$, but instead it is a constant factor with a sign equal to $\text{sign}(\mathbf{w})$.

L^1 Parameter Regularization

L^1 regularization on the model parameter $\mathbf{w}$ is defined by

$$\Omega(\boldsymbol{\theta}) = ||\mathbf{w}||_1 = \sum_i |\mathbf{w}_i|,$$

that is, as the sum of absolute values of the individual parameters.

The gradient of the regularized objective function is

$$\nabla_{\mathrm{w}} \tilde{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \beta \mathrm{sign}(\mathbf{w}) + \nabla_{\mathrm{w}} J(\mathbf{w}; \mathbf{X}, \mathbf{y})$$

where $\mathrm{sign}(\mathbf{w})$ is the sign of $\mathbf{w}$ applied element-wise.

Not that the regularization contribution to the gradient no longer scales linearly with $\mathbf{w}$, but instead it is a constant factor with a sign equal to $\mathrm{sign}(\mathbf{w})$.

L^1 Parameter Regularization

L^1 regularization on the model parameter $\mathbf{w}$ is defined by

$$\Omega(\boldsymbol{\theta}) = ||\mathbf{w}||_1 = \sum_i |\mathbf{w}_i|,$$

that is, as the sum of absolute values of the individual parameters.

The gradient of the regularized objective function is

$$\nabla_{\mathrm{w}} \tilde{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \beta \mathrm{sign}(\mathbf{w}) + \nabla_{\mathrm{w}} J(\mathbf{w}; \mathbf{X}, \mathbf{y})$$

where $\mathrm{sign}(\mathbf{w})$ is the sign of $\mathbf{w}$ applied element-wise.

Not that the regularization contribution to the gradient no longer scales linearly with $\mathbf{w}$, but instead it is a constant factor with a sign equal to $\mathrm{sign}(\mathbf{w})$.

L^1 Parameter Regularization (cont'd)

The gradient of the approximation is given by

$$\nabla_{\mathbf{w}} \hat{J}(\mathbf{w}) = \mathbf{H}(\mathbf{w} - \mathbf{w}^*).$$

where $\mathbf{H}$ is the Hessian matrix of J with respect to $\mathbf{w}$ evaluated at $\mathbf{w}^*$.

To gain insight, we assume that the Hessian is diagonal, that is, $\mathbf{H} = \mathrm{diag}([\gamma_1, ..., \gamma_N])$, where each $\gamma_i > 0$.

With this assumption, the solution of the minimum of the L^1 regularized objective function decomposes into a system of equations of the form:

$$\tilde{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \frac{1}{2}\gamma_i(\mathbf{w}_i - \mathbf{w}_i^*)^2 + \beta|\mathbf{w}_i^*|.$$

L^1 Parameter Regularization (cont'd)

The gradient of the approximation is given by

$$\nabla_{\mathrm{w}} \hat{J}(\mathbf{w}) = \mathbf{H}(\mathbf{w} - \mathbf{w}^*).$$

where $\mathbf{H}$ is the Hessian matrix of J with respect to $\mathbf{w}$ evaluated at $\mathbf{w}^*$.

To gain insight, we assume that the Hessian is diagonal, that is, $\mathbf{H} = \mathrm{diag}([\gamma_1, \ldots, \gamma_N])$, where each $\gamma_i > 0$.

With this assumption, the solution of the minimum of the L^1 regularized objective function decomposes into a system of equations of the form:

$$\tilde{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \frac{1}{2}\gamma_i(\mathbf{w}_i - \mathbf{w}_i^*)^2 + \beta|\mathbf{w}_i^*|.$$

L^1 Parameter Regularization (cont'd)

The gradient of the approximation is given by

$$\nabla_{\mathrm{w}} \hat{J}(\mathbf{w}) = \mathbf{H}(\mathbf{w} - \mathbf{w}^*).$$

where $\mathbf{H}$ is the Hessian matrix of J with respect to $\mathbf{w}$ evaluated at $\mathbf{w}^*$.

To gain insight, we assume that the Hessian is diagonal, that is, $\mathbf{H} = \mathrm{diag}([\gamma_1, \ldots, \gamma_N])$, where each $\gamma_i > 0$.

With this assumption, the solution of the minimum of the L^1 regularized objective function decomposes into a system of equations of the form:

$$\tilde{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \frac{1}{2}\gamma_i(\mathbf{w}_i - \mathbf{w}_i^*)^2 + \beta|\mathbf{w}_i^*|.$$

L^1 Parameter Regularization (cont'd)

For each dimension i, the optimal solution is of the form

$$\mathbf{w}_i = \operatorname{sign}(\mathbf{w}_i^*) \max(|\mathbf{w}_i^*| - \frac{\beta}{\gamma_i}, 0).$$

L^2 and L^1 Regularization

- Regularization is any component of the model, training process or prediction procedure which is included to account for limitations of the training data, including its finiteness.
- Most classical regularization approaches are based on limiting the capacity of models, such as neural networks, linear regression, or logistic regression, by adding a parameter norm penalty $\mathbf{\Omega}(\theta)$ to the objective function J. We denote the regularized objective function by $\tilde{J}$:

$$\tilde{J}(\theta; \mathbf{X}, \mathbf{y}) = J(\theta; \mathbf{X}, \mathbf{y}) + \alpha \mathbf{\Omega}(\theta)$$

- For L^2 regression, $\mathbf{\Omega}(\theta) = \frac{1}{2}|\mathbf{w}||_2^2$
- For L^1 regression, $\mathbf{\Omega}(\theta) = |\mathbf{w}||_1$

Remark

L^1 regularization prefers sparser solutions i.e it prefers some parameters to have an optimal of zero

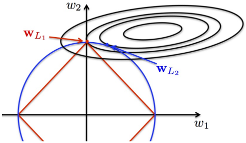


Figure credit: http://g2pi.tsc.uc3m.es/en/Primal-sparse-SVM

- We can also constrain the norm to be smaller than some value, rather than imposing a penalty on it. This is a case of constrained optimization.
- Such constrained optimization can be solved by projecting the weight vector on the constraint space after every update. This can allow us to have a higher learning rate and train networks faster.
- In the case of least squares linear regression, if the system is underconstrained, the matrix $\mathbf{X}^\top\mathbf{X}$ w ill not be invertible. However, the matrix $\mathbf{X}^\top\mathbf{X} + \alpha\mathbf{I}$ will always be invertible

Outline

1 Part I: Review of Previous Lecture
- Summary of Feedforward Deep Networks

2 Regularization from a Bayesian Perspective

3 Classical Regularization: Parameter Norm Penalty
- L^2 Parameter Regularization
- L^1 Parameter Regularization

4 Regularizations that can be approximated by Penalty Regularizations
- Regularization via Injecting Noise
- Early Stopping

5 Other techniques for regularization
- Dataset Augmentation
- Adversarial Training
- Parameter Sharing
- Multi-Task Learning
- Bootstrap Aggregating (Bagging)
- Dropout

Regularization via Injecting Noise

- Neural networks can be sensitive to noisy inputs
- It is possible to use noise as part of a regularization strategy
- Under certain assumptions on the noise model and certain approximations, we can show such regularization strategies as implementing a penalty based regularization
- We can either add noise to the inputs during training or inject noise into the weights during training

Injecting noise at the inputs

- With each input presentation to the model, we also include a random perturbation, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \nu \mathbf{I})$
- We will analyze injecting noise at the inputs in the context of regression, where we are interested in learning a model $\hat{y}(\mathbf{x})$
- The cost function then becomes

$$\begin{aligned}\tilde{J}_{\mathrm{x}} &= \mathbb{E}_{p(\mathrm{x},y,\boldsymbol{\epsilon})}[(\hat{y}(\mathbf{x}+\boldsymbol{\epsilon}) - y)^2] \\ &= \mathbb{E}_{p(\mathrm{x},\boldsymbol{\epsilon})}[\hat{y}^2(\mathbf{x}+\boldsymbol{\epsilon})] - 2\mathbb{E}_{p(\mathrm{x},\boldsymbol{\epsilon})}[y\hat{y}(\mathbf{x}+\boldsymbol{\epsilon})] + \mathbb{E}_{p(y)}[y^2]\end{aligned}$$

- Assuming that the noise is small, we can model its effect using the Taylor series expansion

$$\hat{y}(\mathbf{x}+\boldsymbol{\epsilon}) = \hat{y}(\mathbf{x}) + \boldsymbol{\epsilon}^T \nabla_{\mathrm{x}} \hat{y}(\mathbf{x}) + \frac{1}{2}\boldsymbol{\epsilon}^T \nabla^2_{\mathrm{x}} \hat{y}(\mathbf{x})\boldsymbol{\epsilon} + \mathcal{O}(\boldsymbol{\epsilon}^3)$$

Injecting noise at the inputs

- Plugging the taylor series we get

$$\begin{aligned}
\tilde{J}_{\mathrm{x}} &\approx \mathbb{E}_{p(\mathrm{x},\boldsymbol{\epsilon})}\left[\left(\hat{y}(\mathbf{x}) + \boldsymbol{\epsilon}^T\nabla_{\mathrm{x}}\hat{y}(\mathbf{x}) + \frac{1}{2}\boldsymbol{\epsilon}^T\nabla_{\mathrm{x}}^2\hat{y}(\mathbf{x})\boldsymbol{\epsilon}\right)^2\right] \\
&- 2\mathbb{E}_{p(\mathrm{x},\boldsymbol{\epsilon})}\left[y\left(\hat{y}(\mathbf{x}+\boldsymbol{\epsilon}) = \hat{y}(\mathbf{x}) + \boldsymbol{\epsilon}^T\nabla_{\mathrm{x}}\hat{y}(\mathbf{x}) + \frac{1}{2}\boldsymbol{\epsilon}^T\nabla_{\mathrm{x}}^2\hat{y}(\mathbf{x})\boldsymbol{\epsilon}\right)\right] + \mathbb{E}_{p(y)}[y^2] \\
&= \mathbb{E}_{p(\mathrm{x},y)}[(\hat{y}(\mathbf{x}) - y)^2] + \mathbb{E}_{p(\mathrm{x},\boldsymbol{\epsilon})}\left[\hat{y}(\mathbf{x})\boldsymbol{\epsilon}^T\nabla_{\mathrm{x}}^2\hat{y}(\mathbf{x})\boldsymbol{\epsilon} + \left(\epsilon^T\nabla_{\mathrm{x}}\hat{y}(\mathbf{x})\right)^2 + \mathcal{O}(\boldsymbol{\epsilon}^3)\right] \\
&- \mathbb{E}_{p(\mathrm{x},y,\boldsymbol{\epsilon})}\left[y\boldsymbol{\epsilon}^T\nabla_{\mathrm{x}}^2\hat{y}(\mathbf{x})\boldsymbol{\epsilon}\right] \\
&= J + \nu\mathbb{E}_{p(\mathrm{x},y)}\left[(\hat{y}(\mathbf{x}) - y)\nabla_{\mathrm{x}}^2\hat{y}(\mathbf{x})\right] + \nu\mathbb{E}_{p(\mathrm{x},y)}\left[||\nabla_{\mathrm{x}}\hat{y}(\mathbf{x})||^2\right]
\end{aligned}$$

Injecting noise at the inputs

- If we minimize this objective function, by taking the functional gradient of $\hat{y}(x)$ and setting it to zero, we get

$$\hat{y}(\mathbf{x}) = \mathbb{E}_{p(y|\mathrm{x})}[y] + \mathcal{O}(\nu)$$

- This implies that $\nu\mathbb{E}_{p(\mathrm{x},y)}\left[(\hat{y}(\mathbf{x}) - y)\nabla_{\mathrm{x}}^2\hat{y}(x)\right]$ reduces to $\mathcal{O}(\nu^2)$
- Therefore,

$$\tilde{J}_{\mathrm{x}} \approx J + \nu\mathbb{E}_{p(\mathrm{x},y)}\left[\|\nabla_{\mathrm{x}}\hat{y}(\mathbf{x})\|^2\right] + \mathcal{O}(\nu^2)$$

- This regularization term has the effect of penalizing large gradients of the function $\hat{y}(\mathbf{x})$. That is, it has the effect of reducing the sensitivity of the output of the network with respect to small variations in its input $\mathbf{x}$.
- We can interpret this as attempting to build in some local robustness into the model and thereby promote generalization. We note also that for linear networks, this regularization term reduces to simple weight decay

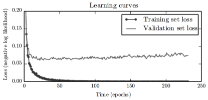
Injecting noise at the Weights

- Through similar analysis, we can approximate the cost function in the case where we add noise $\boldsymbol{\epsilon}_{\mathrm{w}} \sim \mathcal{N}(\mathbf{0}, \eta \mathbf{I})$ at the weights by the following

$$\tilde{J}_{\mathrm{w}} \approx J + \eta \mathbb{E}_{p(\mathrm{x}, y)} \left[||\nabla_{\mathrm{w}} \hat{y}(x)||^2 \right] + \mathcal{O}(\eta^2)$$

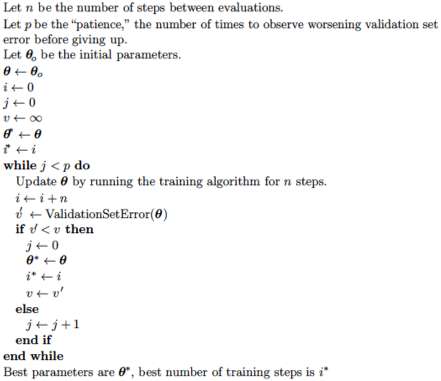
- This form of regularization encourages the parameters to go to regions of parameter space where small perturbations of the weights have a relatively small influence on the output.
- In other words, it pushes the model into regions where the model is relatively insensitive to small variations in the weights.

Early Stopping



- When training large models with large enough capacity, we often observe that training error decreases steadily over time, but validation set error begins to rise again.
- Instead of running our optimization algorithm until we reach a (local) minimum of validation error, we run it until the error on the validation set has not improved for some amount of time.
- Every time the error on the validation set improves, we store a copy of the model parameters.

Early Stopping



```
Let $n$ be the number of steps between evaluations.
Let $p$ be the "patience," the number of times to observe worsening validation set error before giving up.
Let $\boldsymbol{\theta}_o$ be the initial parameters.
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_o$
$i \leftarrow 0$
$j \leftarrow 0$
$v \leftarrow \infty$
$\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}$
$i^* \leftarrow i$
while $j < p$ do
  Update $\boldsymbol{\theta}$ by running the training algorithm for $n$ steps.
  $i \leftarrow i + n$
  $v' \leftarrow \text{ValidationSetError}(\boldsymbol{\theta})$
  if $v' < v$ then
    $j \leftarrow 0$
    $\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}$
    $i^* \leftarrow i$
    $v \leftarrow v'$
  else
    $j \leftarrow j + 1$
  end if
end while
Best parameters are $\boldsymbol{\theta}^*$, best number of training steps is $i^*$
```

Early Stopping

- Consider the quadratic approximation of the cost function in the neighbourhood of the empirically optimal value of the weights $\mathbf{w}^*$

$$\hat{J}(\mathbf{w}) = J(\mathbf{w}^*) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^T \mathbf{H}(\mathbf{w} - \mathbf{w}^*)$$

- Thus,

$$\nabla_{\mathrm{w}} \hat{J}(\mathbf{w}) = \mathbf{H}(\mathbf{w} - \mathbf{w}^*)$$

- For simplicity, we assume $\mathbf{w}^{(0)} = \mathbf{0}$
- Then, the gradient descent updates can be expressed as

$$\mathbf{w}^{(\tau)} = \mathbf{w}^{(\tau-1)} - \eta \mathbf{H}(\mathbf{w}^{(\tau-1)} - \mathbf{w}^*)$$
$$\mathbf{w}^{(\tau)} - \mathbf{w}^* = (\mathbf{I} - \eta \mathbf{H})(\mathbf{w}^{(\tau-1)} - \mathbf{w}^*)$$

Early Stopping

- With $\mathbf{H} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top$, we rewrite the above as

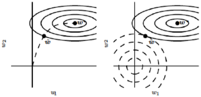
$$\mathbf{Q}^\top(\mathbf{w}^{(\tau)} - \mathbf{w}^*) = (\mathbf{I} - \eta\mathbf{\Lambda})\mathbf{Q}^\top(\mathbf{w}^{(\tau-1)} - \mathbf{w}^*)$$

- Assuming $\mathbf{w}^{(0)} = \mathbf{0}$

$$\mathbf{Q}^\top\mathbf{w}^{(\tau)} = (\mathbf{I} - (\mathbf{I} - \eta\mathbf{\Lambda})^\tau)\,\mathbf{Q}^\top\mathbf{w}^*$$

- The closed form solution of L^2 regularization can be written as

$$\mathbf{Q}^\top\tilde{w} = \left(\mathbf{I} - (\mathbf{I} + \alpha\mathbf{\Lambda})^{-1}\alpha\right)\mathbf{Q}^\top\mathbf{w}^*$$



Outline

1. Part I: Review of Previous Lecture
   - Summary of Feedforward Deep Networks
2. Regularization from a Bayesian Perspective
3. Classical Regularization: Parameter Norm Penalty
   - L^2 Parameter Regularization
   - L^1 Parameter Regularization
4. Regularizations that can be approximated by Penalty Regularizations
   - Regularization via Injecting Noise
   - Early Stopping
5. Other techniques for regularization
   - Dataset Augmentation
   - Adversarial Training
   - Parameter Sharing
   - Multi-Task Learning
   - Bootstrap Aggregating (Bagging)
   - Dropout

Dataset Augmentation

- More complex models require more training data and training data is always finite
- It may not always be possible to have large datasets
- For certain tasks, especially image classification, it may be possible to generate fake data
- For example, operations like translating the training images a few pixels in each direction can often greatly improve generalization
- One must be careful not to apply transformations that would change the correct class. For example, optical character recognition tasks require recognizing the difference between 'b' and 'd' and the difference between '6' and '9', so horizontal flips and 180^o rotations are not appropriate ways of augmenting datasets for these tasks.

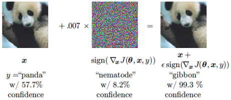
Adversarial Training

- Even neural networks that perform at human level accuracy have a nearly 100% error rate on examples that are intentionally constructed by using an optimization procedure to search for an input $\mathbf{x}'$ near a data point $\mathbf{x}$ such that the model output is very different at $\mathbf{x}'$. In many case, $\mathbf{x}'$ can be so similar to x that a human observer cannot tell the difference between the original example and the adversarial example.
- Neural networks are built out of primarily linear building blocks. In some experiments the overall function they implement proves to be highly linear as a result.
- To understand the implications of it, let's consider a linear mapping,

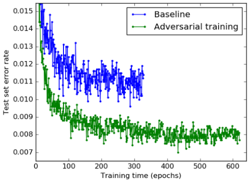
$$\hat{y} = \mathbf{w}^\top \mathbf{x}$$

- If we add a small $\boldsymbol{\epsilon}$ to the input in the linear case we get $\hat{y} = \mathbf{w}^\top \mathbf{x} + \mathbf{w}^\top \boldsymbol{\epsilon}$. We can maximize this increase by assigning $\boldsymbol{\epsilon} = \epsilon \ \text{sign}(\mathbf{w})$. When $\mathbf{w}$ is high dimensional, this can result in a very large change in the output.

Adversarial Training


$+ .007 \times$
$=$
$\boldsymbol{x}$
y ="panda"
w/ 57.7%
confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
w/ 8.2%
confidence
$\boldsymbol{x} +$
$\epsilon\, \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
w/ 99.3 %
confidence

One can reduce the error rate on the original i.i.d. test set via adversarial training—training on adversarially perturbed examples from the training set



Parameter Sharing

- Another way to regularize or reduce the complexity of a deep model is through parameter sharing.
- Various components of the model are made to share a unique set of parameters
- The most popular and extensive use of parameter sharing occurs in convolutional neural networks (CNNs) (To be covered later in class) applied to computer vision

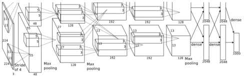
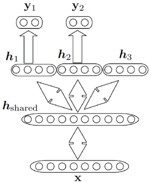


Figure Credit: Krizhevsky et al., NIPS 2012

Multi-Task Learning

- Parameter sharing restricts different components of the model to share parameters for a particular task. On the other hand, in multi-task learning, part of the model is shared across various tasks.
- Biological vision systems share initial processing across all tasks as well
- Assuming that sharing is justified, when part of a model is shared across tasks, that part of the model is more constrained towards "good "(in the context of generalization) values.

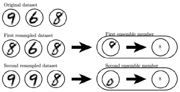


Bootstrap Aggregating (Bagging)

- Bagging (short for bootstrap aggregating) is a technique for reducing generalization error by combining several models
- Consider for example a set of k regression models, eaxh of which makes an error, ϵ_i on each example. Let $\mathbb{E}(\epsilon_i) = 0, \mathbb{E}(\epsilon_i^2) = \nu, \mathbb{E}(\epsilon_i \epsilon_j) = c$. Then

$$\mathbb{E}\left[\left(\frac{1}{k}\sum_i \epsilon_i\right)^2\right] = \frac{1}{k}\nu + \frac{k-1}{k}c$$

- Bagging involves constructing the k models by training them on k different datasets obtained by the process of bootstrapping. Bootstrapping involves uniformly sampling from the training dataset with replacement.



Dropout

- Dropout can be thought of as a method of making bagging practical for large neural networks.
- Dropout trains the ensemble consisting of all sub-networks that can be formed by removing units from an underlying base network.
- Dropout can be more effective than other standard computationally inexpensive regularizers, such as weight decay