Deep Learning Book

Chapter 7
Regularization for Deep Learning

Botian Shi
botianshi@bit.edu.cn

March 7, 2017

You can download the LaTeX source code of this file from Here.

Generalization and Strategy

- How to make an algorithm that will perform well not just on the training data, but also on new inputs?
- Many strategies designed to reduce the test error, possibly at the expense of increased training error.
- These strategies are known collectively as **regularization**.
- Many regularization algorithm have been developed.
- Developing more effective regularization strategies is one of the major research efforts in the field.
- In this chapter, we describe regularization in more detail, focusing on regularization strategies for deep models or models that may be used as building blocks to form deep models.

Generalization and Strategy

- How to make an algorithm that will perform well not just on the training data, but also on new inputs?
- Many strategies designed to reduce the test error, possibly at the expense of increased training error.
- These strategies are known collectively as **regularization**.
- Many regularization algorithm have been developed.
- Developing more effective regularization strategies is one of the major research efforts in the field.
- In this chapter, we describe regularization in more detail, focusing on regularization strategies for deep models or models that may be used as building blocks to form deep models.

Generalization and Strategy

- How to make an algorithm that will perform well not just on the training data, but also on new inputs?
- Many strategies designed to reduce the test error, possibly at the expense of increased training error.
- These strategies are known collectively as **regularization**.
- Many regularization algorithm have been developed.
- Developing more effective regularization strategies is one of the major research efforts in the field.
- In this chapter, we describe regularization in more detail, focusing on regularization strategies for deep models or models that may be used as building blocks to form deep models.

Generalization and Strategy

- How to make an algorithm that will perform well not just on the training data, but also on new inputs?
- Many strategies designed to reduce the test error, possibly at the expense of increased training error.
- These strategies are known collectively as **regularization**.
- Many regularization algorithm have been developed.
- Developing more effective regularization strategies is one of the major research efforts in the field.
- In this chapter, we describe regularization in more detail, focusing on regularization strategies for deep models or models that may be used as building blocks to form deep models.

Generalization and Strategy

- How to make an algorithm that will perform well not just on the training data, but also on new inputs?
- Many strategies designed to reduce the test error, possibly at the expense of increased training error.
- These strategies are known collectively as **regularization**.
- Many regularization algorithm have been developed.
- Developing more effective regularization strategies is one of the major research efforts in the field.
- In this chapter, we describe regularization in more detail, focusing on regularization strategies for deep models or models that may be used as building blocks to form deep models.

Generalization and Strategy

- There are many regularization strategies.
  1. Put extra constrains on a machine learning model. (Adding restrictions on the parameter values.)
  2. Add extra terms in the objective function that can be thought of as corresponding to a soft constraint on the parameter values.
- If chosen carefully, these extra constraints and penalties can lead to improved performance on the test set.
- Sometimes these constraints and penalties are designed to
  1. **encode** specific kinds of **prior knowledge**.
  2. Express a generic preference for a simpler model class in order to promote generalization.
  3. make an under-determined problem determined. (Provide more information)
- Other forms of regularization, known as ensemble methods, combine multiple hypotheses that explain the training data.

Generalization and Strategy

- There are many regularization strategies.
  1. Put extra constrains on a machine learning model. (Adding restrictions on the parameter values.)
  2. Add extra terms in the objective function that can be thought of as corresponding to a soft constraint on the parameter values.
- If chosen carefully, these extra constraints and penalties can lead to improved performance on the test set.
- Sometimes these constraints and penalties are designed to
  1. **encode** specific kinds of **prior knowledge**.
  2. Express a generic preference for a simpler model class in order to promote generalization.
  3. make an under-determined problem determined. (Provide more information)
- Other forms of regularization, known as ensemble methods, combine multiple hypotheses that explain the training data.

Generalization and Strategy

- There are many regularization strategies.
  1. Put extra constrains on a machine learning model. (Adding restrictions on the parameter values.)
  2. Add extra terms in the objective function that can be thought of as corresponding to a soft constraint on the parameter values.
- If chosen carefully, these extra constraints and penalties can lead to improved performance on the test set.
- Sometimes these constraints and penalties are designed to
  1. **encode** specific kinds of **prior knowledge**.
  2. Express a generic preference for a simpler model class in order to promote generalization.
  3. make an under-determined problem determined. (Provide more information)
- Other forms of regularization, known as ensemble methods, combine multiple hypotheses that explain the training data.

Generalization and Strategy

- There are many regularization strategies.
  1. Put extra constrains on a machine learning model. (Adding restrictions on the parameter values.)
  2. Add extra terms in the objective function that can be thought of as corresponding to a soft constraint on the parameter values.
- If chosen carefully, these extra constraints and penalties can lead to improved performance on the test set.
- Sometimes these constraints and penalties are designed to
  1. **encode** specific kinds of **prior knowledge**.
  2. Express a generic preference for a simpler model class in order to promote generalization.
  3. make an under-determined problem determined. (Provide more information)
- Other forms of regularization, known as ensemble methods, combine multiple hypotheses that explain the training data.

Generalization and Strategy

- There are many regularization strategies.
  1. Put extra constrains on a machine learning model. (Adding restrictions on the parameter values.)
  2. Add extra terms in the objective function that can be thought of as corresponding to a soft constraint on the parameter values.
- If chosen carefully, these extra constraints and penalties can lead to improved performance on the test set.
- Sometimes these constraints and penalties are designed to
  1. **encode** specific kinds of **prior knowledge**.
  2. Express a generic preference for a simpler model class in order to promote generalization.
  3. make an under-determined problem determined. (Provide more information)
- Other forms of regularization, known as ensemble methods, combine multiple hypotheses that explain the training data.

Generalization and Strategy

- There are many regularization strategies.
  1. Put extra constrains on a machine learning model. (Adding restrictions on the parameter values.)
  2. Add extra terms in the objective function that can be thought of as corresponding to a soft constraint on the parameter values.
- If chosen carefully, these extra constraints and penalties can lead to improved performance on the test set.
- Sometimes these constraints and penalties are designed to
  1. **encode** specific kinds of **prior knowledge**.
  2. Express a generic preference for a simpler model class in order to promote generalization.
  3. make an under-determined problem determined. (Provide more information)
- Other forms of regularization, known as ensemble methods, combine multiple hypotheses that explain the training data.

Generalization and Strategy

- Principle: Treading increased bias for reduced variance.
- An effective regularizer is one that makes a profitable trade, **reducing variance** significantly while not overly **increasing the bias**.
- In practice, **an overly complex model family does not necessarily include the target function or the true data generating process, or even a close approximation**.
- We almost never have access to the true data generating process so we can never know for sure **if the model family being estimated includes the generating process or not**.

Generalization and Strategy

- Principle: Treading increased bias for reduced variance.
- An effective regularizer is one that makes a profitable trade, **reducing variance** significantly while not overly **increasing the bias**.
- In practice, **an overly complex model family does not necessarily include the target function or the true data generating process, or even a close approximation**.
- We almost never have access to the true data generating process so we can never know for sure **if the model family being estimated includes the generating process or not**.

Generalization and Strategy

- Principle: Treading increased bias for reduced variance.
- An effective regularizer is one that makes a profitable trade, **reducing variance** significantly while not overly **increasing the bias**.
- In practice, **an overly complex model family does not necessarily include the target function or the true data generating process, or even a close approximation**.
- We almost never have access to the true data generating process so we can never know for sure **if the model family being estimated includes the generating process or not**.

Generalization and Strategy

- Principle: Treading increased bias for reduced variance.
- An effective regularizer is one that makes a profitable trade, **reducing variance** significantly while not overly **increasing the bias**.
- In practice, **an overly complex model family does not necessarily include the target function or the true data generating process, or even a close approximation**.
- We almost never have access to the true data generating process so we can never know for sure **if the model family being estimated includes the generating process or not**.

Generalization and Strategy

- However, most applications of deep learning algorithms are to domains where the true data generating process is almost certainly outside the model family.
- Deep learning algorithms are typically applied to **extremely complicated domains** such as images, audio sequences and text, for which the true generation process essentially involves **simulating the entire universe**.
- To some extent, we are always trying to fit a square peg(the data generating process) into a round hole (our model family) 『持方枘 (ruì) 而欲内圆凿』.
- What this means is that controlling the complexity of the model is not a simple matter of finding the model of the right size, with the **right number of parameters**.
- Insteamd, we might find that the best fitting model is a large model that has been regularized appropriately.
- We now review several strategies for how to create such a large, deep, regularized model.

Generalization and Strategy

- However, most applications of deep learning algorithms are to domains where the true data generating process is almost certainly outside the model family.
- Deep learning algorithms are typically applied to **extremely complicated domains** such as images, audio sequences and text, for which the true generation process essentially involves **simulating the entire universe**.
- To some extent, we are always trying to fit a square peg(the data generating process) into a round hole (our model family) 『持方枘 (ruì) 而欲内圆凿』.
- What this means is that controlling the complexity of the model is not a simple matter of finding the model of the right size, with the **right number of parameters**.
- Insteamd, we might find that the best fitting model is a large model that has been regularized appropriately.
- We now review several strategies for how to create such a large, deep, regularized model.

Generalization and Strategy

- However, most applications of deep learning algorithms are to domains where the true data generating process is almost certainly outside the model family.
- Deep learning algorithms are typically applied to **extremely complicated domains** such as images, audio sequences and text, for which the true generation process essentially involves **simulating the entire universe**.
- To some extent, we are always trying to fit a square peg(the data generating process) into a round hole (our model family) 『持方枘 (ruì) 而欲内圆凿』.
- What this means is that controlling the complexity of the model is not a simple matter of finding the model of the right size, with the **right number of parameters**.
- Insteamd, we might find that the best fitting model is a large model that has been regularized appropriately.
- We now review several strategies for how to create such a large, deep, regularized model.

Generalization and Strategy

- However, most applications of deep learning algorithms are to domains where the true data generating process is almost certainly outside the model family.
- Deep learning algorithms are typically applied to **extremely complicated domains** such as images, audio sequences and text, for which the true generation process essentially involves **simulating the entire universe**.
- To some extent, we are always trying to fit a square peg(the data generating process) into a round hole (our model family) 『持方枘 (ruì) 而欲内圆凿』.
- What this means is that controlling the complexity of the model is not a simple matter of finding the model of the right size, with the **right number of parameters**.
- Insteamd, we might find that the best fitting model is a large model that has been regularized appropriately.
- We now review several strategies for how to create such a large, deep, regularized model.

Generalization and Strategy

- However, most applications of deep learning algorithms are to domains where the true data generating process is almost certainly outside the model family.
- Deep learning algorithms are typically applied to **extremely complicated domains** such as images, audio sequences and text, for which the true generation process essentially involves **simulating the entire universe**.
- To some extent, we are always trying to fit a square peg(the data generating process) into a round hole (our model family) **『持方枘 (ruì) 而欲内圆凿』**.
- What this means is that controlling the complexity of the model is not a simple matter of finding the model of the right size, with the **right number of parameters**.
- Insteamd, we might find that the best fitting model is a large model that has been regularized appropriately.
- We now review several strategies for how to create such a large, deep, regularized model.

Generalization and Strategy

- However, most applications of deep learning algorithms are to domains where the true data generating process is almost certainly outside the model family.
- Deep learning algorithms are typically applied to **extremely complicated domains** such as images, audio sequences and text, for which the true generation process essentially involves **simulating the entire universe**.
- To some extent, we are always trying to fit a square peg(the data generating process) into a round hole (our model family) **『持方枘** (ruì) **而欲内圆凿』**.
- What this means is that controlling the complexity of the model is not a simple matter of finding the model of the right size, with the **right number of parameters**.
- Insteamd, we might find that the best fitting model is a large model that has been regularized appropriately.
- We now review several strategies for how to create such a large, deep, regularized model.

Parameter Norm Penalties

- Regularization has been used for decades prior to the advent of deep learning.
- Linear models allow simple straightforward and effective regularization strategies.
- Most approaches are based on limiting the capacity of models by adding a **parameter norm penalty** $\Omega(\theta)$ to the objective function J:

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

  where $\alpha \in [0, +\infty)$ weights the relative contribution of the norm penalty term.
- Setting α to 0 results in no regularization. Larger values of α correspond to more regularization.
- Optimize both J and norm
- Different Ω has different result.

Parameter Norm Penalties

- Regularization has been used for decades prior to the advent of deep learning.
- Linear models allow simple straightforward and effective regularization strategies.
- Most approaches are based on limiting the capacity of models by adding a **parameter norm penalty** $\Omega(\theta)$ to the objective function J:

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

  where $\alpha \in [0, +\infty)$ weights the relative contribution of the norm penalty term.
- Setting α to 0 results in no regularization. Larger values of α correspond to more regularization.
- Optimize both J and norm
- Different Ω has different result.

Parameter Norm Penalties

- Regularization has been used for decades prior to the advent of deep learning.
- Linear models allow simple straightforward and effective regularization strategies.
- Most approaches are based on limiting the capacity of models by adding a **parameter norm penalty** $\Omega(\theta)$ to the objective function J:

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

  where $\alpha \in [0, +\infty)$ weights the relative contribution of the norm penalty term.
- Setting α to 0 results in no regularization. Larger values of α correspond to more regularization.
- Optimize both J and norm
- Different Ω has different result.

Parameter Norm Penalties

- Regularization has been used for decades prior to the advent of deep learning.
- Linear models allow simple straightforward and effective regularization strategies.
- Most approaches are based on limiting the capacity of models by adding a **parameter norm penalty** $\Omega(\theta)$ to the objective function J:

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

  where $\alpha \in [0, +\infty)$ weights the relative contribution of the norm penalty term.
- Setting α to 0 results in no regularization. Larger values of α correspond to more regularization.
- Optimize both J and norm
- Different Ω has different result.

Parameter Norm Penalties

- Regularization has been used for decades prior to the advent of deep learning.
- Linear models allow simple straightforward and effective regularization strategies.
- Most approaches are based on limiting the capacity of models by adding a **parameter norm penalty** $\Omega(\theta)$ to the objective function J:

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

  where $\alpha \in [0, +\infty)$ weights the relative contribution of the norm penalty term.
- Setting α to 0 results in no regularization. Larger values of α correspond to more regularization.
- Optimize both J and norm
- Different Ω has different result.

Parameter Norm Penalties

- Regularization has been used for decades prior to the advent of deep learning.
- Linear models allow simple straightforward and effective regularization strategies.
- Most approaches are based on limiting the capacity of models by adding a **parameter norm penalty** $\Omega(\theta)$ to the objective function J:

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

  where $\alpha \in [0, +\infty)$ weights the relative contribution of the norm penalty term.
- Setting α to 0 results in no regularization. Larger values of α correspond to more regularization.
- Optimize both J and norm
- Different Ω has different result.

Parameter Norm Penalties

- We penalize **only the weights** of the affine transformation at each layer and leaves the biases unregularized.
- We do not induce too much variance by leaving the biases unregularized.
- Regularizing the bias parameters can introduce a significant amount of under-fitting.
- We therefore use the vector $\boldsymbol{w}$ to indicate all of the weights that should be affected by a norm penalty, while the vector $\boldsymbol{\theta}$ denotes all of the parameters, including both $\boldsymbol{w}$ and the unregularized parameters.
- Sometime we use a separate penalty with a different α coefficient for each layer.
- But it can be expensive to search for the correct value of multiple hyper-parameters, it is still reasonable to use the same weight decay at all layers just to reduce the size of search space.

Parameter Norm Penalties

- We penalize **only the weights** of the affine transformation at each layer and leaves the biases unregularized.
- We do not induce too much variance by leaving the biases unregularized.
- Regularizing the bias parameters can introduce a significant amount of under-fitting.
- We therefore use the vector $\boldsymbol{w}$ to indicate all of the weights that should be affected by a norm penalty, while the vector $\boldsymbol{\theta}$ denotes all of the parameters, including both $\boldsymbol{w}$ and the unregularized parameters.
- Sometime we use a separate penalty with a different α coefficient for each layer.
- But it can be expensive to search for the correct value of multiple hyper-parameters, it is still reasonable to use the same weight decay at all layers just to reduce the size of search space.

Parameter Norm Penalties

- We penalize **only the weights** of the affine transformation at each layer and leaves the biases unregularized.
- We do not induce too much variance by leaving the biases unregularized.
- Regularizing the bias parameters can introduce a significant amount of under-fitting.
- We therefore use the vector $\boldsymbol{w}$ to indicate all of the weights that should be affected by a norm penalty, while the vector $\boldsymbol{\theta}$ denotes all of the parameters, including both $\boldsymbol{w}$ and the unregularized parameters.
- Sometime we use a separate penalty with a different α coefficient for each layer.
- But it can be expensive to search for the correct value of multiple hyper-parameters, it is still reasonable to use the same weight decay at all layers just to reduce the size of search space.

Parameter Norm Penalties

- We penalize **only the weights** of the affine transformation at each layer and leaves the biases unregularized.
- We do not induce too much variance by leaving the biases unregularized.
- Regularizing the bias parameters can introduce a significant amount of under-fitting.
- We therefore use the vector $\boldsymbol{w}$ to indicate all of the weights that should be affected by a norm penalty, while the vector $\boldsymbol{\theta}$ denotes all of the parameters, including both $\boldsymbol{w}$ and the unregularized parameters.
- Sometime we use a separate penalty with a different α coefficient for each layer.
- But it can be expensive to search for the correct value of multiple hyper-parameters, it is still reasonable to use the same weight decay at all layers just to reduce the size of search space.

Parameter Norm Penalties

- We penalize **only the weights** of the affine transformation at each layer and leaves the biases unregularized.
- We do not induce too much variance by leaving the biases unregularized.
- Regularizing the bias parameters can introduce a significant amount of under-fitting.
- We therefore use the vector $\boldsymbol{w}$ to indicate all of the weights that should be affected by a norm penalty, while the vector $\boldsymbol{\theta}$ denotes all of the parameters, including both $\boldsymbol{w}$ and the unregularized parameters.
- Sometime we use a separate penalty with a different α coefficient for each layer.
- But it can be expensive to search for the correct value of multiple hyper-parameters, it is still reasonable to use the same weight decay at all layers just to reduce the size of search space.

L^2 Parameter Regularization

- The L^2 norm penalty commonly known as *weight decay*.

$$\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|w\|_2^2$$

- This regularization strategy drives the weights closer to the origin. (as well as *ridge regression* or *Tikhonov regularization*)
- We can gain some insight into the behavior of weight decay regularization. (assume no bias for simplification)

$$\tilde{J}(w; X, y) = \frac{\alpha}{2} w^T w + J(w; X, y)$$

$$\nabla_w \tilde{J}(w; X, y) = \alpha w + \nabla_w J(w; X, y)$$

- The update

$$w \leftarrow w - \epsilon(\alpha w + \nabla_w J(w; X, y))$$

$$w \leftarrow (1 - \epsilon\alpha) w - \epsilon \nabla_w J(w; X, y)$$

- Shrink the weight vector by a constant factor on each step.
- What happens over the entire course of training?

L^2 Parameter Regularization

- The L^2 norm penalty commonly known as *weight decay*.

$$\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|w\|_2^2$$

- This regularization strategy drives the weights closer to the origin. (as well as *ridge regression* or *Tikhonov regularization*)
- We can gain some insight into the behavior of weight decay regularization. (assume no bias for simplification)

$$\tilde{J}(w; X, y) = \frac{\alpha}{2} w^T w + J(w; X, y)$$

$$\nabla_w \tilde{J}(w; X, y) = \alpha w + \nabla_w J(w; X, y)$$

- The update

$$w \leftarrow w - \epsilon(\alpha w + \nabla_w J(w; X, y))$$

$$w \leftarrow (1 - \epsilon\alpha) w - \epsilon \nabla_w J(w; X, y)$$

- Shrink the weight vector by a constant factor on each step.
- What happens over the entire course of training?

L^2 Parameter Regularization

- The L^2 norm penalty commonly known as *weight decay*.

$$\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|w\|_2^2$$

- This regularization strategy drives the weights closer to the origin. (as well as *ridge regression* or *Tikhonov regularization*)
- We can gain some insight into the behavior of weight decay regularization. (assume no bias for simplification)

$$\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \frac{\alpha}{2}\boldsymbol{w}^T\boldsymbol{w} + J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$$

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

- The update

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

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

- Shrink the weight vector by a constant factor on each step.
- What happens over the entire course of training?

L^2 Parameter Regularization

- The L^2 norm penalty commonly known as *weight decay*.

$$\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|w\|_2^2$$

- This regularization strategy drives the weights closer to the origin. (as well as *ridge regression* or *Tikhonov regularization*)
- We can gain some insight into the behavior of weight decay regularization. (assume no bias for simplification)

$$\tilde{J}(w; X, y) = \frac{\alpha}{2} w^T w + J(w; X, y)$$

$$\nabla_w \tilde{J}(w; X, y) = \alpha w + \nabla_w J(w; X, y)$$

- The update

$$w \leftarrow w - \epsilon(\alpha w + \nabla_w J(w; X, y))$$

$$w \leftarrow (1 - \epsilon\alpha) w - \epsilon \nabla_w J(w; X, y)$$

- Shrink the weight vector by a constant factor on each step.
- What happens over the entire course of training?

Recall: Quadratic Approximation

- In mathematics, approximation theory is concerned with how functions can best be approximated with simpler functions.
- **local linear approximation** and **taylor expansion**
    1. For example, when the independent variable of function $y = x^3$ changes, which is Δx, the variation of y is
    $$\Delta y = (x + \Delta x)^3 - x^3 = 3x^2 \Delta x + 3x(\Delta x)^2 + (\Delta x)^3$$
    2. When $\Delta x \to 0$, omit last two terms: $\Delta y = 3x^2 \Delta x$
    3. In general:
    $$\Delta y = f(x_0 + \Delta x) - f(x_0) \approx f'(x_0) \times \Delta x$$
    $$\Delta y = f(x) - f(x_0),\ \Delta x = x - x_0$$
    $$f(x) - f(x_0) = f'(x_0) \times (x - x_0)$$
    $$f(x) = f(x_0) + f'(x_0)(x - x_0)$$
    4. In order to improve the precision, we can use second-order approximation, which is the second-order Taylor series expansion.
    $$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2$$

Recall: Quadratic Approximation

- In mathematics, approximation theory is concerned with how functions can best be approximated with simpler functions.
- **local linear approximation** and **taylor expansion**
  1. For example, when the independent variable of function $y = x^3$ changes, which is Δx, the variation of y is

  $$\Delta y = (x + \Delta x)^3 - x^3 = 3x^2 \Delta x + 3x(\Delta x)^2 + (\Delta x)^3$$

  2. When $\Delta x \to 0$, omit last two terms: $\Delta y = 3x^2 \Delta x$
  3. In general:

  $$\Delta y = f(x_0 + \Delta x) - f(x_0) \approx f'(x_0) \times \Delta x$$

  $$\Delta y = f(x) - f(x_0),\ \Delta x = x - x_0$$

  $$f(x) - f(x_0) = f'(x_0) \times (x - x_0)$$

  $$f(x) = f(x_0) + f'(x_0)(x - x_0)$$

  4. In order to improve the precision, we can use second-order approximation, which is the second-order Taylor series expansion.

  $$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2$$

Recall: Quadratic Approximation

- In mathematics, approximation theory is concerned with how functions can best be approximated with simpler functions.
- **local linear approximation** and **taylor expansion**
  1. For example, when the independent variable of function $y = x^3$ changes, which is Δx, the variation of y is

$$\Delta y = (x + \Delta x)^3 - x^3 = 3x^2 \Delta x + 3x(\Delta x)^2 + (\Delta x)^3$$

  2. When $\Delta x \to 0$, omit last two terms: $\Delta y = 3x^2 \Delta x$
  3. In general:

$$\Delta y = f(x_0 + \Delta x) - f(x_0) \approx f'(x_0) \times \Delta x$$

$$\Delta y = f(x) - f(x_0),\ \Delta x = x - x_0$$

$$f(x) - f(x_0) = f'(x_0) \times (x - x_0)$$

$$f(x) = f(x_0) + f'(x_0)(x - x_0)$$

  4. In order to improve the precision, we can use second-order approximation, which is the second-order Taylor series expansion.

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2$$

Recall: Quadratic Approximation

- In mathematics, approximation theory is concerned with how functions can best be approximated with simpler functions.
- **local linear approximation** and **taylor expansion**
  1. For example, when the independent variable of function $y = x^3$ changes, which is Δx, the variation of y is

$$\Delta y = (x + \Delta x)^3 - x^3 = 3x^2 \Delta x + 3x(\Delta x)^2 + (\Delta x)^3$$

  2. When $\Delta x \to 0$, omit last two terms: $\Delta y = 3x^2 \Delta x$
  3. In general:

$$\Delta y = f(x_0 + \Delta x) - f(x_0) \approx f'(x_0) \times \Delta x$$

$$\Delta y = f(x) - f(x_0),\ \Delta x = x - x_0$$

$$f(x) - f(x_0) = f'(x_0) \times (x - x_0)$$

$$f(x) = f(x_0) + f'(x_0)(x - x_0)$$

  4. In order to improve the precision, we can use second-order approximation, which is the second-order Taylor series expansion.

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2$$

Recall: Quadratic Approximation

- In mathematics, approximation theory is concerned with how functions can best be approximated with simpler functions.
- **local linear approximation** and **taylor expansion**
  1. For example, when the independent variable of function $y = x^3$ changes, which is Δx, the variation of y is

$$\Delta y = (x + \Delta x)^3 - x^3 = 3x^2 \Delta x + 3x(\Delta x)^2 + (\Delta x)^3$$

  2. When $\Delta x \to 0$, omit last two terms: $\Delta y = 3x^2 \Delta x$
  3. In general:

$$\Delta y = f(x_0 + \Delta x) - f(x_0) \approx f'(x_0) \times \Delta x$$

$$\Delta y = f(x) - f(x_0),\ \Delta x = x - x_0$$

$$f(x) - f(x_0) = f'(x_0) \times (x - x_0)$$

$$f(x) = f(x_0) + f'(x_0)(x - x_0)$$

  4. In order to improve the precision, we can use second-order approximation, which is the second-order Taylor series expansion.

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2$$

Recall: Quadratic Approximation

- In mathematics, approximation theory is concerned with how functions can best be approximated with simpler functions.
- **local linear approximation** and **taylor expansion**
  1. For example, when the independent variable of function $y = x^3$ changes, which is Δx, the variation of y is

$$\Delta y = (x + \Delta x)^3 - x^3 = 3x^2 \Delta x + 3x(\Delta x)^2 + (\Delta x)^3$$

  2. When $\Delta x \to 0$, omit last two terms: $\Delta y = 3x^2 \Delta x$
  3. In general:

$$\Delta y = f(x_0 + \Delta x) - f(x_0) \approx f'(x_0) \times \Delta x$$

$$\Delta y = f(x) - f(x_0),\ \Delta x = x - x_0$$

$$f(x) - f(x_0) = f'(x_0) \times (x - x_0)$$

$$f(x) = f(x_0) + f'(x_0)(x - x_0)$$

  4. In order to improve the precision, we can use second-order approximation, which is the second-order Taylor series expansion.

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2$$

Recall: Quadratic Approximation

- In mathematics, approximation theory is concerned with how functions can best be approximated with simpler functions.
- **local linear approximation** and **taylor expansion**
  1. For example, when the independent variable of function $y = x^3$ changes, which is Δx, the variation of y is
  $$\Delta y = (x + \Delta x)^3 - x^3 = 3x^2\Delta x + 3x(\Delta x)^2 + (\Delta x)^3$$
  2. When $\Delta x \to 0$, omit last two terms: $\Delta y = 3x^2\Delta x$
  3. In general:
  $$\Delta y = f(x_0 + \Delta x) - f(x_0) \approx f'(x_0) \times \Delta x$$
  $$\Delta y = f(x) - f(x_0),\ \Delta x = x - x_0$$
  $$f(x) - f(x_0) = f'(x_0) \times (x - x_0)$$
  $$f(x) = f(x_0) + f'(x_0)(x - x_0)$$
  4. In order to improve the precision, we can use second-order approximation, which is the second-order Taylor series expansion.
  $$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2$$

L^2 Parameter Regularization

- Let $w^* = \arg\min_w J(w)$ (unregularized training cost)
- Making a quadratic approximation to the objective function in the neighborhood of the value of the weights. (In DLBook, they used $\hat{J}(\theta)$, but here we use $\hat{J}(w)$ to explain easier)

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

- Where H is the Hessian matrix of J with respect to w evaluated at w^*.
- There is no first-order term in this quadratic approximation, because w^* is defined to be a minimum, where the gradient vanishes.
- The minimum of $\hat{J}$ occurs where its gradient

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

is equal to 0.

L^2 Parameter Regularization

- Let $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} J(\boldsymbol{w})$ (unregularized training cost)
- Making a quadratic approximation to the objective function in the neighborhood of the value of the weights. (In DLBook, they used $\hat{J}(\boldsymbol{\theta})$, but here we use $\hat{J}(\boldsymbol{w})$ to explain easier)

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

- Where $\mathbf{H}$ is the Hessian matrix of J with respect to $\boldsymbol{w}$ evaluated at $\boldsymbol{w}^*$.
- There is no first-order term in this quadratic approximation, because $\boldsymbol{w}^*$ is defined to be a minimum, where the gradient vanishes.
- The minimum of $\hat{J}$ occurs where its gradient

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

is equal to 0.

L^2 Parameter Regularization

- Let $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} J(\boldsymbol{w})$ (unregularized training cost)
- Making a quadratic approximation to the objective function in the neighborhood of the value of the weights. (In DLBook, they used $\hat{J}(\boldsymbol{\theta})$, but here we use $\hat{J}(\boldsymbol{w})$ to explain easier)

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

- Where $\mathbf{H}$ is the Hessian matrix of J with respect to $\boldsymbol{w}$ evaluated at $\boldsymbol{w}^*$.
- There is no first-order term in this quadratic approximation, because $\boldsymbol{w}^*$ is defined to be a minimum, where the gradient vanishes.
- The minimum of $\hat{J}$ occurs where its gradient

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

is equal to 0.

L^2 Parameter Regularization

- Let $w^* = \arg\min_w J(w)$ (unregularized training cost)
- Making a quadratic approximation to the objective function in the neighborhood of the value of the weights. (In DLBook, they used $\hat{J}(\boldsymbol{\theta})$, but here we use $\hat{J}(w)$ to explain easier)

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

- Where H is the Hessian matrix of J with respect to w evaluated at w^*.
- There is no first-order term in this quadratic approximation, because w^* is defined to be a minimum, where the gradient vanishes.
- The minimum of $\hat{J}$ occurs where its gradient

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

is equal to 0.

L^2 Parameter Regularization

- Let $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} J(\boldsymbol{w})$ (unregularized training cost)
- Making a quadratic approximation to the objective function in the neighborhood of the value of the weights. (In DLBook, they used $\hat{J}(\boldsymbol{\theta})$, but here we use $\hat{J}(\boldsymbol{w})$ to explain easier)

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

- Where H is the Hessian matrix of J with respect to $\boldsymbol{w}$ evaluated at $\boldsymbol{w}^*$.
- There is no first-order term in this quadratic approximation, because $\boldsymbol{w}^*$ is defined to be a minimum, where the gradient vanishes.
- The minimum of $\hat{J}$ occurs where its gradient

$$\nabla_{\boldsymbol{w}} \hat{J}(\boldsymbol{w}) = \mathsf{H}(\boldsymbol{w} - \boldsymbol{w}^*)$$

is equal to 0.

L^2 Parameter Regularization

- To study the effect of weight decay, we modify $\nabla_w \hat{J}(w) = \mathrm{H}(w - w^*)$ by adding the weight decay gradient.
- We can solve for the minimum of the regularized version of $\hat{J}$.
- We use the variable $\tilde{w}$ to represent the location of the minimum.

$$\alpha \tilde{w} + \mathrm{H}(\tilde{w} - w^*) = 0$$

$$(\mathrm{H} + \alpha \mathrm{I})\tilde{w} = \mathrm{H}w^*$$

$$\tilde{w} = \frac{\mathrm{H}w^*}{(\mathrm{H} + \alpha \mathrm{I})}$$

- As α approaches 0, the regularized solution $\tilde{w}$ approaches w^*.
- But what happens as α grows?

L^2 Parameter Regularization

- To study the effect of weight decay, we modify $\nabla_w \hat{J}(w) = \mathsf{H}(w - w^*)$ by adding the weight decay gradient.
- We can solve for the minimum of the regularized version of $\hat{J}$.
- We use the variable $\tilde{w}$ to represent the location of the minimum.

$$\alpha \tilde{w} + \mathsf{H}(\tilde{w} - w^*) = 0$$

$$(\mathsf{H} + \alpha \mathsf{I})\tilde{w} = \mathsf{H}w^*$$

$$\tilde{w} = \frac{\mathsf{H}w^*}{(\mathsf{H} + \alpha \mathsf{I})}$$

- As α approaches 0, the regularized solution $\tilde{w}$ approaches w^*.
- But what happens as α grows?

L^2 Parameter Regularization

- To study the effect of weight decay, we modify $\nabla_w \hat{J}(w) = \mathrm{H}(w - w^*)$ by adding the weight decay gradient.
- We can solve for the minimum of the regularized version of $\hat{J}$.
- We use the variable $\tilde{w}$ to represent the location of the minimum.

$$\alpha \tilde{w} + \mathrm{H}(\tilde{w} - w^*) = 0$$

$$(\mathrm{H} + \alpha \mathrm{I})\tilde{w} = \mathrm{H}w^*$$

$$\tilde{w} = \frac{\mathrm{H}w^*}{(\mathrm{H} + \alpha \mathrm{I})}$$

- As α approaches 0, the regularized solution $\tilde{w}$ approaches w^*.
- But what happens as α grows?

L^2 Parameter Regularization

- Because H is real and symmetric, we can decompose it into a diagonal matrix $\mathbf{\Lambda}$ and an orthonormal basis of eigenvectors, Q, such that $\mathsf{H} = Q\mathbf{\Lambda}Q^T$.
- Applying the decomposition $\tilde{w} = (\mathsf{H} + \alpha \mathbf{I})^{-1}\mathsf{H}w^*$

$$
\begin{aligned}
\tilde{w} &= (Q\mathbf{\Lambda}Q^T + \alpha I)^{-1} Q\mathbf{\Lambda}Q^T w^* && (1)\\
&= \left[Q(\mathbf{\Lambda} + \alpha I)Q^T\right]^{-1} Q\mathbf{\Lambda}Q^T w^* && (2)\\
&= Q(\mathbf{\Lambda} + \alpha I)^{-1}\mathbf{\Lambda}Q^T w^* && (3)\\
&= Q\frac{\mathbf{\Lambda}}{\mathbf{\Lambda} + \alpha I}Q^T w^* && (4)
\end{aligned}
$$

- We see that the effect of weight decay is to rescale w^* along the axes defined by the eigenvectors of H.
- Specifically, the component of w^* that is aligned with the i-th eigenvector of H is rescaled by a factor of $\frac{\lambda_i}{\lambda_i + \alpha}$

L^2 Parameter Regularization

- Because H is real and symmetric, we can decompose it into a diagonal matrix $\boldsymbol{\Lambda}$ and an orthonormal basis of eigenvectors, Q, such that $\mathsf{H} = Q\boldsymbol{\Lambda}Q^T$.
- Applying the decomposition $\tilde{w} = (\mathsf{H} + \alpha\mathsf{I})^{-1}\mathsf{H}w^*$

$$\begin{aligned}
\tilde{w} &= (Q\boldsymbol{\Lambda}Q^T + \alpha I)^{-1}Q\boldsymbol{\Lambda}Q^T w^* &(1)\\
&= \left[Q(\boldsymbol{\Lambda} + \alpha I)Q^T\right]^{-1} Q\boldsymbol{\Lambda}Q^T w^* &(2)\\
&= Q(\boldsymbol{\Lambda} + \alpha I)^{-1}\boldsymbol{\Lambda}Q^T w^* &(3)\\
&= Q\frac{\boldsymbol{\Lambda}}{\boldsymbol{\Lambda} + \alpha I}Q^T w^* &(4)
\end{aligned}$$

- We see that the effect of weight decay is to rescale w^* along the axes defined by the eigenvectors of H.
- Specifically, the component of w^* that is aligned with the i-th eigenvector of H is rescaled by a factor of $\frac{\lambda_i}{\lambda_i+\alpha}$

L^2 Parameter Regularization

- Because H is real and symmetric, we can decompose it into a diagonal matrix Λ and an orthonormal basis of eigenvectors, Q, such that $\mathsf{H} = Q\Lambda Q^T$.
- Applying the decomposition $\tilde{w} = (\mathsf{H} + \alpha \mathsf{I})^{-1}\mathsf{H}w^*$

$$\begin{aligned} \tilde{w} &= (Q\Lambda Q^T + \alpha I)^{-1} Q\Lambda Q^T w^* && (1) \\ &= \left[Q(\Lambda + \alpha I)Q^T\right]^{-1} Q\Lambda Q^T w^* && (2) \\ &= Q(\Lambda + \alpha I)^{-1} \Lambda Q^T w^* && (3) \\ &= Q\frac{\Lambda}{\Lambda + \alpha I} Q^T w^* && (4) \end{aligned}$$

- We see that the effect of weight decay is to rescale w^* along the axes defined by the eigenvectors of H.
- Specifically, the component of w^* that is aligned with the i-th eigenvector of H is rescaled by a factor of $\frac{\lambda_i}{\lambda_i + \alpha}$

L^2 Parameter Regularization

- Because H is real and symmetric, we can decompose it into a diagonal matrix $\mathbf{\Lambda}$ and an orthonormal basis of eigenvectors, Q, such that $\mathsf{H} = Q\mathbf{\Lambda}Q^T$.
- Applying the decomposition $\tilde{w} = (\mathsf{H} + \alpha\mathsf{I})^{-1}\mathsf{H}w^*$

$$
\begin{aligned}
\tilde{w} &= (Q\mathbf{\Lambda}Q^T + \alpha I)^{-1}Q\mathbf{\Lambda}Q^T w^* && (1)\\
&= \left[Q(\mathbf{\Lambda} + \alpha I)Q^T\right]^{-1} Q\mathbf{\Lambda}Q^T w^* && (2)\\
&= Q(\mathbf{\Lambda} + \alpha I)^{-1}\mathbf{\Lambda}Q^T w^* && (3)\\
&= Q\frac{\mathbf{\Lambda}}{\mathbf{\Lambda} + \alpha I}Q^T w^* && (4)
\end{aligned}
$$

- We see that the effect of weight decay is to rescale w^* along the axes defined by the eigenvectors of H.
- Specifically, the component of w^* that is aligned with the i-th eigenvector of H is rescaled by a factor of $\frac{\lambda_i}{\lambda_i+\alpha}$

References