Numerical Optimization

Shan-Hung Wu
shwu@cs.nthu.edu.tw

Department of Computer Science,
National Tsing Hua University, Taiwan

Machine Learning

Outline

1. **Numerical Computation**
2. **Optimization Problems**
3. **Unconstrained Optimization**
   - Gradient Descent
   - Newton's Method
4. **Optimization in ML: Stochastic Gradient Descent**
   - Perceptron
   - Adaline
   - Stochastic Gradient Descent
5. **Constrained Optimization**
6. **Optimization in ML: Regularization**
   - Linear Regression
   - Polynomial Regression
   - Generalizability & Regularization
7. **Duality***

Outline

Numerical Computation

- Machine learning algorithms usually require a high amount of numerical computation in involving real numbers

Numerical Computation

- Machine learning algorithms usually require a high amount of numerical computation in involving real numbers
- However, real numbers cannot be represented precisely using a finite amount of memory

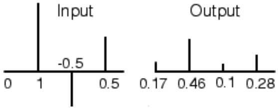
Numerical Computation

- Machine learning algorithms usually require a high amount of numerical computation in involving real numbers
- However, real numbers cannot be represented precisely using a finite amount of memory
- Watch out the ***numeric errors*** when implementing machine learning algorithms

Overflow and Underflow I

- Consider the ***softmax function***
  $\mathbf{softmax} : \mathbb{R}^d \to \mathbb{R}^d$:

$$\text{softmax}(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^{d} \exp(x_j)}$$

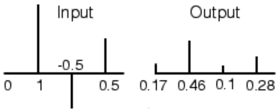


- Commonly used to transform a group of real values to "probabilities"

Overflow and Underflow I

- Consider the ***softmax function***
  $\textbf{softmax} : \mathbb{R}^d \to \mathbb{R}^d$:

$$\mathrm{softmax}(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^{d} \exp(x_j)}$$

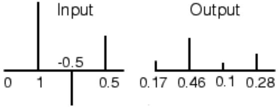


- Commonly used to transform a group of real values to "probabilities"
- Analytically, if $x_i = c$ for all i, then $\mathrm{softmax}(\boldsymbol{x})_i = 1/d$

Overflow and Underflow I

- Consider the ***softmax function***
softmax : $\mathbb{R}^d \to \mathbb{R}^d$:

$$\text{softmax}(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^{d} \exp(x_j)}$$



- Commonly used to transform a group of real values to "probabilities"
- Analytically, if $x_i = c$ for all i, then $\text{softmax}(\boldsymbol{x})_i = 1/d$
- Numerically, this may not occur when $|c|$ is large
    - A positive c causes overflow
    - A negative c causes underflow and divide-by-zero error
- How to avoid these errors?

Overflow and Underflow II

- Instead of evaluating $\mathrm{softmax}(\boldsymbol{x})$ directly, we can transform $\boldsymbol{x}$ into

$$\boldsymbol{z} = \boldsymbol{x} - \max_i x_i \mathbf{1}$$

  and then evaluate $\mathrm{softmax}(\boldsymbol{z})$

Overflow and Underflow II

- Instead of evaluating $\mathrm{softmax}(\boldsymbol{x})$ directly, we can transform $\boldsymbol{x}$ into

$$z = \boldsymbol{x} - \max_i x_i \mathbf{1}$$

  and then evaluate $\mathrm{softmax}(z)$
- $\mathrm{softmax}(z)_i = \frac{\exp(x_i - m)}{\sum \exp(x_j - m)} = \frac{\exp(x_i)/\exp(m)}{\sum \exp(x_j)/\exp(m)} = \frac{\exp(x_i)}{\sum \exp(x_j)} = \mathrm{softmax}(\boldsymbol{x})_i$

Overflow and Underflow II

- Instead of evaluating $\text{softmax}(\boldsymbol{x})$ directly, we can transform $\boldsymbol{x}$ into

$$z = \boldsymbol{x} - \max_i x_i \mathbf{1}$$

  and then evaluate $\text{softmax}(z)$
- $\text{softmax}(z)_i = \frac{\exp(x_i - m)}{\sum \exp(x_j - m)} = \frac{\exp(x_i)/\exp(m)}{\sum \exp(x_j)/\exp(m)} = \frac{\exp(x_i)}{\sum \exp(x_j)} = \text{softmax}(\boldsymbol{x})_i$
- No overflow, as $\exp(\text{largest attribute of } \boldsymbol{x}) = 1$
- Denominator is at least 1, no divide-by-zero error

Overflow and Underflow II

- Instead of evaluating $\mathrm{softmax}(\boldsymbol{x})$ directly, we can transform $\boldsymbol{x}$ into

$$z = \boldsymbol{x} - \max_i x_i \mathbf{1}$$

  and then evaluate $\mathrm{softmax}(z)$
- $\mathrm{softmax}(z)_i = \frac{\exp(x_i - m)}{\sum \exp(x_j - m)} = \frac{\exp(x_i)/\exp(m)}{\sum \exp(x_j)/\exp(m)} = \frac{\exp(x_i)}{\sum \exp(x_j)} = \mathrm{softmax}(\boldsymbol{x})_i$
- No overflow, as $\exp(\text{largest attribute of } \boldsymbol{x}) = 1$
- Denominator is at least 1, no divide-by-zero error
- What are the numerical issues of $\log \mathrm{softmax}(z)$? How to stabilize it? [Homework]

Poor Conditioning I

- The "conditioning" refer to how much the input of a function can change given a small change in the output

Poor Conditioning I

- The "conditioning" refer to how much the input of a function can change given a small change in the output
- Suppose we want to solve $\boldsymbol{x}$ in $f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$, where $\boldsymbol{A}^{-1}$ exists

Poor Conditioning I

- The "conditioning" refer to how much the input of a function can change given a small change in the output
- Suppose we want to solve $\boldsymbol{x}$ in $f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$, where $\boldsymbol{A}^{-1}$ exists
- The ***condition umber*** of $\boldsymbol{A}$ can be expressed by

$$\kappa(\boldsymbol{A}) = \max_{i,j} \left| \frac{\lambda_i}{\lambda_j} \right|$$

Poor Conditioning I

- The "conditioning" refer to how much the input of a function can change given a small change in the output
- Suppose we want to solve $\boldsymbol{x}$ in $f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$, where $\boldsymbol{A}^{-1}$ exists
- The ***condition umber*** of $\boldsymbol{A}$ can be expressed by

$$\kappa(\boldsymbol{A}) = \max_{i,j} \left| \frac{\lambda_i}{\lambda_j} \right|$$

- We say the problem is ***poorly*** (or ***ill-***) ***conditioned*** when $\kappa(\boldsymbol{A})$ is large

Poor Conditioning I

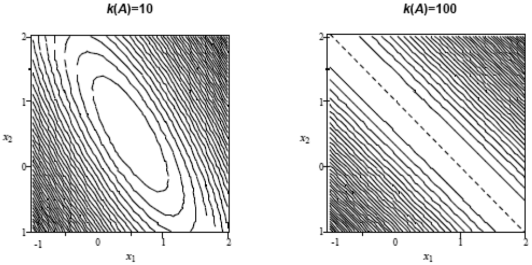
- The "conditioning" refer to how much the input of a function can change given a small change in the output
- Suppose we want to solve $\boldsymbol{x}$ in $f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$, where $\boldsymbol{A}^{-1}$ exists
- The ***condition umber*** of $\boldsymbol{A}$ can be expressed by

$$\kappa(\boldsymbol{A}) = \max_{i,j} \left| \frac{\lambda_i}{\lambda_j} \right|$$

- We say the problem is ***poorly*** (or ***ill-***) ***conditioned*** when $\kappa(\boldsymbol{A})$ is large
- Hard to solve $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{y}$ precisely given a rounded $\boldsymbol{y}$
  - $\boldsymbol{A}^{-1}$ amplifies pre-existing numeric errors

Poor Conditioning II

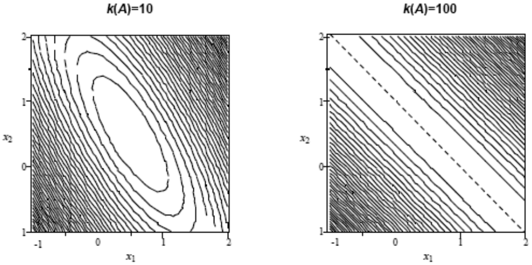
- The contours of $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^\top \boldsymbol{x} + c$, where $\boldsymbol{A}$ is symmetric:



- When $\kappa(\boldsymbol{A})$ is large, f stretches space differently along different attribute directions
  - Surface is flat in some directions but steep in others

Poor Conditioning II

- The contours of $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^\top \boldsymbol{x} + c$, where $\boldsymbol{A}$ is symmetric:



- When $\kappa(\boldsymbol{A})$ is large, f stretches space differently along different attribute directions
  - Surface is flat in some directions but steep in others
- Hard to solve $f'(\boldsymbol{x}) = \boldsymbol{0} \Rightarrow \boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$

Outline

Optimization Problems

- An ***optimization problem*** is to minimize a ***cost function*** $f : \mathbb{R}^d \to \mathbb{R}$:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$
$$\text{subject to } \boldsymbol{x} \in \mathbb{C}$$

where $\mathbb{C} \subseteq \mathbb{R}^d$ is called the ***feasible set*** containing ***feasible points***

Optimization Problems

- An ***optimization problem*** is to minimize a ***cost function*** $f : \mathbb{R}^d \to \mathbb{R}$:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$
$$\text{subject to } \boldsymbol{x} \in \mathbb{C}$$

where $\mathbb{C} \subseteq \mathbb{R}^d$ is called the ***feasible set*** containing ***feasible points***
  - Or, maximizing an ***objective function***
  - Maximizing f equals to minimizing $-f$

Optimization Problems

- An ***optimization problem*** is to minimize a ***cost function*** $f : \mathbb{R}^d \to \mathbb{R}$:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$
$$\text{subject to } \boldsymbol{x} \in \mathbb{C}$$

  where $\mathbb{C} \subseteq \mathbb{R}^d$ is called the ***feasible set*** containing ***feasible points***
  - Or, maximizing an ***objective function***
  - Maximizing f equals to minimizing $-f$
- If $\mathbb{C} = \mathbb{R}^d$, we say the optimization problem is unconstrained

Optimization Problems

- An ***optimization problem*** is to minimize a ***cost function*** $f : \mathbb{R}^d \to \mathbb{R}$:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$
$$\text{subject to } \boldsymbol{x} \in \mathbb{C}$$

  where $\mathbb{C} \subseteq \mathbb{R}^d$ is called the ***feasible set*** containing ***feasible points***
  - Or, maximizing an ***objective function***
  - Maximizing f equals to minimizing $-f$
- If $\mathbb{C} = \mathbb{R}^d$, we say the optimization problem is unconstrained
- $\mathbb{C}$ can be a set of function ***constrains***, i.e., $\mathbb{C} = \{\boldsymbol{x} : g^{(i)}(\boldsymbol{x}) \leq 0\}_i$

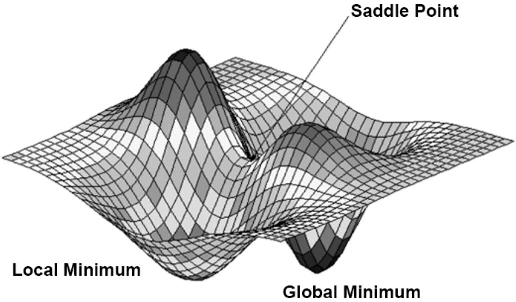
Optimization Problems

- An ***optimization problem*** is to minimize a ***cost function*** $f : \mathbb{R}^d \to \mathbb{R}$:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$
$$\text{subject to } \boldsymbol{x} \in \mathbb{C}$$

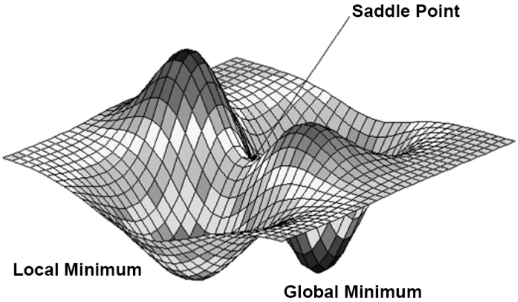
  where $\mathbb{C} \subseteq \mathbb{R}^d$ is called the ***feasible set*** containing ***feasible points***
  - Or, maximizing an ***objective function***
  - Maximizing f equals to minimizing $-f$
- If $\mathbb{C} = \mathbb{R}^d$, we say the optimization problem is unconstrained
- $\mathbb{C}$ can be a set of function ***constrains***, i.e., $\mathbb{C} = \{\boldsymbol{x} : g^{(i)}(\boldsymbol{x}) \leq 0\}_i$
- Sometimes, we single out equality constrains $\mathbb{C} = \{\boldsymbol{x} : g^{(i)}(\boldsymbol{x}) \leq 0, h^{(j)}(\boldsymbol{x}) = 0\}_{i,j}$
  - Each equality constrain can be written as two inequality constrains

Minimums and Optimal Points



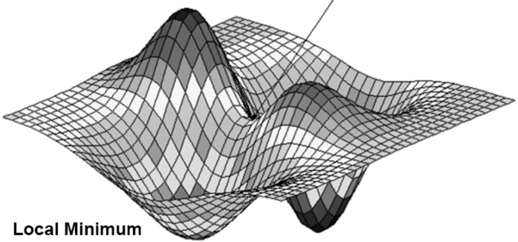
- ***Critical points***: $\{\boldsymbol{x} : f'(\boldsymbol{x}) = \boldsymbol{0}\}$

Minimums and Optimal Points



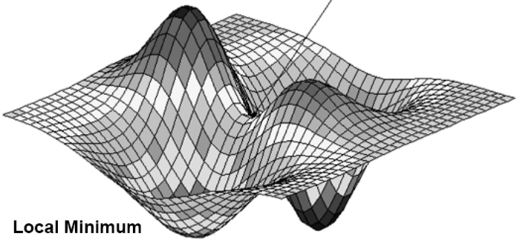
- ***Critical points***: $\{\boldsymbol{x} : f'(\boldsymbol{x}) = \boldsymbol{0}\}$
  - Minima: $\{\boldsymbol{x} : f'(\boldsymbol{x}) = \boldsymbol{0}$ and $\boldsymbol{H}(f)(\boldsymbol{x}) \succ \boldsymbol{O}\}$, where $\boldsymbol{H}(f)(\boldsymbol{x})$ is the Hessian matrix (containing curvatures) of f at point $\boldsymbol{x}$

Minimums and Optimal Points



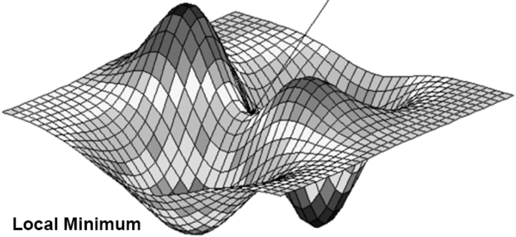
- ***Critical points***: $\{\boldsymbol{x} : f'(\boldsymbol{x}) = \boldsymbol{0}\}$
  - Minima: $\{\boldsymbol{x} : f'(\boldsymbol{x}) = \boldsymbol{0} \text{ and } \boldsymbol{H}(f)(\boldsymbol{x}) \succ \boldsymbol{O}\}$, where $\boldsymbol{H}(f)(\boldsymbol{x})$ is the Hessian matrix (containing curvatures) of f at point $\boldsymbol{x}$
  - Maxima: $\{\boldsymbol{x} : f'(\boldsymbol{x}) = \boldsymbol{0} \text{ and } \boldsymbol{H}(f)(\boldsymbol{x}) \prec \boldsymbol{O}\}$

Minimums and Optimal Points



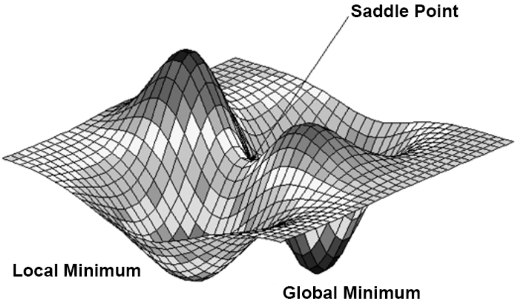
- ***Critical points***: $\{\boldsymbol{x} : f'(\boldsymbol{x}) = \mathbf{0}\}$
  - Minima: $\{\boldsymbol{x} : f'(\boldsymbol{x}) = \mathbf{0} \text{ and } \boldsymbol{H}(f)(\boldsymbol{x}) \succ \boldsymbol{O}\}$, where $\boldsymbol{H}(f)(\boldsymbol{x})$ is the Hessian matrix (containing curvatures) of f at point $\boldsymbol{x}$
  - Maxima: $\{\boldsymbol{x} : f'(\boldsymbol{x}) = \mathbf{0} \text{ and } \boldsymbol{H}(f)(\boldsymbol{x}) \prec \boldsymbol{O}\}$
  - Plateau or saddle points: $\{\boldsymbol{x} : f'(\boldsymbol{x}) = \mathbf{0} \text{ and } \boldsymbol{H}(f)(\boldsymbol{x}) = \boldsymbol{O} \text{ or indefinite}\}$

Minimums and Optimal Points



- ***Critical points***: $\{\boldsymbol{x} : f'(\boldsymbol{x}) = \boldsymbol{0}\}$
  - Minima: $\{\boldsymbol{x} : f'(\boldsymbol{x}) = \boldsymbol{0} \text{ and } \boldsymbol{H}(f)(\boldsymbol{x}) \succ \boldsymbol{O}\}$, where $\boldsymbol{H}(f)(\boldsymbol{x})$ is the Hessian matrix (containing curvatures) of f at point $\boldsymbol{x}$
  - Maxima: $\{\boldsymbol{x} : f'(\boldsymbol{x}) = \boldsymbol{0} \text{ and } \boldsymbol{H}(f)(\boldsymbol{x}) \prec \boldsymbol{O}\}$
  - Plateau or saddle points: $\{\boldsymbol{x} : f'(\boldsymbol{x}) = \boldsymbol{0} \text{ and } \boldsymbol{H}(f)(\boldsymbol{x}) = \boldsymbol{O} \text{ or indefinite}\}$
- $y^* = \min_{\boldsymbol{x} \in \mathbb{C}} f(\boldsymbol{x}) \in \mathbb{R}$ is called the ***global minimum***
  - Global minima vs. local minima

Minimums and Optimal Points

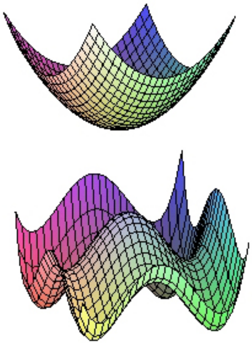


- ***Critical points***: $\{\boldsymbol{x} : f'(\boldsymbol{x}) = \boldsymbol{0}\}$
  - Minima: $\{\boldsymbol{x} : f'(\boldsymbol{x}) = \boldsymbol{0} \text{ and } \boldsymbol{H}(f)(\boldsymbol{x}) \succ \boldsymbol{O}\}$, where $\boldsymbol{H}(f)(\boldsymbol{x})$ is the Hessian matrix (containing curvatures) of f at point $\boldsymbol{x}$
  - Maxima: $\{\boldsymbol{x} : f'(\boldsymbol{x}) = \boldsymbol{0} \text{ and } \boldsymbol{H}(f)(\boldsymbol{x}) \prec \boldsymbol{O}\}$
  - Plateau or saddle points: $\{\boldsymbol{x} : f'(\boldsymbol{x}) = \boldsymbol{0} \text{ and } \boldsymbol{H}(f)(\boldsymbol{x}) = \boldsymbol{O} \text{ or indefinite}\}$
- $y^* = \min_{\boldsymbol{x} \in \mathbb{C}} f(\boldsymbol{x}) \in \mathbb{R}$ is called the ***global minimum***
  - Global minima vs. local minima
- $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x} \in \mathbb{C}} f(\boldsymbol{x})$ is called the ***optimal point***

Convex Optimization Problems

- An optimization problem is ***convex*** iff
  1. f is convex by having a "convex hull" surface, i.e.,

$$\boldsymbol{H}(f)(\boldsymbol{x}) \succeq \boldsymbol{0}, \forall \boldsymbol{x}$$

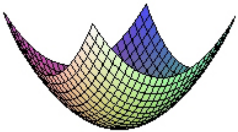


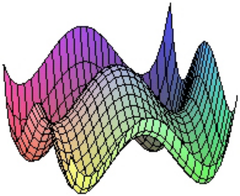
Convex Optimization Problems

- An optimization problem is ***convex*** iff
  1. f is convex by having a "convex hull" surface, i.e.,

$$\boldsymbol{H}(f)(\boldsymbol{x}) \succeq \boldsymbol{0}, \forall \boldsymbol{x}$$

  2. $g_i(\boldsymbol{x})$'s are convex and $h_j(\boldsymbol{x})$'s are affine



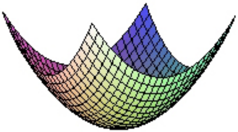


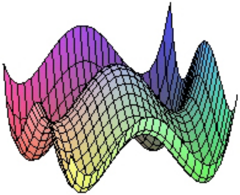
Convex Optimization Problems

- An optimization problem is ***convex*** iff
  1. f is convex by having a "convex hull" surface, i.e.,

$$\boldsymbol{H}(f)(\boldsymbol{x}) \succeq \boldsymbol{0}, \forall \boldsymbol{x}$$

  2. $g_i(\boldsymbol{x})$'s are convex and $h_j(\boldsymbol{x})$'s are affine
- Convex problems are "easier" since
  - Local minima are necessarily global minima
  - No saddle point

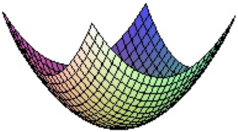




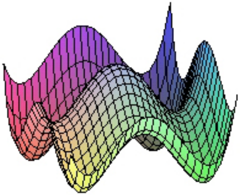
Convex Optimization Problems

- An optimization problem is ***convex*** iff
  1. f is convex by having a "convex hull" surface, i.e.,

$$\boldsymbol{H}(f)(\boldsymbol{x}) \succeq \boldsymbol{0}, \forall \boldsymbol{x}$$



  2. $g_i(\boldsymbol{x})$'s are convex and $h_j(\boldsymbol{x})$'s are affine
- Convex problems are "easier" since
  - Local minima are necessarily global minima
  - No saddle point
  - We can get the global minimum by solving $f'(\boldsymbol{x}) = \boldsymbol{0}$



Analytical Solutions vs. Numerical Solutions I

- Consider the problem:

$$\arg\min_{\boldsymbol{x}} \frac{1}{2}\left(\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|^2+\lambda\|\boldsymbol{x}\|^2\right)$$

- Analytical solutions?

Analytical Solutions vs. Numerical Solutions I

- Consider the problem:

$$\arg\min_{\boldsymbol{x}} \frac{1}{2}\left(\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|^2+\lambda\|\boldsymbol{x}\|^2\right)$$

- Analytical solutions?
- The cost function $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top\left(\boldsymbol{A}^\top\boldsymbol{A}+\lambda\boldsymbol{I}\right)\boldsymbol{x}-\boldsymbol{b}^\top\boldsymbol{A}\boldsymbol{x}+\frac{1}{2}\|\boldsymbol{b}\|^2$ is convex

Analytical Solutions vs. Numerical Solutions I

- Consider the problem:

$$\arg\min_{\boldsymbol{x}} \frac{1}{2}\left(\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|^2+\lambda\|\boldsymbol{x}\|^2\right)$$

- Analytical solutions?
- The cost function $f(\boldsymbol{x})=\frac{1}{2}\boldsymbol{x}^\top\left(\boldsymbol{A}^\top\boldsymbol{A}+\lambda\boldsymbol{I}\right)\boldsymbol{x}-\boldsymbol{b}^\top\boldsymbol{A}\boldsymbol{x}+\frac{1}{2}\|\boldsymbol{b}\|^2$ is convex
- Solving $f'(\boldsymbol{x})=\boldsymbol{x}^\top\left(\boldsymbol{A}^\top\boldsymbol{A}+\lambda\boldsymbol{I}\right)-\boldsymbol{b}^\top\boldsymbol{A}=0$, we have

$$\boldsymbol{x}^*=\left(\boldsymbol{A}^\top\boldsymbol{A}+\lambda\boldsymbol{I}\right)^{-1}\boldsymbol{A}^\top\boldsymbol{b}$$

Analytical Solutions vs. Numerical Solutions II

- Problem ($\boldsymbol{A} \in \mathbb{R}^{n \times d}$, $\boldsymbol{b} \in \mathbb{R}^n$, $\lambda \in \mathbb{R}$):

$$\arg\min_{\boldsymbol{x} \in \mathbb{R}^d} \frac{1}{2} \left(\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|^2 + \lambda \|\boldsymbol{x}\|^2 \right)$$

- Analytical solution: $\boldsymbol{x}^* = \left(\boldsymbol{A}^\top \boldsymbol{A} + \lambda \boldsymbol{I} \right)^{-1} \boldsymbol{A}^\top \boldsymbol{b}$

Analytical Solutions vs. Numerical Solutions II

- Problem ($\boldsymbol{A} \in \mathbb{R}^{n \times d}$, $\boldsymbol{b} \in \mathbb{R}^n$, $\lambda \in \mathbb{R}$):

$$\arg\min_{\boldsymbol{x} \in \mathbb{R}^d} \frac{1}{2}\left(\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|^2 + \lambda \|\boldsymbol{x}\|^2\right)$$

- Analytical solution: $\boldsymbol{x}^* = \left(\boldsymbol{A}^\top \boldsymbol{A} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{A}^\top \boldsymbol{b}$
- In practice, we may not be able to solve $f'(\boldsymbol{x}) = \boldsymbol{0}$ analytically and get $\boldsymbol{x}$ in a closed form
  - E.g., when $\lambda = 0$ and $n < d$

Analytical Solutions vs. Numerical Solutions II

- Problem ($\boldsymbol{A} \in \mathbb{R}^{n \times d}$, $\boldsymbol{b} \in \mathbb{R}^n$, $\lambda \in \mathbb{R}$):

$$\arg\min_{\boldsymbol{x} \in \mathbb{R}^d} \frac{1}{2}\left(\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|^2 + \lambda \|\boldsymbol{x}\|^2\right)$$

- Analytical solution: $\boldsymbol{x}^* = \left(\boldsymbol{A}^\top \boldsymbol{A} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{A}^\top \boldsymbol{b}$
- In practice, we may not be able to solve $f'(\boldsymbol{x}) = \boldsymbol{0}$ analytically and get $\boldsymbol{x}$ in a closed form
  - E.g., when $\lambda = 0$ and $n < d$
- Even if we can, the computation cost may be too hight
  - E.g, inverting $\boldsymbol{A}^\top \boldsymbol{A} + \lambda \boldsymbol{I} \in \mathbb{R}^{d \times d}$ takes $O(d^3)$ time

Analytical Solutions vs. Numerical Solutions II

- Problem ($\boldsymbol{A} \in \mathbb{R}^{n\times d}$, $\boldsymbol{b} \in \mathbb{R}^n$, $\lambda \in \mathbb{R}$):

$$\arg\min_{\boldsymbol{x}\in\mathbb{R}^d} \frac{1}{2}\left(\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|^2 + \lambda\|\boldsymbol{x}\|^2\right)$$

- Analytical solution: $\boldsymbol{x}^* = \left(\boldsymbol{A}^\top\boldsymbol{A}+\lambda\boldsymbol{I}\right)^{-1}\boldsymbol{A}^\top\boldsymbol{b}$
- In practice, we may not be able to solve $f'(\boldsymbol{x}) = \boldsymbol{0}$ analytically and get $\boldsymbol{x}$ in a closed form
    - E.g., when $\lambda = 0$ and $n < d$
- Even if we can, the computation cost may be too hight
    - E.g, inverting $\boldsymbol{A}^\top\boldsymbol{A}+\lambda\boldsymbol{I} \in \mathbb{R}^{d\times d}$ takes $O(d^3)$ time
- ***Numerical methods***: since numerical errors are inevitable, why not just obtain an ***approximation*** of $\boldsymbol{x}^*$?

Analytical Solutions vs. Numerical Solutions II

- Problem ($\boldsymbol{A} \in \mathbb{R}^{n \times d}$, $\boldsymbol{b} \in \mathbb{R}^n$, $\lambda \in \mathbb{R}$):

$$\arg\min_{\boldsymbol{x} \in \mathbb{R}^d} \frac{1}{2} \left(\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|^2 + \lambda \|\boldsymbol{x}\|^2 \right)$$

- Analytical solution: $\boldsymbol{x}^* = \left(\boldsymbol{A}^\top \boldsymbol{A} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{A}^\top \boldsymbol{b}$
- In practice, we may not be able to solve $f'(\boldsymbol{x}) = \boldsymbol{0}$ analytically and get $\boldsymbol{x}$ in a closed form
  - E.g., when $\lambda = 0$ and $n < d$
- Even if we can, the computation cost may be too hight
  - E.g, inverting $\boldsymbol{A}^\top \boldsymbol{A} + \lambda \boldsymbol{I} \in \mathbb{R}^{d \times d}$ takes $O(d^3)$ time
- ***Numerical methods***: since numerical errors are inevitable, why not just obtain an ***approximation*** of $\boldsymbol{x}^*$?
- Start from $\boldsymbol{x}^{(0)}$, iteratively calculating $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots$ such that $f(\boldsymbol{x}^{(1)}) \geq f(\boldsymbol{x}^{(2)}) \geq \cdots$
  - Usually require much less time to have a good enough $\boldsymbol{x}^{(t)} \approx \boldsymbol{x}^*$

Outline

Unconstrained Optimization

- Problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}),$$

where $f:\mathbb{R}^d\to\mathbb{R}$ is not necessarily convex

General Descent Algorithm

Input: $\boldsymbol{x}^{(0)} \in \mathbb{R}^d$, an initial guess
repeat

Determine a ***descent direction*** $\boldsymbol{d}^{(t)} \in \mathbb{R}^d$;
Line search: choose a ***step size*** or ***learning rate*** $\eta^{(t)} > 0$ such that $f(\boldsymbol{x}^{(t)} + \eta^{(t)}\boldsymbol{d}^{(t)})$ is minimal along the ray $\boldsymbol{x}^{(t)} + \eta^{(t)}\boldsymbol{d}^{(t)}$;
Update rule: $\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} + \eta^{(t)}\boldsymbol{d}^{(t)}$;

until *convergence criterion is satisfied*;

General Descent Algorithm

Input: $\boldsymbol{x}^{(0)} \in \mathbb{R}^d$, an initial guess
repeat
  Determine a ***descent direction*** $\boldsymbol{d}^{(t)} \in \mathbb{R}^d$;
  Line search: choose a ***step size*** or ***learning rate*** $\eta^{(t)} > 0$ such that $f(\boldsymbol{x}^{(t)} + \eta^{(t)}\boldsymbol{d}^{(t)})$ is minimal along the ray $\boldsymbol{x}^{(t)} + \eta^{(t)}\boldsymbol{d}^{(t)}$;
  Update rule: $\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} + \eta^{(t)}\boldsymbol{d}^{(t)}$;
until *convergence criterion is satisfied*;

- Convergence criterion: $\|\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)}\| \leq \varepsilon$, $\|\nabla f(\boldsymbol{x}^{(t+1)})\| \leq \varepsilon$, etc.
- Line search step could be skipped by letting $\eta^{(t)}$ be a small constant

Outline

Gradient Descent I

- By Taylor's theorem, we can approximate f locally at point $\boldsymbol{x}^{(t)}$ using a linear function $\tilde{f}$, i.e.,

$$f(\boldsymbol{x}) \approx \tilde{f}(\boldsymbol{x};\boldsymbol{x}^{(t)}) = f(\boldsymbol{x}^{(t)}) + \nabla f(\boldsymbol{x}^{(t)})^\top (\boldsymbol{x} - \boldsymbol{x}^{(t)})$$

  for $\boldsymbol{x}$ close enough to $\boldsymbol{x}^{(t)}$

Gradient Descent I

- By Taylor's theorem, we can approximate f locally at point $\boldsymbol{x}^{(t)}$ using a linear function $\tilde{f}$, i.e.,

$$f(\boldsymbol{x}) \approx \tilde{f}(\boldsymbol{x};\boldsymbol{x}^{(t)}) = f(\boldsymbol{x}^{(t)}) + \nabla f(\boldsymbol{x}^{(t)})^\top (\boldsymbol{x} - \boldsymbol{x}^{(t)})$$

  for $\boldsymbol{x}$ close enough to $\boldsymbol{x}^{(t)}$
- This implies that if we pick a close $\boldsymbol{x}^{(t+1)}$ that decreases $\tilde{f}$, we are likely to decrease f as well

Gradient Descent I

- By Taylor's theorem, we can approximate f locally at point $\boldsymbol{x}^{(t)}$ using a linear function $\tilde{f}$, i.e.,

$$f(\boldsymbol{x}) \approx \tilde{f}(\boldsymbol{x};\boldsymbol{x}^{(t)}) = f(\boldsymbol{x}^{(t)}) + \nabla f(\boldsymbol{x}^{(t)})^\top (\boldsymbol{x} - \boldsymbol{x}^{(t)})$$

  for $\boldsymbol{x}$ close enough to $\boldsymbol{x}^{(t)}$
- This implies that if we pick a close $\boldsymbol{x}^{(t+1)}$ that decreases $\tilde{f}$, we are likely to decrease f as well
- We can pick $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \eta \nabla f(\boldsymbol{x}^{(t)})$ for some small $\eta > 0$, since

$$\tilde{f}(\boldsymbol{x}^{(t+1)}) = f(\boldsymbol{x}^{(t)}) - \eta \|\nabla f(\boldsymbol{x}^{(t)})\|^2 \leq \tilde{f}(\boldsymbol{x}^{(t)})$$

Gradient Descent II

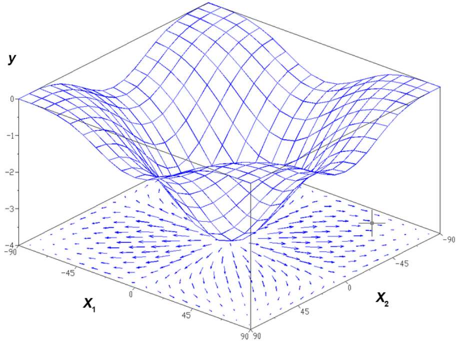
Input: $\boldsymbol{x}^{(0)} \in \mathbb{R}^d$ an initial guess, a small $\eta > 0$
repeat
  $\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} - \eta \nabla f(\boldsymbol{x}^{(t)})$;
until *convergence criterion is satisfied*;

Is Negative Gradient a Good Direction? I

- Update rule:
  $\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} - \eta \nabla f(\boldsymbol{x}^{(t)})$

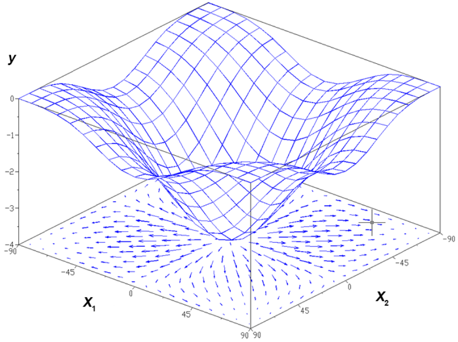
Is Negative Gradient a Good Direction? I

- Update rule:
  $\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} - \eta \nabla f(\boldsymbol{x}^{(t)})$
- Yes, as $\nabla f(\boldsymbol{x}^{(t)}) \in \mathbb{R}^d$ denotes the steepest ascent direction of f at point $\boldsymbol{x}^{(t)}$
- $-\nabla f(\boldsymbol{x}^{(t)}) \in \mathbb{R}^d$ the steepest descent direction



Is Negative Gradient a Good Direction? I

- Update rule:
  $\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} - \eta \nabla f(\boldsymbol{x}^{(t)})$
- Yes, as $\nabla f(\boldsymbol{x}^{(t)}) \in \mathbb{R}^d$ denotes the steepest ascent direction of f at point $\boldsymbol{x}^{(t)}$
- $-\nabla f(\boldsymbol{x}^{(t)}) \in \mathbb{R}^d$ the steepest descent direction
- But why?



Is Negative Gradient a Good Direction? II

- Consider the slope of f in a given direction $\boldsymbol{u}$ at point $\boldsymbol{x}^{(t)}$
- This is the ***directional derivative*** of f, i.e., the derivative of function $f(\boldsymbol{x}^{(t)}+\varepsilon\boldsymbol{u})$ with respect to ε, evaluated at $\varepsilon = 0$

Is Negative Gradient a Good Direction? II

- Consider the slope of f in a given direction $\boldsymbol{u}$ at point $\boldsymbol{x}^{(t)}$
- This is the ***directional derivative*** of f, i.e., the derivative of function $f(\boldsymbol{x}^{(t)}+\varepsilon\boldsymbol{u})$ with respect to ε, evaluated at $\varepsilon=0$
- By the chain rule, we have $\frac{\partial}{\partial\varepsilon}f(\boldsymbol{x}^{(t)}+\varepsilon\boldsymbol{u})=\nabla f(\boldsymbol{x}^{(t)}+\varepsilon\boldsymbol{u})^\top\boldsymbol{u}$, which equals to $\nabla f(\boldsymbol{x}^{(t)})^\top\boldsymbol{u}$ when $\varepsilon=0$

Theorem (Chain Rule)

Let $\boldsymbol{g}:\mathbb{R}\to\mathbb{R}^d$ *and* $f:\mathbb{R}^d\to\mathbb{R}$, *then*

$$(f\circ\boldsymbol{g})'(x)=f'(\boldsymbol{g}(x))\boldsymbol{g}'(x)=\nabla f(\boldsymbol{g}(x))^\top\begin{bmatrix}g_1'(x)\\ \vdots\\ g_n'(x)\end{bmatrix}.$$

Is Negative Gradient a Good Direction? III

- To find the direction that decreases f fastest at $\boldsymbol{x}^{(t)}$, we solve the problem:

$$\arg\min_{\boldsymbol{u},\|\boldsymbol{u}\|=1} \nabla f(\boldsymbol{x}^{(t)})^\top \boldsymbol{u} = \arg\min_{\boldsymbol{u},\|\boldsymbol{u}\|=1} \|\nabla f(\boldsymbol{x}^{(t)})\|\|\boldsymbol{u}\|\cos\theta$$

  where θ is the the angle between $\boldsymbol{u}$ and $\nabla f(\boldsymbol{x}^{(t)})$

Is Negative Gradient a Good Direction? III

- To find the direction that decreases f fastest at $\boldsymbol{x}^{(t)}$, we solve the problem:

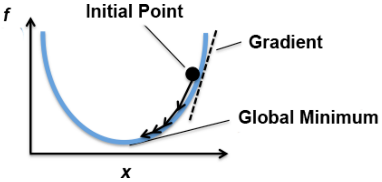
$$\arg\min_{\boldsymbol{u},\|\boldsymbol{u}\|=1} \nabla f(\boldsymbol{x}^{(t)})^\top \boldsymbol{u} = \arg\min_{\boldsymbol{u},\|\boldsymbol{u}\|=1} \|\nabla f(\boldsymbol{x}^{(t)})\| \|\boldsymbol{u}\| \cos\theta$$

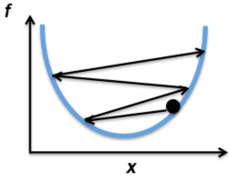
  where θ is the the angle between $\boldsymbol{u}$ and $\nabla f(\boldsymbol{x}^{(t)})$
- This amounts to solve

$$\arg\min_{\boldsymbol{u}} \cos\theta$$

- So, $\boldsymbol{u}^* = -\nabla f(\boldsymbol{x}^{(t)})$ is the steepest descent direction of f at point $\boldsymbol{x}^{(t)}$

How to Set Learning Rate η? I





- Too small an η results in slow descent speed and many iterations
- Too large an η may overshoot the optimal point along the gradient and goes uphill

How to Set Learning Rate η? I

- Too small an η results in slow descent speed and many iterations
- Too large an η may overshoot the optimal point along the gradient and goes uphill
- One way to set a better η is to leverage the ***curvatures*** of f
    - The more curvy f at point $\boldsymbol{x}^{(t)}$, the smaller the η

How to Set Learning Rate η? II

- By Taylor's theorem, we can approximate f locally at point $\boldsymbol{x}^{(t)}$ using a quadratic function $\tilde{f}$:

$$f(\boldsymbol{x}) \approx \tilde{f}(\boldsymbol{x};\boldsymbol{x}^{(t)}) = f(\boldsymbol{x}^{(t)}) + \nabla f(\boldsymbol{x}^{(t)})^\top(\boldsymbol{x}-\boldsymbol{x}^{(t)}) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}^{(t)})^\top \boldsymbol{H}(f)(\boldsymbol{x}^{(t)})(\boldsymbol{x}-\boldsymbol{x}^{(t)})$$

  for $\boldsymbol{x}$ close enough to $\boldsymbol{x}^{(t)}$
  - $\boldsymbol{H}(f)(\boldsymbol{x}^{(t)}) \in \mathbb{R}^{d\times d}$ is the (symmetric) Hessian matrix of f at $\boldsymbol{x}^{(t)}$

How to Set Learning Rate η? II

- By Taylor's theorem, we can approximate f locally at point $\boldsymbol{x}^{(t)}$ using a quadratic function $\tilde{f}$:

$$f(\boldsymbol{x}) \approx \tilde{f}(\boldsymbol{x};\boldsymbol{x}^{(t)}) = f(\boldsymbol{x}^{(t)}) + \nabla f(\boldsymbol{x}^{(t)})^\top (\boldsymbol{x} - \boldsymbol{x}^{(t)}) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(t)})^\top \boldsymbol{H}(f)(\boldsymbol{x}^{(t)})(\boldsymbol{x} - \boldsymbol{x}^{(t)})$$

  for $\boldsymbol{x}$ close enough to $\boldsymbol{x}^{(t)}$
  - $\boldsymbol{H}(f)(\boldsymbol{x}^{(t)}) \in \mathbb{R}^{d\times d}$ is the (symmetric) Hessian matrix of f at $\boldsymbol{x}^{(t)}$
- Line search at step t:

$$\arg\min_\eta \tilde{f}(\boldsymbol{x}^{(t)} - \eta \nabla f(\boldsymbol{x}^{(t)})) = \arg\min_\eta f(\boldsymbol{x}^{(t)}) - \eta \nabla f(\boldsymbol{x}^{(t)})^\top \nabla f(\boldsymbol{x}^{(t)}) + \frac{\eta^2}{2} \nabla f(\boldsymbol{x}^{(t)})^\top \boldsymbol{H}(f)(\boldsymbol{x}^{(t)}) \nabla f(\boldsymbol{x}^{(t)})$$

- If $f(\boldsymbol{x}^{(t)})^\top \boldsymbol{H}(f)(\boldsymbol{x}^{(t)}) \nabla f(\boldsymbol{x}^{(t)}) > 0$, we can solve $\frac{\partial}{\partial \eta}\tilde{f}(\boldsymbol{x}^{(t)} - \eta \nabla f(\boldsymbol{x}^{(t)})) = 0$ and get:

$$\eta^{(t)} = \frac{\nabla f(\boldsymbol{x}^{(t)})^\top \nabla f(\boldsymbol{x}^{(t)})}{\nabla f(\boldsymbol{x}^{(t)})^\top \boldsymbol{H}(f)(\boldsymbol{x}^{(t)}) \nabla f(\boldsymbol{x}^{(t)})}$$

Problems of Gradient Descent

- Gradient descent is designed to find the steepest descent direction at step $\boldsymbol{x}^{(t)}$
  - ***Not aware of the conditioning*** of the Hessian matrix $\boldsymbol{H}(f)(\boldsymbol{x}^{(t)})$

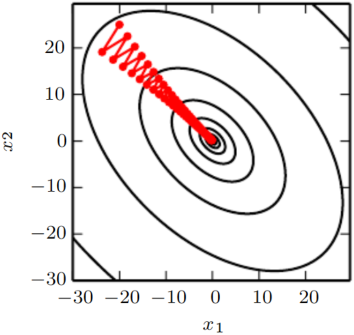
Problems of Gradient Descent

- Gradient descent is designed to find the steepest descent direction at step $\boldsymbol{x}^{(t)}$
  - ***Not aware of the conditioning*** of the Hessian matrix $\boldsymbol{H}(f)(\boldsymbol{x}^{(t)})$
- If $\boldsymbol{H}(f)(\boldsymbol{x}^{(t)})$ has a large condition number, then f is curvy in some directions but flat in others at $\boldsymbol{x}^{(t)}$

Problems of Gradient Descent

- Gradient descent is designed to find the steepest descent direction at step $\boldsymbol{x}^{(t)}$
  - ***Not aware of the conditioning*** of the Hessian matrix $\boldsymbol{H}(f)(\boldsymbol{x}^{(t)})$
- If $\boldsymbol{H}(f)(\boldsymbol{x}^{(t)})$ has a large condition number, then f is curvy in some directions but flat in others at $\boldsymbol{x}^{(t)}$

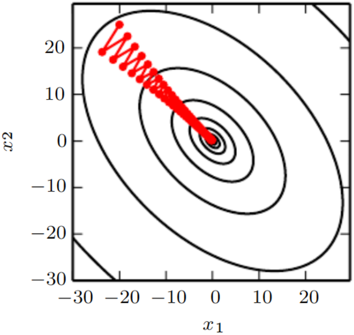
- E.g., suppose f is a quadratic function whose Hessian has a large condition number
- A step in gradient descent may overshoot the optimal points along flat attributes
  - "Zig-zags" around a narrow valley



Problems of Gradient Descent

- Gradient descent is designed to find the steepest descent direction at step $\boldsymbol{x}^{(t)}$
  - ***Not aware of the conditioning*** of the Hessian matrix $\boldsymbol{H}(f)(\boldsymbol{x}^{(t)})$
- If $\boldsymbol{H}(f)(\boldsymbol{x}^{(t)})$ has a large condition number, then f is curvy in some directions but flat in others at $\boldsymbol{x}^{(t)}$

- E.g., suppose f is a quadratic function whose Hessian has a large condition number
- A step in gradient descent may overshoot the optimal points along flat attributes
  - "Zig-zags" around a narrow valley
- Why not take conditioning into account when picking descent directions?



Outline

Newton's Method I

- By Taylor's theorem, we can approximate f locally at point $\boldsymbol{x}^{(t)}$ using a quadratic function $\tilde{f}$, i.e.,

$$f(\boldsymbol{x}) \approx \tilde{f}(\boldsymbol{x};\boldsymbol{x}^{(t)}) = f(\boldsymbol{x}^{(t)}) + \nabla f(\boldsymbol{x}^{(t)})^\top (\boldsymbol{x} - \boldsymbol{x}^{(t)}) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(t)})^\top \boldsymbol{H}(f)(\boldsymbol{x}^{(t)})(\boldsymbol{x} - \boldsymbol{x}^{(t)})$$

  for $\boldsymbol{x}$ close enough to $\boldsymbol{x}^{(t)}$

Newton's Method I

- By Taylor's theorem, we can approximate f locally at point $\boldsymbol{x}^{(t)}$ using a quadratic function $\tilde{f}$, i.e.,

$$f(\boldsymbol{x}) \approx \tilde{f}(\boldsymbol{x};\boldsymbol{x}^{(t)}) = f(\boldsymbol{x}^{(t)}) + \nabla f(\boldsymbol{x}^{(t)})^\top(\boldsymbol{x}-\boldsymbol{x}^{(t)}) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}^{(t)})^\top \boldsymbol{H}(f)(\boldsymbol{x}^{(t)})(\boldsymbol{x}-\boldsymbol{x}^{(t)})$$

  for $\boldsymbol{x}$ close enough to $\boldsymbol{x}^{(t)}$
- If f is strictly convex (i.e., $\boldsymbol{H}(f)(\boldsymbol{a}) \succ \boldsymbol{O}, \forall \boldsymbol{a}$), we can find $\boldsymbol{x}^{(t+1)}$ that minimizes $\tilde{f}$ in order to decrease f
- Solving $\nabla\tilde{f}(\boldsymbol{x}^{(t+1)};\boldsymbol{x}^{(t)}) = \boldsymbol{0}$, we have

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \boldsymbol{H}(f)(\boldsymbol{x}^{(t)})^{-1}\nabla f(\boldsymbol{x}^{(t)})$$

  - $\boldsymbol{H}(f)(\boldsymbol{x}^{(t)})^{-1}$ as a "corrector" to the negative gradient

Newton's Method II

Input: $\boldsymbol{x}^{(0)} \in \mathbb{R}^d$ an initial guess, $\eta > 0$
repeat
$\quad \boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} - \eta \boldsymbol{H}(f)(\boldsymbol{x}^{(t)})^{-1} \nabla f(\boldsymbol{x}^{(t)})$;
until *convergence criterion is satisfied*;

Newton's Method II

Input: $\boldsymbol{x}^{(0)} \in \mathbb{R}^d$ an initial guess, $\eta > 0$

repeat

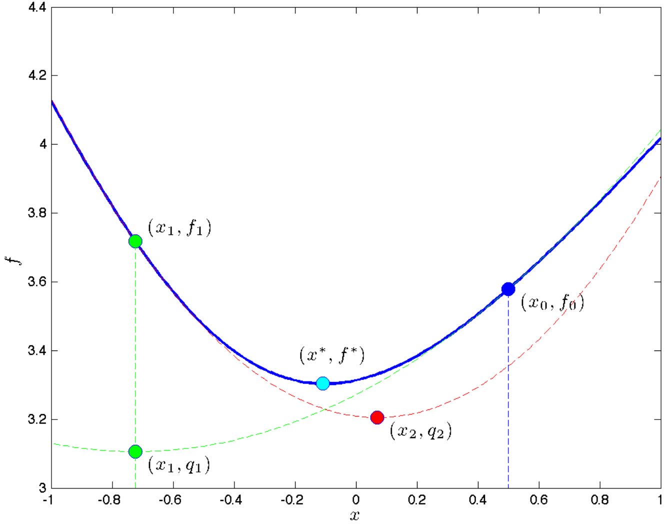
$\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} - \eta \boldsymbol{H}(f)(\boldsymbol{x}^{(t)})^{-1} \nabla f(\boldsymbol{x}^{(t)})$;

until *convergence criterion is satisfied*;

- In practice, we multiply the shift by a small $\eta > 0$ to make sure that $\boldsymbol{x}^{(t+1)}$ is close to $\boldsymbol{x}^{(t)}$

Newton's Method III

- If f is positive definite quadratic, then only one step is required



General Functions

- Update rule: $\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} - \eta \boldsymbol{H}(f)(\boldsymbol{x}^{(t)})^{-1} \nabla f(\boldsymbol{x}^{(t)})$
- What if f is not strictly convex?
  - $\boldsymbol{H}(f)(\boldsymbol{x}^{(t)}) \preceq \boldsymbol{O}$ or indefinite

General Functions

- Update rule: $\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} - \eta \boldsymbol{H}(f)(\boldsymbol{x}^{(t)})^{-1}\nabla f(\boldsymbol{x}^{(t)})$
- What if f is not strictly convex?
  - $\boldsymbol{H}(f)(\boldsymbol{x}^{(t)}) \preceq \boldsymbol{O}$ or indefinite
- The ***Levenberg–Marquardt extension***:

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \eta \left(\boldsymbol{H}(f)(\boldsymbol{x}^{(t)}) + \alpha \boldsymbol{I}\right)^{-1} \nabla f(\boldsymbol{x}^{(t)}) \text{ for some } \alpha > 0$$

  - With a large α, degenerates into gradient descent of learning rate $1/\alpha$

General Functions

- Update rule: $\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} - \eta \boldsymbol{H}(f)(\boldsymbol{x}^{(t)})^{-1} \nabla f(\boldsymbol{x}^{(t)})$
- What if f is not strictly convex?
  - $\boldsymbol{H}(f)(\boldsymbol{x}^{(t)}) \preceq \boldsymbol{O}$ or indefinite
- The ***Levenberg–Marquardt extension***:

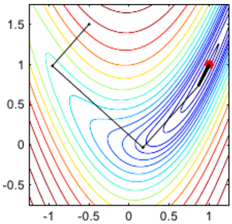
$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \eta \left(\boldsymbol{H}(f)(\boldsymbol{x}^{(t)}) + \alpha \boldsymbol{I}\right)^{-1} \nabla f(\boldsymbol{x}^{(t)}) \text{ for some } \alpha > 0$$

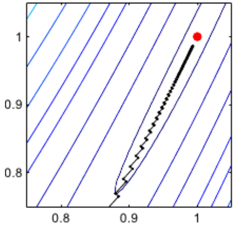
  - With a large α, degenerates into gradient descent of learning rate $1/\alpha$

Input: $\boldsymbol{x}^{(0)} \in \mathbb{R}^d$ an initial guess, $\eta > 0$, $\alpha > 0$
repeat
$\quad \boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} - \eta \left(\boldsymbol{H}(f)(\boldsymbol{x}^{(t)}) + \alpha \boldsymbol{I}\right)^{-1} \nabla f(\boldsymbol{x}^{(t)})$;
until *convergence criterion is satisfied*;

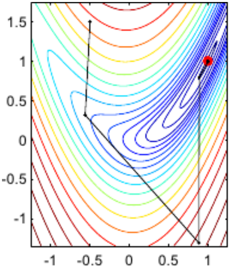
Gradient Descent vs. Newton's Method

- Steps of Gradient descent when f is a Rosenbrock's banana:





- Steps of Newton's method:
  - Only 6 steps in total



Problems of Newton's Method

- Computing $\boldsymbol{H}(f)(\boldsymbol{x}^{(t)})^{-1}$ is slow
  - Takes $O(d^3)$ time at each step, which is ***much slower*** then $O(d)$ of gradient descent

Problems of Newton's Method

- Computing $\boldsymbol{H}(f)(\boldsymbol{x}^{(t)})^{-1}$ is slow
  - Takes $O(d^3)$ time at each step, which is ***much slower*** then $O(d)$ of gradient descent
- Imprecise $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \eta \boldsymbol{H}(f)(\boldsymbol{x}^{(t)})^{-1} \nabla f(\boldsymbol{x}^{(t)})$ due to numerical errors
  - $\boldsymbol{H}(f)(\boldsymbol{x}^{(t)})$ may have a large condition number

Problems of Newton's Method

- Computing $\boldsymbol{H}(f)(\boldsymbol{x}^{(t)})^{-1}$ is slow
  - Takes $O(d^3)$ time at each step, which is ***much slower*** then $O(d)$ of gradient descent
- Imprecise $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \eta \boldsymbol{H}(f)(\boldsymbol{x}^{(t)})^{-1} \nabla f(\boldsymbol{x}^{(t)})$ due to numerical errors
  - $\boldsymbol{H}(f)(\boldsymbol{x}^{(t)})$ may have a large condition number
- Attracted to ***saddle points*** (when f is not convex)
  - The $\boldsymbol{x}^{(t+1)}$ solved from $\nabla \tilde{f}(\boldsymbol{x}^{(t+1)}; \boldsymbol{x}^{(t)}) = \boldsymbol{0}$ is a critical point

Outline

Who is Afraid of Non-convexity?

- In ML, the function to solve is usually the cost function $C(\boldsymbol{w})$ of a model $\mathbb{F} = \{f : f \text{ parametrized by } \boldsymbol{w}\}$

Who is Afraid of Non-convexity?

- In ML, the function to solve is usually the cost function $C(\boldsymbol{w})$ of a model $\mathbb{F} = \{f : f \text{ parametrized by } \boldsymbol{w}\}$
- Many ML models have convex cost functions in order to take advantages of convex optimization
  - E.g., perceptron, linear regression, logistic regression, SVMs, etc.

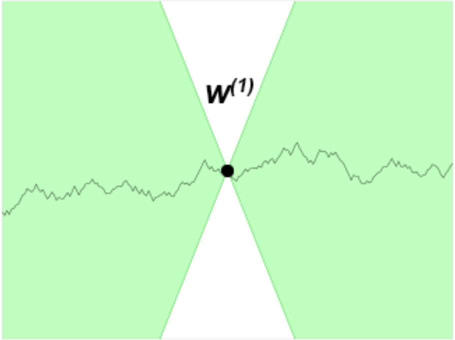
Who is Afraid of Non-convexity?

- In ML, the function to solve is usually the cost function $C(\boldsymbol{w})$ of a model $\mathbb{F} = \{f : f \text{ parametrized by } \boldsymbol{w}\}$
- Many ML models have convex cost functions in order to take advantages of convex optimization
  - E.g., perceptron, linear regression, logistic regression, SVMs, etc.
- However, in deep learning, the cost function of a neural network is typically ***not*** convex
  - We will discuss techniques that tackle non-convexity later

Assumption on Cost Functions

- In ML, we usually assume that the (real-valued) cost function is ***Lipschitz continuous*** and/or have Lipschitz continuous derivatives
- I.e., the rate of change of C if bounded by a ***Lipschitz constant*** K:

$$|C(\boldsymbol{w}^{(1)}) - C(\boldsymbol{w}^{(2)})| \leq K\|\boldsymbol{w}^{(1)} - \boldsymbol{w}^{(2)}\|, \forall \boldsymbol{w}^{(1)}, \boldsymbol{w}^{(2)}$$



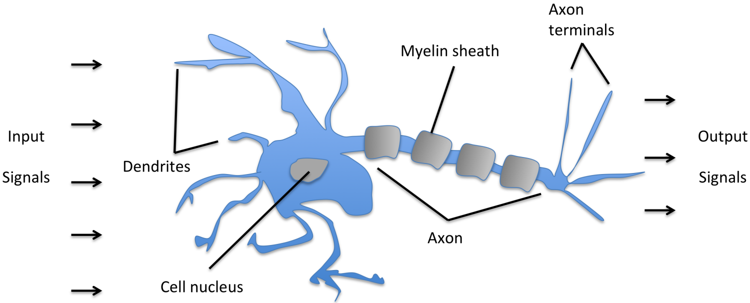
Outline

Perceptron & Neurons

- ***Perceptron***, proposed in 1950's by Rosenblatt, is one of the first ML algorithms for binary classification

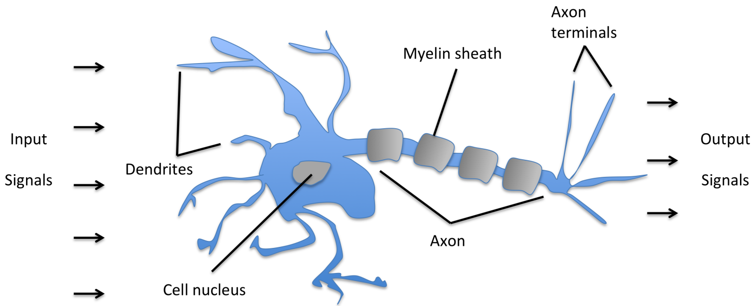
Perceptron & Neurons

- ***Perceptron***, proposed in 1950's by Rosenblatt, is one of the first ML algorithms for binary classification
- Inspired by McCullock-Pitts (MCP) neuron, published in 1943



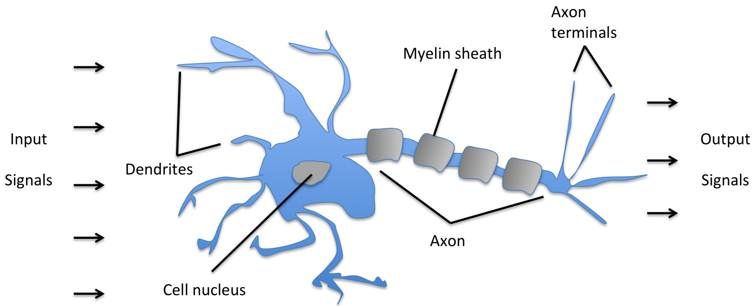
Perceptron & Neurons

- ***Perceptron***, proposed in 1950's by Rosenblatt, is one of the first ML algorithms for binary classification
- Inspired by McCullock-Pitts (MCP) neuron, published in 1943
  - Our brains consist of interconnected ***neurons***



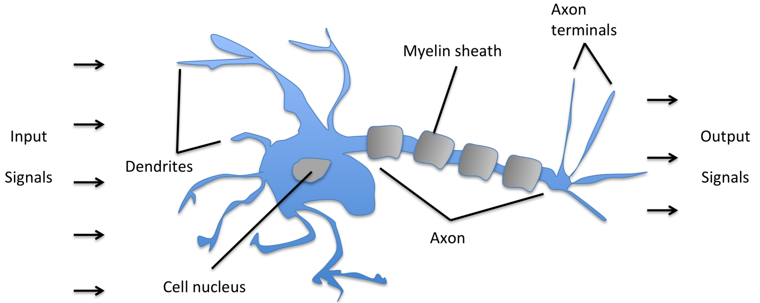
Perceptron & Neurons

- ***Perceptron***, proposed in 1950's by Rosenblatt, is one of the first ML algorithms for binary classification
- Inspired by McCullock-Pitts (MCP) neuron, published in 1943
  - Our brains consist of interconnected ***neurons***
  - Each neuron takes signals from other neurons as input



Perceptron & Neurons

- ***Perceptron***, proposed in 1950's by Rosenblatt, is one of the first ML algorithms for binary classification
- Inspired by McCullock-Pitts (MCP) neuron, published in 1943
  - Our brains consist of interconnected ***neurons***
  - Each neuron takes signals from other neurons as input
  - If the accumulated signal exceeds a certain threshold, an output signal is generated

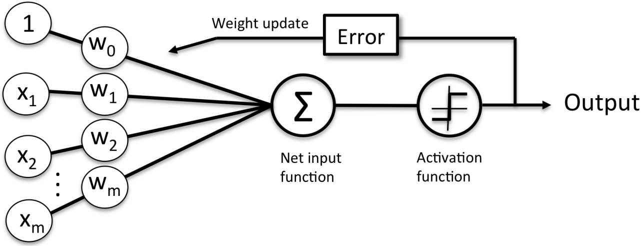


Model

- Binary classification problem:
  - Training dataset: $\mathbb{X} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_i$, where $\boldsymbol{x}^{(i)} \in \mathbb{R}^D$ and $y^{(i)} \in \{1, -1\}$
  - Output: a function $f(\boldsymbol{x}) = \hat{y}$ such that $\hat{y}$ is close to the true label y

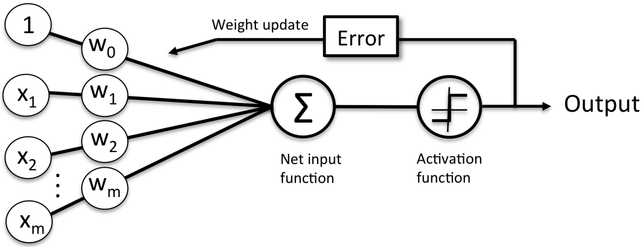
Model

- Binary classification problem:
  - Training dataset: $\mathbb{X} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_i$, where $\boldsymbol{x}^{(i)} \in \mathbb{R}^D$ and $y^{(i)} \in \{1, -1\}$
  - Output: a function $f(\boldsymbol{x}) = \hat{y}$ such that $\hat{y}$ is close to the true label y
- Model: $\{f : f(\boldsymbol{x}; \boldsymbol{w}, b) = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x} - b)\}$
  - $\text{sign}(a) = 1$ if $a \geq 0$; otherwise 0



Model

- Binary classification problem:
  - Training dataset: $\mathbb{X} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_i$, where $\boldsymbol{x}^{(i)} \in \mathbb{R}^D$ and $y^{(i)} \in \{1, -1\}$
  - Output: a function $f(\boldsymbol{x}) = \hat{y}$ such that $\hat{y}$ is close to the true label y
- Model: $\{f : f(\boldsymbol{x}; \boldsymbol{w}, b) = \mathrm{sign}(\boldsymbol{w}^\top \boldsymbol{x} - b)\}$
  - $\mathrm{sign}(a) = 1$ if $a \geq 0$; otherwise 0
  - For simplicity, we use shorthand $f(\boldsymbol{x}; \boldsymbol{w}) = \mathrm{sign}(\boldsymbol{w}^\top \boldsymbol{x})$ where $\boldsymbol{w} = [-b, w_1, \cdots, w_D]^\top$ and $\boldsymbol{x} = [1, x_1, \cdots, x_D]^\top$

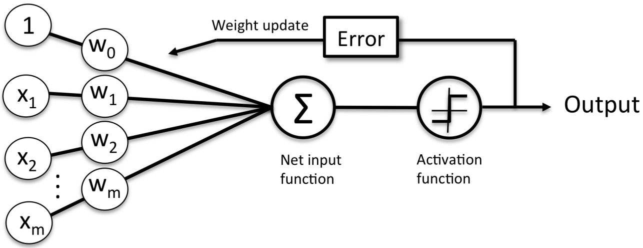


Iterative Training Algorithm I

1. Initiate $\boldsymbol{w}^{(0)}$ and learning rate $\eta > 0$
2. Epoch: for each example $(\boldsymbol{x}^{(t)}, y^{(t)})$, update $\boldsymbol{w}$ by

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \eta(y^{(t)} - \hat{y}^{(t)})\boldsymbol{x}^{(t)}$$

   where $\hat{y}^{(t)} = f(\boldsymbol{x}^{(t)}; \boldsymbol{w}^{(t)}) = \text{sign}(\boldsymbol{w}^{(t)\top}\boldsymbol{x}^{(t)})$
3. Repeat epoch several times (or until converge)



Iterative Training Algorithm II

- Update rule:
$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \eta(y^{(t)} - \hat{y}^{(t)})\boldsymbol{x}^{(t)}$$
- If $\hat{y}^{(t)}$ is correct, we have $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)}$

Iterative Training Algorithm II

- Update rule:
$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \eta(y^{(t)} - \hat{y}^{(t)})\boldsymbol{x}^{(t)}$$
- If $\hat{y}^{(t)}$ is correct, we have $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)}$
- If $\hat{y}^{(t)}$ is incorrect, we have $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + 2\eta y^{(t)}\boldsymbol{x}^{(t)}$

Iterative Training Algorithm II

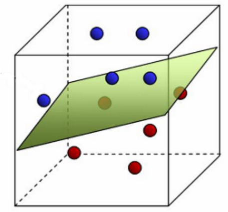
- Update rule:
$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \eta(y^{(t)} - \hat{y}^{(t)})\boldsymbol{x}^{(t)}$$
- If $\hat{y}^{(t)}$ is correct, we have $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)}$
- If $\hat{y}^{(t)}$ is incorrect, we have $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + 2\eta y^{(t)}\boldsymbol{x}^{(t)}$
  - If $y^{(t)} = 1$, the updated prediction will more likely to be positive, as $\mathrm{sign}(\boldsymbol{w}^{(t+1)\top}\boldsymbol{x}^{(t)}) = \mathrm{sign}(\boldsymbol{w}^{(t)\top}\boldsymbol{x}^{(t)} + c)$ for some $c > 0$

Iterative Training Algorithm II

- Update rule:
$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \eta(y^{(t)} - \hat{y}^{(t)})\boldsymbol{x}^{(t)}$$
- If $\hat{y}^{(t)}$ is correct, we have $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)}$
- If $\hat{y}^{(t)}$ is incorrect, we have $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + 2\eta y^{(t)}\boldsymbol{x}^{(t)}$
  - If $y^{(t)} = 1$, the updated prediction will more likely to be positive, as $\text{sign}(\boldsymbol{w}^{(t+1)\top}\boldsymbol{x}^{(t)}) = \text{sign}(\boldsymbol{w}^{(t)\top}\boldsymbol{x}^{(t)} + c)$ for some $c > 0$
  - If $y^{(t)} = -1$, the updated prediction will more likely to be negative

Iterative Training Algorithm II

- Update rule:
$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \eta(y^{(t)} - \hat{y}^{(t)})\boldsymbol{x}^{(t)}$$
- If $\hat{y}^{(t)}$ is correct, we have $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)}$
- If $\hat{y}^{(t)}$ is incorrect, we have $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + 2\eta y^{(t)}\boldsymbol{x}^{(t)}$
  - If $y^{(t)} = 1$, the updated prediction will more likely to be positive, as $\mathrm{sign}(\boldsymbol{w}^{(t+1)\top}\boldsymbol{x}^{(t)}) = \mathrm{sign}(\boldsymbol{w}^{(t)\top}\boldsymbol{x}^{(t)} + c)$ for some $c > 0$
  - If $y^{(t)} = -1$, the updated prediction will more likely to be negative
- Does ***not*** converge if the dataset cannot be separated by a hyperplane



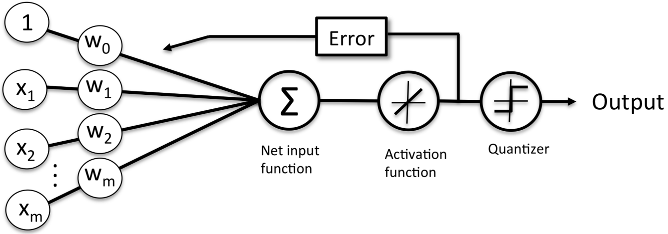
Outline

ADAptive LInear NEuron (Adaline)

- Proposed in 1960's by Widrow et al.
- Defines and minimizes a ***cost function*** for training:

$$\arg\min_{\boldsymbol{w}} C(\boldsymbol{w};\mathbb{X}) = \arg\min_{\boldsymbol{w}} \frac{1}{2}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{w}^\top \boldsymbol{x}^{(i)}\right)^2$$

  - Links numerical optimization to ML

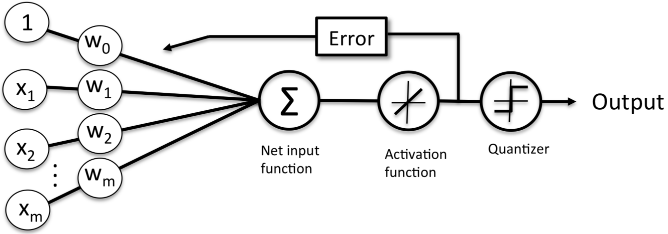


ADAptive LInear NEuron (Adaline)

- Proposed in 1960's by Widrow et al.
- Defines and minimizes a ***cost function*** for training:

$$\arg\min_{\boldsymbol{w}} C(\boldsymbol{w};\mathbb{X}) = \arg\min_{\boldsymbol{w}} \frac{1}{2}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{w}^\top \boldsymbol{x}^{(i)}\right)^2$$

  - Links numerical optimization to ML
- Sign function is only used for binary prediction ***after*** training



Training Using Gradient Descent

- Update rule:

$$\begin{aligned} \boldsymbol{w}^{(t+1)} &= \boldsymbol{w}^{(t)} - \eta \nabla C(\boldsymbol{w}^{(t)}) \\ &= \boldsymbol{w}^{(t)} + \eta \sum_i (y^{(i)} - \boldsymbol{w}^{(t)\top} \boldsymbol{x}^{(i)}) \boldsymbol{x}^{(i)} \end{aligned}$$

Training Using Gradient Descent

- Update rule:

$$\begin{aligned} \boldsymbol{w}^{(t+1)} &= \boldsymbol{w}^{(t)} - \eta \nabla C(\boldsymbol{w}^{(t)}) \\ &= \boldsymbol{w}^{(t)} + \eta \sum_i (y^{(i)} - \boldsymbol{w}^{(t)\top} \boldsymbol{x}^{(i)}) \boldsymbol{x}^{(i)} \end{aligned}$$

- Since the cost function is convex, the training iterations will converge

Outline

Cost as an Expectation

- In ML, the cost function to minimize is usually a sum of ***losses*** over training examples
- E.g., in Adaline: sum of square losses (functions)

$$\arg\min_{\boldsymbol{w}} C(\boldsymbol{w};\mathbb{X}) = \arg\min_{\boldsymbol{w}} \frac{1}{2}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{w}^\top \boldsymbol{x}^{(i)}\right)^2$$

Cost as an Expectation

- In ML, the cost function to minimize is usually a sum of ***losses*** over training examples
- E.g., in Adaline: sum of square losses (functions)

$$\arg\min_{\boldsymbol{w}} C(\boldsymbol{w};\mathbb{X}) = \arg\min_{\boldsymbol{w}} \frac{1}{2}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{w}^\top \boldsymbol{x}^{(i)}\right)^2$$

- Let examples be i.i.d. samples of random variables $(\mathbf{x}, \mathrm{y})$
- We effectively minimize the estimate of $\mathrm{E}[C(\boldsymbol{w})]$ ***over the distribution*** $\mathrm{P}(\mathbf{x}, \mathrm{y})$:

$$\arg\min_{\boldsymbol{w}} \mathrm{E}_{\mathbf{x},\mathrm{y}\sim\mathrm{P}}[C(\boldsymbol{w})]$$

  - $\mathrm{P}(\mathbf{x}, \mathrm{y})$ may be unknown

Cost as an Expectation

- In ML, the cost function to minimize is usually a sum of ***losses*** over training examples
- E.g., in Adaline: sum of square losses (functions)

$$\arg\min_{\boldsymbol{w}} C(\boldsymbol{w};\mathbb{X}) = \arg\min_{\boldsymbol{w}} \frac{1}{2}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{w}^\top \boldsymbol{x}^{(i)}\right)^2$$

- Let examples be i.i.d. samples of random variables $(\mathbf{x}, \mathrm{y})$
- We effectively minimize the estimate of $\mathrm{E}[C(\boldsymbol{w})]$ ***over the distribution*** $\mathrm{P}(\mathbf{x}, \mathrm{y})$:

$$\arg\min_{\boldsymbol{w}} \mathrm{E}_{\mathbf{x},\mathrm{y}\sim\mathrm{P}}[C(\boldsymbol{w})]$$

  - $\mathrm{P}(\mathbf{x}, \mathrm{y})$ may be unknown
- Since the problem is stochastic by nature, why not make the training algorithm stochastic too?

Stochastic Gradient Descent

Input: $w^{(0)} \in \mathbb{R}^d$ an initial guess, $\eta > 0$, $M \geq 1$
repeat
  epoch:
  Randomly partition the training set $\mathbb{X}$ into the ***minibatches*** $\{\mathbb{X}^{(j)}\}_j$, $|\mathbb{X}^{(j)}| = M$;
  foreach j **do**
    $w^{(t+1)} \leftarrow w^{(t)} - \eta \nabla C(w^{(t)}; \mathbb{X}^{(j)})$;
  end
until *convergence criterion is satisfied*;

Stochastic Gradient Descent

Input: $\boldsymbol{w}^{(0)} \in \mathbb{R}^d$ an initial guess, $\eta > 0$, $M \geq 1$
repeat
  epoch:
  Randomly partition the training set $\mathbb{X}$ into the ***minibatches*** $\{\mathbb{X}^{(j)}\}_j$, $|\mathbb{X}^{(j)}| = M$;
  foreach j **do**
    $\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \eta \nabla C(\boldsymbol{w}^{(t)}; \mathbb{X}^{(j)})$;
  end
until *convergence criterion is satisfied*;

- $C(\boldsymbol{w}; \mathbb{X}^{(j)})$ is still an estimate of $\mathrm{E}_{\mathbf{x},\mathrm{y}\sim\mathrm{P}}[C(\boldsymbol{w})]$
  - $\mathbb{X}^{(j)}$ are samples of the same distribution $\mathrm{P}(\mathbf{x}, \mathrm{y})$
- It's common to set $M = 1$ on a single machine
  - E.g., update rule for Adaline: $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \eta(y^{(t)} - \boldsymbol{w}^{(t)\top}\boldsymbol{x}^{(t)})\boldsymbol{x}^{(t)}$, which is similar to that of Perceptron

Stochastic Gradient Descent

Input: $\boldsymbol{w}^{(0)} \in \mathbb{R}^d$ an initial guess, $\eta > 0$, $M \geq 1$
repeat
  epoch:
  Randomly partition the training set $\mathbb{X}$ into the ***minibatches*** $\{\mathbb{X}^{(j)}\}_j$, $|\mathbb{X}^{(j)}| = M$;
  foreach j **do**
    $\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \eta \nabla C(\boldsymbol{w}^{(t)}; \mathbb{X}^{(j)})$;
  end
until *convergence criterion is satisfied*;

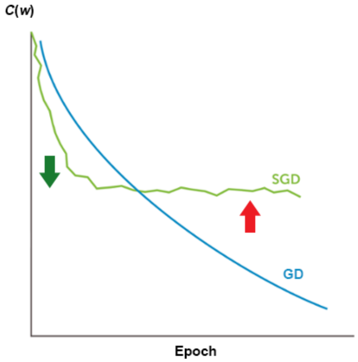
- $C(\boldsymbol{w}; \mathbb{X}^{(j)})$ is still an estimate of $\mathrm{E}_{\mathbf{x},\mathrm{y}\sim\mathrm{P}}[C(\boldsymbol{w})]$
  - $\mathbb{X}^{(j)}$ are samples of the same distribution $\mathrm{P}(\mathbf{x}, \mathrm{y})$
- It's common to set $M = 1$ on a single machine
  - E.g., update rule for Adaline: $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \eta(y^{(t)} - \boldsymbol{w}^{(t)\top}\boldsymbol{x}^{(t)})\boldsymbol{x}^{(t)}$, which is similar to that of Perceptron

SGD vs. GD

- Each iteration can run ***much faster*** when $M \ll N$

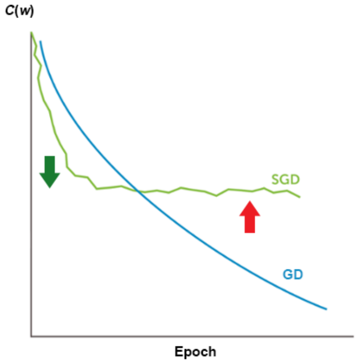
SGD vs. GD

- Each iteration can run ***much faster*** when $M \ll N$
- Converges faster (in both #epochs and time) with large datasets



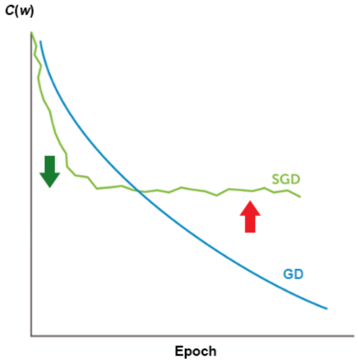
SGD vs. GD

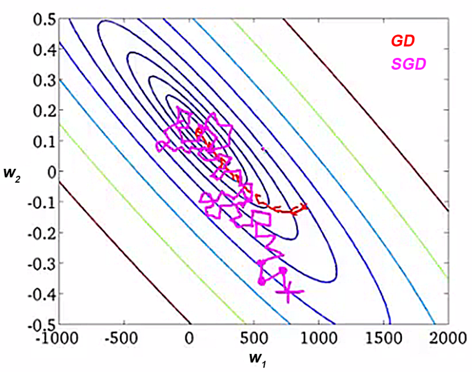
- Each iteration can run ***much faster*** when $M \ll N$
- Converges faster (in both #epochs and time) with large datasets
- Supports ***online*** learning



SGD vs. GD

- Each iteration can run ***much faster*** when $M \ll N$
- Converges faster (in both #epochs and time) with large datasets
- Supports ***online*** learning
- But may wander around the optimal points
  - In practice, we set $\eta = O(t^{-1})$





Outline

Constrained Optimization

- Problem:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$
$$\text{subject to } \boldsymbol{x} \in \mathbb{C}$$

- $f : \mathbb{R}^d \to \mathbb{R}$ is not necessarily convex
- $\mathbb{C} = \{\boldsymbol{x} : g^{(i)}(\boldsymbol{x}) \leq 0, h^{(j)}(\boldsymbol{x}) = 0\}_{i,j}$

Constrained Optimization

- Problem:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$
$$\text{subject to } \boldsymbol{x} \in \mathbb{C}$$

- $f : \mathbb{R}^d \to \mathbb{R}$ is not necessarily convex
- $\mathbb{C} = \{\boldsymbol{x} : g^{(i)}(\boldsymbol{x}) \leq 0, h^{(j)}(\boldsymbol{x}) = 0\}_{i,j}$
- Iterative descent algorithm?

Common Methods

- ***Projective gradient descent***: if $\boldsymbol{x}^{(t)}$ falls outside $\mathbb{C}$ at step t, we "project" back the point to the tangent space (edge) of $\mathbb{C}$

Common Methods

- ***Projective gradient descent***: if $\boldsymbol{x}^{(t)}$ falls outside $\mathbb{C}$ at step t, we "project" back the point to the tangent space (edge) of $\mathbb{C}$
- ***Penalty/barrier methods***: convert the constrained problem into one or more unconstrained ones
- And more...

Karush-Kuhn-Tucker (KKT) Methods I

- Converts the problem

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$
$$\text{subject to } \boldsymbol{x} \in \{\boldsymbol{x} : g^{(i)}(\boldsymbol{x}) \leq 0, h^{(j)}(\boldsymbol{x}) = 0\}_{i,j}$$

into

$$\min_{\boldsymbol{x}} \max_{\alpha,\beta,\alpha \geq \mathbf{0}} L(\boldsymbol{x}, \alpha, \beta) =$$
$$\min_{\boldsymbol{x}} \max_{\alpha,\beta,\alpha \geq \mathbf{0}} f(\boldsymbol{x}) + \sum_i \alpha_i g^{(i)}(\boldsymbol{x}) + \sum_j \beta_j h^{(j)}(\boldsymbol{x})$$

Karush-Kuhn-Tucker (KKT) Methods I

- Converts the problem

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$
$$\text{subject to } \boldsymbol{x} \in \{\boldsymbol{x} : g^{(i)}(\boldsymbol{x}) \leq 0, h^{(j)}(\boldsymbol{x}) = 0\}_{i,j}$$

  into

$$\min_{\boldsymbol{x}} \max_{\alpha,\beta,\alpha \geq \mathbf{0}} L(\boldsymbol{x}, \alpha, \beta) =$$
$$\min_{\boldsymbol{x}} \max_{\alpha,\beta,\alpha \geq \mathbf{0}} f(\boldsymbol{x}) + \sum_i \alpha_i g^{(i)}(\boldsymbol{x}) + \sum_j \beta_j h^{(j)}(\boldsymbol{x})$$

- $\min_{\boldsymbol{x}} \max_{\alpha,\beta} L$ means "minimize L with respect to $\boldsymbol{x}$, at which L is maximized with respect to α and β"

Karush-Kuhn-Tucker (KKT) Methods I

- Converts the problem

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$
$$\text{subject to } \boldsymbol{x} \in \{\boldsymbol{x} : g^{(i)}(\boldsymbol{x}) \leq 0, h^{(j)}(\boldsymbol{x}) = 0\}_{i,j}$$

  into

$$\min_{\boldsymbol{x}} \max_{\alpha,\beta,\alpha\geq\mathbf{0}} L(\boldsymbol{x},\alpha,\beta) =$$
$$\min_{\boldsymbol{x}} \max_{\alpha,\beta,\alpha\geq\mathbf{0}} f(\boldsymbol{x}) + \sum_i \alpha_i g^{(i)}(\boldsymbol{x}) + \sum_j \beta_j h^{(j)}(\boldsymbol{x})$$

- $\min_{\boldsymbol{x}} \max_{\alpha,\beta} L$ means "minimize L with respect to $\boldsymbol{x}$, at which L is maximized with respect to α and β"
- The function $L(\boldsymbol{x},\alpha,\beta)$ is called the (generalized) ***Lagrangian***

Karush-Kuhn-Tucker (KKT) Methods I

- Converts the problem

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$
$$\text{subject to } \boldsymbol{x} \in \{\boldsymbol{x} : g^{(i)}(\boldsymbol{x}) \leq 0, h^{(j)}(\boldsymbol{x}) = 0\}_{i,j}$$

  into

$$\min_{\boldsymbol{x}} \max_{\alpha,\beta,\alpha \geq \boldsymbol{0}} L(\boldsymbol{x}, \alpha, \beta) =$$
$$\min_{\boldsymbol{x}} \max_{\alpha,\beta,\alpha \geq \boldsymbol{0}} f(\boldsymbol{x}) + \sum_i \alpha_i g^{(i)}(\boldsymbol{x}) + \sum_j \beta_j h^{(j)}(\boldsymbol{x})$$

- $\min_{\boldsymbol{x}} \max_{\alpha,\beta} L$ means "minimize L with respect to $\boldsymbol{x}$, at which L is maximized with respect to α and β"
- The function $L(\boldsymbol{x}, \alpha, \beta)$ is called the (generalized) ***Lagrangian***
- α and β are called ***KKT multipliers***

Karush-Kuhn-Tucker (KKT) Methods II

- Converts the problem

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$
$$\text{subject to } \boldsymbol{x} \in \{\boldsymbol{x} : g^{(i)}(\boldsymbol{x}) \leq 0, h^{(j)}(\boldsymbol{x}) = 0\}_{i,j}$$

  into

$$\min_{\boldsymbol{x}} \max_{\alpha,\beta,\alpha\geq\mathbf{0}} L(\boldsymbol{x},\alpha,\beta) =$$
$$\min_{\boldsymbol{x}} \max_{\alpha,\beta,\alpha\geq\mathbf{0}} f(\boldsymbol{x}) + \sum_i \alpha_i g^{(i)}(\boldsymbol{x}) + \sum_j \beta_j h^{(j)}(\boldsymbol{x})$$

Karush-Kuhn-Tucker (KKT) Methods II

- Converts the problem

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$
$$\text{subject to } \boldsymbol{x} \in \{\boldsymbol{x} : g^{(i)}(\boldsymbol{x}) \leq 0, h^{(j)}(\boldsymbol{x}) = 0\}_{i,j}$$

  into

$$\min_{\boldsymbol{x}} \max_{\alpha,\beta,\alpha\geq\mathbf{0}} L(\boldsymbol{x},\alpha,\beta) =$$
$$\min_{\boldsymbol{x}} \max_{\alpha,\beta,\alpha\geq\mathbf{0}} f(\boldsymbol{x}) + \sum_i \alpha_i g^{(i)}(\boldsymbol{x}) + \sum_j \beta_j h^{(j)}(\boldsymbol{x})$$

- Observe that for any feasible point $\boldsymbol{x}$, we have

$$\max_{\alpha,\beta,\alpha\geq\mathbf{0}} L(\boldsymbol{x},\alpha,\beta) = f(\boldsymbol{x})$$

  - The optimal feasible point is unchanged

Karush-Kuhn-Tucker (KKT) Methods II

- Converts the problem

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$
$$\text{subject to } \boldsymbol{x} \in \{\boldsymbol{x} : g^{(i)}(\boldsymbol{x}) \leq 0, h^{(j)}(\boldsymbol{x}) = 0\}_{i,j}$$

  into

$$\min_{\boldsymbol{x}} \max_{\alpha,\beta,\alpha \geq \mathbf{0}} L(\boldsymbol{x}, \alpha, \beta) =$$
$$\min_{\boldsymbol{x}} \max_{\alpha,\beta,\alpha \geq \mathbf{0}} f(\boldsymbol{x}) + \sum_i \alpha_i g^{(i)}(\boldsymbol{x}) + \sum_j \beta_j h^{(j)}(\boldsymbol{x})$$

- Observe that for any feasible point $\boldsymbol{x}$, we have

$$\max_{\alpha,\beta,\alpha \geq \mathbf{0}} L(\boldsymbol{x}, \alpha, \beta) = f(\boldsymbol{x})$$

  - The optimal feasible point is unchanged
- And for any infeasible point $\boldsymbol{x}$, we have

$$\max_{\alpha,\beta,\alpha \geq \mathbf{0}} L(\boldsymbol{x}, \alpha, \beta) = \infty$$

  - Infeasible points will never be optimal (if there are feasible points)

Alternate Iterative Algorithm

$$\min_{\boldsymbol{x}} \max_{\alpha,\beta,\alpha \geq \mathbf{0}} f(\boldsymbol{x}) + \sum_i \alpha_i g^{(i)}(\boldsymbol{x}) + \sum_j \beta_j h^{(j)}(\boldsymbol{x})$$

Alternate Iterative Algorithm

$$\min_{\boldsymbol{x}} \max_{\alpha,\beta,\alpha\geq\mathbf{0}} f(\boldsymbol{x}) + \sum_i \alpha_i g^{(i)}(\boldsymbol{x}) + \sum_j \beta_j h^{(j)}(\boldsymbol{x})$$

- "Large" α and β create a "barrier" for feasible solutions

Input: $\boldsymbol{x}^{(0)}$ an initial guess, $\alpha^{(0)} = \mathbf{0}$, $\beta^{(0)} = \mathbf{0}$
repeat
  Solve $\boldsymbol{x}^{(t+1)} = \arg\min_{\boldsymbol{x}} L(\boldsymbol{x};\alpha^{(t)},\beta^{(t)})$ using some iterative algorithm starting at $\boldsymbol{x}^{(t)}$;
  if $\boldsymbol{x}^{(t+1)} \notin \mathbb{C}$ **then**
    Increase $\alpha^{(t)}$ to get $\alpha^{(t+1)}$;
    Get $\beta^{(t+1)}$ by increasing the magnitude of $\beta^{(t)}$ and set $\mathrm{sign}(\beta_j^{(t+1)}) = \mathrm{sign}(h^{(j)}(\boldsymbol{x}^{(t+1)}))$;
  end
until $\boldsymbol{x}^{(t+1)} \in \mathbb{C}$;

KKT Conditions

Theorem (KKT Conditions)

If $\boldsymbol{x}^*$ is an optimal point, then there exists KKT multipliers α^* and β^* such that the ***Karush-Kuhn-Tucker (KKT) conditions*** *are satisfied:*

KKT Conditions

Theorem (KKT Conditions)

If $\boldsymbol{x}^*$ is an optimal point, then there exists KKT multipliers α^* and β^* such that the ***Karush-Kuhn-Tucker (KKT) conditions*** *are satisfied:*
Lagrangian stationarity: $\nabla L(\boldsymbol{x}^*, \alpha^*, \beta^*) = 0$

KKT Conditions

Theorem (KKT Conditions)

If $\boldsymbol{x}^*$ is an optimal point, then there exists KKT multipliers α^* and β^* such that the ***Karush-Kuhn-Tucker (KKT) conditions*** *are satisfied:*
Lagrangian stationarity: $\nabla L(\boldsymbol{x}^*, \alpha^*, \beta^*) = 0$
Primal feasibility: $g^{(i)}(\boldsymbol{x}^*) \leq 0$ *and* $h^{(j)}(\boldsymbol{x}^*) = 0$ *for all* i *and* j
Dual feasibility: $\alpha^* \geq \mathbf{0}$

KKT Conditions

Theorem (KKT Conditions)

*If $\boldsymbol{x}^*$ is an optimal point, then there exists KKT multipliers α^* and β^* such that the **Karush-Kuhn-Tucker (KKT) conditions** are satisfied:*
Lagrangian stationarity: $\nabla L(\boldsymbol{x}^*, \alpha^*, \beta^*) = 0$
Primal feasibility: $g^{(i)}(\boldsymbol{x}^*) \leq 0$ *and* $h^{(j)}(\boldsymbol{x}^*) = 0$ *for all* i *and* j
Dual feasibility: $\alpha^* \geq \boldsymbol{0}$
***Complementary slackness**:* $\alpha_i^* g^{(i)}(\boldsymbol{x}^*) = 0$ *for all* i*.*

KKT Conditions

Theorem (KKT Conditions)

If $\boldsymbol{x}^*$ is an optimal point, then there exists KKT multipliers α^* and β^* such that the ***Karush-Kuhn-Tucker (KKT) conditions*** *are satisfied:*
Lagrangian stationarity: $\nabla L(\boldsymbol{x}^*, \alpha^*, \beta^*) = 0$
Primal feasibility: $g^{(i)}(\boldsymbol{x}^*) \leq 0$ *and* $h^{(j)}(\boldsymbol{x}^*) = 0$ *for all* i *and* j
Dual feasibility: $\alpha^* \geq \mathbf{0}$
Complementary slackness*:* $\alpha_i^* g^{(i)}(\boldsymbol{x}^*) = 0$ *for all* i*.*

- Only a necessary condition for $\boldsymbol{x}^*$ being optimal

KKT Conditions

Theorem (KKT Conditions)

If $\boldsymbol{x}^*$ is an optimal point, then there exists KKT multipliers α^* and β^* such that the ***Karush-Kuhn-Tucker (KKT) conditions*** *are satisfied:*
Lagrangian stationarity: $\nabla L(\boldsymbol{x}^*, \alpha^*, \beta^*) = 0$
Primal feasibility: $g^{(i)}(\boldsymbol{x}^*) \leq 0$ *and* $h^{(j)}(\boldsymbol{x}^*) = 0$ *for all* i *and* j
Dual feasibility: $\alpha^* \geq \mathbf{0}$
Complementary slackness*:* $\alpha_i^* g^{(i)}(\boldsymbol{x}^*) = 0$ *for all* i*.*

- Only a necessary condition for $\boldsymbol{x}^*$ being optimal
- Sufficient if the original problem is ***convex***

Complementary Slackness

- Why $\alpha_i^* g^{(i)}(\boldsymbol{x}^*) = 0$?
- For $\boldsymbol{x}^*$ being feasible, we have $g^{(i)}(\boldsymbol{x}^*) \leq 0$

Complementary Slackness

- Why $\alpha_i^* g^{(i)}(\boldsymbol{x}^*) = 0$?
- For $\boldsymbol{x}^*$ being feasible, we have $g^{(i)}(\boldsymbol{x}^*) \leq 0$
- If $g^{(i)}$ is ***active*** (i.e., $g^{(i)}(\boldsymbol{x}^*) = 0$) then $\alpha_i^* g^{(i)}(\boldsymbol{x}^*) = 0$

Complementary Slackness

- Why $\alpha_i^* g^{(i)}(\boldsymbol{x}^*) = 0$?
- For $\boldsymbol{x}^*$ being feasible, we have $g^{(i)}(\boldsymbol{x}^*) \leq 0$
- If $g^{(i)}$ is ***active*** (i.e., $g^{(i)}(\boldsymbol{x}^*) = 0$) then $\alpha_i^* g^{(i)}(\boldsymbol{x}^*) = 0$
- If $g^{(i)}$ is ***inactive*** (i.e., $g^{(i)}(\boldsymbol{x}^*) < 0$), then
  - To maximize the $\alpha_i g^{(i)}(\boldsymbol{x}^*)$ term in the Lagrangian in terms of α_i subject to $\alpha_i \geq 0$, we have $\alpha_i^* = 0$

Complementary Slackness

- Why $\alpha_i^* g^{(i)}(\boldsymbol{x}^*) = 0$?
- For $\boldsymbol{x}^*$ being feasible, we have $g^{(i)}(\boldsymbol{x}^*) \leq 0$
- If $g^{(i)}$ is ***active*** (i.e., $g^{(i)}(\boldsymbol{x}^*) = 0$) then $\alpha_i^* g^{(i)}(\boldsymbol{x}^*) = 0$
- If $g^{(i)}$ is ***inactive*** (i.e., $g^{(i)}(\boldsymbol{x}^*) < 0$), then
  - To maximize the $\alpha_i g^{(i)}(\boldsymbol{x}^*)$ term in the Lagrangian in terms of α_i subject to $\alpha_i \geq 0$, we have $\alpha_i^* = 0$
  - Again $\alpha_i^* g^{(i)}(\boldsymbol{x}^*) = 0$

Complementary Slackness

- Why $\alpha_i^* g^{(i)}(\boldsymbol{x}^*) = 0$?
- For $\boldsymbol{x}^*$ being feasible, we have $g^{(i)}(\boldsymbol{x}^*) \leq 0$
- If $g^{(i)}$ is ***active*** (i.e., $g^{(i)}(\boldsymbol{x}^*) = 0$) then $\alpha_i^* g^{(i)}(\boldsymbol{x}^*) = 0$
- If $g^{(i)}$ is ***inactive*** (i.e., $g^{(i)}(\boldsymbol{x}^*) < 0$), then
    - To maximize the $\alpha_i g^{(i)}(\boldsymbol{x}^*)$ term in the Lagrangian in terms of α_i subject to $\alpha_i \geq 0$, we have $\alpha_i^* = 0$
    - Again $\alpha_i^* g^{(i)}(\boldsymbol{x}^*) = 0$
- So what?

Complementary Slackness

- Why $\alpha_i^* g^{(i)}(\boldsymbol{x}^*) = 0$?
- For $\boldsymbol{x}^*$ being feasible, we have $g^{(i)}(\boldsymbol{x}^*) \leq 0$
- If $g^{(i)}$ is ***active*** (i.e., $g^{(i)}(\boldsymbol{x}^*) = 0$) then $\alpha_i^* g^{(i)}(\boldsymbol{x}^*) = 0$
- If $g^{(i)}$ is ***inactive*** (i.e., $g^{(i)}(\boldsymbol{x}^*) < 0$), then
  - To maximize the $\alpha_i g^{(i)}(\boldsymbol{x}^*)$ term in the Lagrangian in terms of α_i subject to $\alpha_i \geq 0$, we have $\alpha_i^* = 0$
  - Again $\alpha_i^* g^{(i)}(\boldsymbol{x}^*) = 0$
- So what?
  - $\alpha_i^* > 0$ implies $g^{(i)}(\boldsymbol{x}^*) = 0$
  - Once $\boldsymbol{x}^*$ is solved, we can quickly find out the active inequality constrains by checking $\alpha_i^* > 0$

Outline

Outline

The Regression Problem

- Given a training dataset: $\mathbb{X} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$
  - $\boldsymbol{x}^{(i)} \in \mathbb{R}^D$, called explanatory variables (attributes/features)
  - $y^{(i)} \in \mathbb{R}$, called response/target variables (labels)
- Goal: to find a function $f(\boldsymbol{x}) = \hat{y}$ such that $\hat{y}$ is close to the true label y

The Regression Problem

- Given a training dataset: $\mathbb{X} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$
  - $\boldsymbol{x}^{(i)} \in \mathbb{R}^D$, called explanatory variables (attributes/features)
  - $y^{(i)} \in \mathbb{R}$, called response/target variables (labels)
- Goal: to find a function $f(\boldsymbol{x}) = \hat{y}$ such that $\hat{y}$ is close to the true label y
- Example: to predict the price of a stock tomorrow

The Regression Problem

- Given a training dataset: $\mathbb{X} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$
  - $\boldsymbol{x}^{(i)} \in \mathbb{R}^D$, called explanatory variables (attributes/features)
  - $y^{(i)} \in \mathbb{R}$, called response/target variables (labels)
- Goal: to find a function $f(\boldsymbol{x}) = \hat{y}$ such that $\hat{y}$ is close to the true label y
- Example: to predict the price of a stock tomorrow
- Could you define a model $\mathbb{F} = \{f\}$ and cost function $C[f]$?

The Regression Problem

- Given a training dataset: $\mathbb{X} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$
  - $\boldsymbol{x}^{(i)} \in \mathbb{R}^D$, called explanatory variables (attributes/features)
  - $y^{(i)} \in \mathbb{R}$, called response/target variables (labels)
- Goal: to find a function $f(\boldsymbol{x}) = \hat{y}$ such that $\hat{y}$ is close to the true label y
- Example: to predict the price of a stock tomorrow
- Could you define a model $\mathbb{F} = \{f\}$ and cost function $C[f]$?
- How about "relaxing" the Adaline by removing the sign function when making the final prediction?
  - Adaline: $\hat{y} = \mathrm{sign}(\boldsymbol{w}^\top \boldsymbol{x} - b)$
  - Regressor: $\hat{y} = \boldsymbol{w}^\top \boldsymbol{x} - b$

Linear Regression I

- Model: $\mathbb{F} = \{f : f(\boldsymbol{x}; \boldsymbol{w}, b) = \boldsymbol{w}^\top \boldsymbol{x} - b\}$
  - Shorthand: $f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{x}$, where $\boldsymbol{w} = [-b, w_1, \cdots, w_D]^\top$ and $\boldsymbol{x} = [1, x_1, \cdots, x_D]^\top$

Linear Regression I

- Model: $\mathbb{F} = \{f : f(\boldsymbol{x};\boldsymbol{w},b) = \boldsymbol{w}^\top \boldsymbol{x} - b\}$
    - Shorthand: $f(\boldsymbol{x};\boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{x}$, where $\boldsymbol{w} = [-b, w_1, \cdots, w_D]^\top$ and $\boldsymbol{x} = [1, x_1, \cdots, x_D]^\top$
- Cost function and optimization problem:

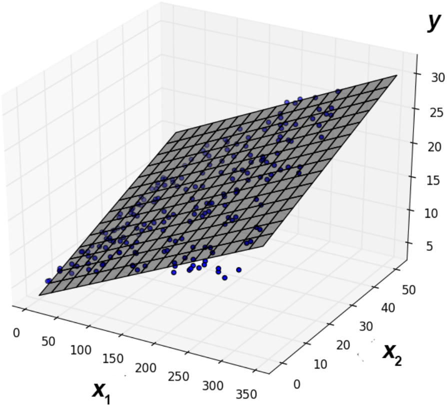
$$\arg\min_{\boldsymbol{w}} \frac{1}{2} \sum_{i=1}^{N} \|y^{(i)} - \boldsymbol{w}^\top \boldsymbol{x}^{(i)}\|^2 = \arg\min_{\boldsymbol{w}} \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2$$

    - $\boldsymbol{X} = \begin{bmatrix} 1 & \boldsymbol{x}^{(1)\top} \\ \vdots & \vdots \\ 1 & \boldsymbol{x}^{(N)\top} \end{bmatrix} \in \mathbb{R}^{N \times (D+1)}$ the design matrix
    - $\boldsymbol{y} = [y^{(1)}, \cdots, y^{(N)}]^\top$ the label vector

Linear Regression II

$$\arg\min_{\boldsymbol{w}} \frac{1}{2}\sum_{i=1}^{N} \|y^{(i)} - \boldsymbol{w}^\top \boldsymbol{x}^{(i)}\|^2 = \arg\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2$$

- Basically, we fit a hyperplane to training data
  - Each $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} - b \in \mathbb{F}$ is a hyperplane in the graph



Training Using Gradient Descent

$$\arg\min_{\boldsymbol{w}} \frac{1}{2}\sum_{i=1}^{N} \|y^{(i)} - \boldsymbol{w}^\top \boldsymbol{x}^{(i)}\|^2 = \arg\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2$$

- Batch:

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \eta \sum_{i=1}^{N} (y^{(i)} - \boldsymbol{w}^{(t)\top}\boldsymbol{x}^{(i)})\boldsymbol{x}^{(i)} = \boldsymbol{w}^{(t)} + \eta \boldsymbol{X}^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})$$

- Stochastic (with minibatch size $|M| = 1$):

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \eta (y^{(t)} - \boldsymbol{w}^{(t)\top}\boldsymbol{x}^{(t)})\boldsymbol{x}^{(t)}$$

Evaluation Metrics of Regression Models

- Given a training/testing set $\mathbb{X} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$
- How to evaluate the predictions $\hat{y}^{(i)}$ made by a function f?

Evaluation Metrics of Regression Models

- Given a training/testing set $\mathbb{X} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$
- How to evaluate the predictions $\hat{y}^{(i)}$ made by a function f?
- Sum of Square Errors (SSE): $\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2$

Evaluation Metrics of Regression Models

- Given a training/testing set $\mathbb{X} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$
- How to evaluate the predictions $\hat{y}^{(i)}$ made by a function f?
- Sum of Square Errors (SSE): $\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2$
- Mean Square Error (MSE): $\frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2$

Evaluation Metrics of Regression Models

- Given a training/testing set $\mathbb{X} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$
- How to evaluate the predictions $\hat{y}^{(i)}$ made by a function f?
- Sum of Square Errors (SSE): $\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2$
- Mean Square Error (MSE): $\frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2$
- Relative Square Error (RSE):

$$\frac{\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2}{\sum_{i=1}^{N}(y^{(i)} - \bar{y}^{(i)})^2},$$

  where $\bar{y} = \frac{1}{N}\sum_i y^{(i)}$
  - What does it mean?

Evaluation Metrics of Regression Models

- Given a training/testing set $\mathbb{X} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^N$
- How to evaluate the predictions $\hat{y}^{(i)}$ made by a function f?
- Sum of Square Errors (SSE): $\sum_{i=1}^N (y^{(i)} - \hat{y}^{(i)})^2$
- Mean Square Error (MSE): $\frac{1}{N}\sum_{i=1}^N (y^{(i)} - \hat{y}^{(i)})^2$
- Relative Square Error (RSE):

$$\frac{\sum_{i=1}^N (y^{(i)} - \hat{y}^{(i)})^2}{\sum_{i=1}^N (y^{(i)} - \bar{y}^{(i)})^2},$$

  where $\bar{y} = \frac{1}{N}\sum_i y^{(i)}$
  - What does it mean? Compares f with a dummy prediction $\bar{y}$

Evaluation Metrics of Regression Models

- Given a training/testing set $\mathbb{X} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$
- How to evaluate the predictions $\hat{y}^{(i)}$ made by a function f?
- Sum of Square Errors (SSE): $\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2$
- Mean Square Error (MSE): $\frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2$
- Relative Square Error (RSE):

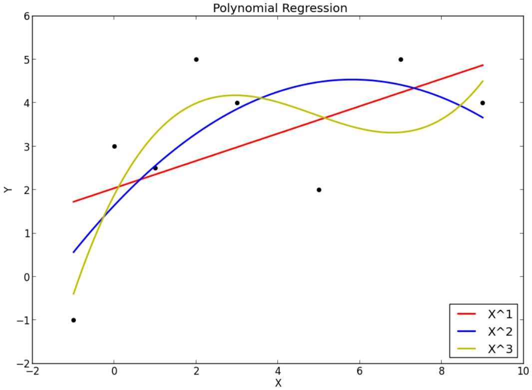
$$\frac{\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2}{\sum_{i=1}^{N}(y^{(i)} - \bar{y}^{(i)})^2},$$

  where $\bar{y} = \frac{1}{N}\sum_i y^{(i)}$
  - What does it mean? Compares f with a dummy prediction $\bar{y}$
- Coefficient of Determination: $R^2 = 1 - RSE \in [0, 1]$
  - Higher the better

Outline

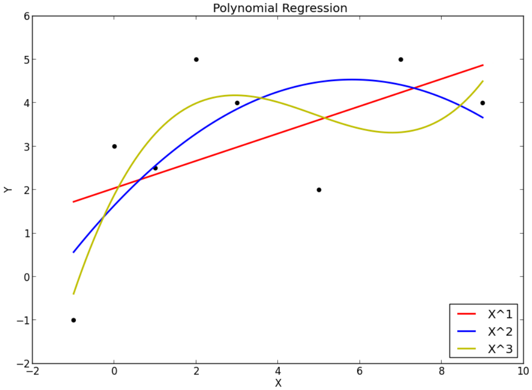
Polynomial Regression

- In practice, the relationship between explanatory variables and target variables may not be linear
- ***Polynomial regression*** fits a high-order polynomial to the training data



Polynomial Regression

- In practice, the relationship between explanatory variables and target variables may not be linear
- ***Polynomial regression*** fits a high-order polynomial to the training data
- How?



Data Augmentation

- Suppose $D = 2$, i.e., $\boldsymbol{x} = [x_1, x_2]^\top$
- Linear model:

$$\mathbb{F} = \{f : f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{x} + w_0 = w_0 + w_1 x_1 + w_2 x_2\}$$

Data Augmentation

- Suppose $D = 2$, i.e., $\boldsymbol{x} = [x_1, x_2]^\top$
- Linear model:

$$\mathbb{F} = \{f : f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{x} + w_0 = w_0 + w_1 x_1 + w_2 x_2\}$$

- Quadratic model:

$$\mathbb{F} = \{f : f(\boldsymbol{x}; \boldsymbol{w}) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_1 x_2 + w_5 x_2^2\}$$

Data Augmentation

- Suppose $D = 2$, i.e., $\boldsymbol{x} = [x_1, x_2]^\top$
- Linear model:

$$\mathbb{F} = \{f : f(\boldsymbol{x};\boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{x} + w_0 = w_0 + w_1 x_1 + w_2 x_2\}$$

- Quadratic model:

$$\mathbb{F} = \{f : f(\boldsymbol{x};\boldsymbol{w}) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_1 x_2 + w_5 x_2^2\}$$

- We can simply ***augment*** the data dimension to reduce a quadratic model to a linear one
  - A general technique in ML to "transform" a linear model into a nonlinear one

Data Augmentation

- Suppose $D = 2$, i.e., $\boldsymbol{x} = [x_1, x_2]^\top$
- Linear model:

$$\mathbb{F} = \{f : f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{x} + w_0 = w_0 + w_1 x_1 + w_2 x_2\}$$

- Quadratic model:

$$\mathbb{F} = \{f : f(\boldsymbol{x}; \boldsymbol{w}) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_1 x_2 + w_5 x_2^2\}$$

- We can simply ***augment*** the data dimension to reduce a quadratic model to a linear one
  - A general technique in ML to "transform" a linear model into a nonlinear one
- How many variables to solve in $\boldsymbol{w}$ for a polynomial regression problem of degree P? [Homework]

Outline

Regularization

- There's another major difference between the ML algorithms and optimization techniques:

Regularization

- There's another major difference between the ML algorithms and optimization techniques:
- We usually care about the ***testing performance*** rather than the training performance
  - E.g., in classification, we report the testing accuracy

Regularization

- There's another major difference between the ML algorithms and optimization techniques:
- We usually care about the ***testing performance*** rather than the training performance
  - E.g., in classification, we report the testing accuracy
- Goal: to learn a function that ***generalizes*** to unseen data well

Regularization

- There's another major difference between the ML algorithms and optimization techniques:
- We usually care about the ***testing performance*** rather than the training performance
  - E.g., in classification, we report the testing accuracy
- Goal: to learn a function that ***generalizes*** to unseen data well
- ***Regularization***: techniques that improve the generalizability of the learned function

Regularization

- There's another major difference between the ML algorithms and optimization techniques:
- We usually care about the ***testing performance*** rather than the training performance
  - E.g., in classification, we report the testing accuracy
- Goal: to learn a function that ***generalizes*** to unseen data well
- ***Regularization***: techniques that improve the generalizability of the learned function
- How to regularize the linear regression?

$$\arg\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2$$

Regularized Linear Regression

- One way to improve the generalizability of f is to make it "flat:"

$$\begin{array}{c}\arg\min_{\boldsymbol{w}\in\mathbb{R}^D, b} \frac{1}{2}\|\boldsymbol{y}-(\boldsymbol{X}\boldsymbol{w}-b)\|^2 \\ \text{subject to } \|\boldsymbol{w}\|^2 \leq T\end{array} = \begin{array}{c}\arg\min_{\boldsymbol{w}\in\mathbb{R}^{D+1}} \frac{1}{2}\|\boldsymbol{y}-(\boldsymbol{X}\boldsymbol{w})\|^2 \\ \text{subject to } \boldsymbol{w}^\top \boldsymbol{S}\boldsymbol{w} \leq T\end{array}$$

  - $\boldsymbol{S} = \mathrm{diag}([0, 1, \cdots, 1]^\top) \in \mathbb{R}^{(D+1)\times(D+1)}$ (b is not regularized)

Regularized Linear Regression

- One way to improve the generalizability of f is to make it "flat:"

$$\begin{array}{c}\arg\min_{\boldsymbol{w}\in\mathbb{R}^D,b} \frac{1}{2}\|\boldsymbol{y}-(\boldsymbol{X}\boldsymbol{w}-b)\|^2 \\ \text{subject to } \|\boldsymbol{w}\|^2 \leq T\end{array} = \begin{array}{c}\arg\min_{\boldsymbol{w}\in\mathbb{R}^{D+1}} \frac{1}{2}\|\boldsymbol{y}-(\boldsymbol{X}\boldsymbol{w})\|^2 \\ \text{subject to } \boldsymbol{w}^\top\boldsymbol{S}\boldsymbol{w} \leq T\end{array}$$

  - $\boldsymbol{S} = \mathrm{diag}([0,1,\cdots,1]^\top) \in \mathbb{R}^{(D+1)\times(D+1)}$ (b is not regularized)
- We will explain why this works later

Regularized Linear Regression

- One way to improve the generalizability of f is to make it "flat:"

$$\begin{array}{c}\arg\min_{\boldsymbol{w}\in\mathbb{R}^D,b}\frac{1}{2}\|\boldsymbol{y}-(\boldsymbol{X}\boldsymbol{w}-b)\|^2\\ \text{subject to } \|\boldsymbol{w}\|^2\leq T\end{array} = \begin{array}{c}\arg\min_{\boldsymbol{w}\in\mathbb{R}^{D+1}}\frac{1}{2}\|\boldsymbol{y}-(\boldsymbol{X}\boldsymbol{w})\|^2\\ \text{subject to } \boldsymbol{w}^\top\boldsymbol{S}\boldsymbol{w}\leq T\end{array}$$

  - $\boldsymbol{S}=\text{diag}([0,1,\cdots,1]^\top)\in\mathbb{R}^{(D+1)\times(D+1)}$ (b is not regularized)
- We will explain why this works later
- How to solve this problem?

Regularized Linear Regression

- One way to improve the generalizability of f is to make it "flat:"

$$\begin{array}{c}\arg\min_{\boldsymbol{w}\in\mathbb{R}^D,b}\frac{1}{2}\|\boldsymbol{y}-(\boldsymbol{X}\boldsymbol{w}-b)\|^2\\ \text{subject to } \color{red}{\|\boldsymbol{w}\|^2\leq T}\end{array} = \begin{array}{c}\arg\min_{\boldsymbol{w}\in\mathbb{R}^{D+1}}\frac{1}{2}\|\boldsymbol{y}-(\boldsymbol{X}\boldsymbol{w})\|^2\\ \text{subject to } \color{red}{\boldsymbol{w}^\top\boldsymbol{S}\boldsymbol{w}\leq T}\end{array}$$

  - $\boldsymbol{S}=\mathrm{diag}([0,1,\cdots,1]^\top)\in\mathbb{R}^{(D+1)\times(D+1)}$ (b is not regularized)
- We will explain why this works later
- How to solve this problem?
- Using the KKT method, we have

$$\arg\min_{\boldsymbol{w}}\max_{\alpha,\alpha\geq 0} L(\boldsymbol{w},\alpha)=\arg\min_{\boldsymbol{w}}\max_{\alpha,\alpha\geq 0}\frac{1}{2}\left(\|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w}\|^2+\alpha(\boldsymbol{w}^\top\boldsymbol{S}\boldsymbol{w}-T)\right)$$

Alternate Iterative Algorithm

$$\arg\min_{\boldsymbol{w}} \max_{\alpha,\alpha\geq 0} L(\boldsymbol{w},\alpha) = \arg\min_{\boldsymbol{w}} \max_{\alpha,\alpha\geq 0} \frac{1}{2}\left(\|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w}\|^2 + \alpha(\boldsymbol{w}^\top \boldsymbol{S}\boldsymbol{w} - T)\right)$$

Input: $\boldsymbol{w}^{(0)} \in \mathbb{R}^d$ an initial guess, $\alpha^{(0)} = 0$, $\delta > 0$

repeat

Solve $\boldsymbol{w}^{(t+1)} = \arg\min_{\boldsymbol{w}} L(\boldsymbol{w};\alpha^{(t)})$ using some iterative algorithm starting at $\boldsymbol{w}^{(t)}$;

if $\boldsymbol{w}^{(t+1)\top}\boldsymbol{w}^{(t+1)} > T$ **then**

$\alpha^{(t+1)} = \alpha^{(t)} + \delta$ in order to increase $L(\alpha;\boldsymbol{w}^{(t+1)})$;

end

until $\boldsymbol{w}^{(t+1)\top}\boldsymbol{w}^{(t+1)} \leq T$;

Alternate Iterative Algorithm

$$\arg\min_{\boldsymbol{w}} \max_{\alpha,\alpha\geq 0} L(\boldsymbol{w},\alpha) = \arg\min_{\boldsymbol{w}} \max_{\alpha,\alpha\geq 0} \frac{1}{2}\left(\|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w}\|^2 + \alpha(\boldsymbol{w}^\top \boldsymbol{S}\boldsymbol{w} - T)\right)$$

Input: $\boldsymbol{w}^{(0)} \in \mathbb{R}^d$ an initial guess, $\alpha^{(0)} = 0$, $\delta > 0$

repeat

   Solve $\boldsymbol{w}^{(t+1)} = \arg\min_{\boldsymbol{w}} L(\boldsymbol{w};\alpha^{(t)})$ using some iterative algorithm starting at $\boldsymbol{w}^{(t)}$;

   if $\boldsymbol{w}^{(t+1)\top}\boldsymbol{w}^{(t+1)} > T$ **then**

      $\alpha^{(t+1)} = \alpha^{(t)} + \delta$ in order to increase $L(\alpha;\boldsymbol{w}^{(t+1)})$;

   end

until $\boldsymbol{w}^{(t+1)\top}\boldsymbol{w}^{(t+1)} \leq T$;

- We could also solve $\boldsymbol{w}^{(t+1)}$ analytically from $\frac{\partial}{\partial \boldsymbol{x}} L(\boldsymbol{w};\alpha^{(t)}) = \boldsymbol{0}$:

$$\boldsymbol{w}^{(t+1)} = \left(\boldsymbol{X}^\top \boldsymbol{X} + \alpha^{(t)}\boldsymbol{S}\right)^{-1}\boldsymbol{X}^\top \boldsymbol{y}$$

Outline

Dual Problem

- Given a problem (called ***primal problem***):

$$p^* = \min_{\boldsymbol{x}} \max_{\alpha,\beta,\alpha \geq \mathbf{0}} L(\boldsymbol{x}, \alpha, \beta)$$

Dual Problem

- Given a problem (called ***primal problem***):

$$p^* = \min_{\boldsymbol{x}} \max_{\alpha,\beta,\alpha\geq\boldsymbol{0}} L(\boldsymbol{x},\alpha,\beta)$$

- We define its ***dual problem*** as:

$$d^* = \max_{\alpha,\beta,\alpha\geq\boldsymbol{0}} \min_{\boldsymbol{x}} L(\boldsymbol{x},\alpha,\beta)$$

Dual Problem

- Given a problem (called ***primal problem***):

$$p^* = \min_{\boldsymbol{x}} \max_{\alpha,\beta,\alpha \geq \boldsymbol{0}} L(\boldsymbol{x}, \alpha, \beta)$$

- We define its ***dual problem*** as:

$$d^* = \max_{\alpha,\beta,\alpha \geq \boldsymbol{0}} \min_{\boldsymbol{x}} L(\boldsymbol{x}, \alpha, \beta)$$

- By the max-min inequality, we have $d^* \leq p^*$ [Homework]

Dual Problem

- Given a problem (called ***primal problem***):

$$p^* = \min_{\boldsymbol{x}} \max_{\alpha,\beta,\alpha \geq \mathbf{0}} L(\boldsymbol{x}, \alpha, \beta)$$

- We define its ***dual problem*** as:

$$d^* = \max_{\alpha,\beta,\alpha \geq \mathbf{0}} \min_{\boldsymbol{x}} L(\boldsymbol{x}, \alpha, \beta)$$

- By the max-min inequality, we have $d^* \leq p^*$ [Homework]
- $(p^* - d^*)$ is called the ***duality gap***
  - p^* and d^* are called the ***primal*** and ***dual values***, respectively

Strong Duality

- ***Strong duality*** holds if $d^* = p^*$
- When will it happen?

Strong Duality

- ***Strong duality*** holds if $d^* = p^*$
- When will it happen?
- If the primal problem has solution and ***convex***
- Why considering dual problem?

Example

- Consider a primal problem:

$$\begin{array}{c}\arg\min_{\boldsymbol{x}\in\mathbb{R}^d} \frac{1}{2}\|\boldsymbol{x}\|^2 \\ \text{subject to } \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{A} \in \mathbb{R}^{n\times d}\end{array} = \arg\min_{\boldsymbol{x}} \max_{\alpha,\alpha\geq\boldsymbol{0}} \frac{1}{2}\|\boldsymbol{x}\|^2 - \alpha^\top(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b})$$

  - Convex, so strong duality holds

Example

- Consider a primal problem:

$$\begin{array}{c}\arg\min_{\boldsymbol{x}\in\mathbb{R}^d} \frac{1}{2}\|\boldsymbol{x}\|^2 \\ \text{subject to } \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{A} \in \mathbb{R}^{n\times d}\end{array} = \arg\min_{\boldsymbol{x}} \max_{\alpha,\alpha\geq\boldsymbol{0}} \frac{1}{2}\|\boldsymbol{x}\|^2 - \alpha^\top(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b})$$

  - Convex, so strong duality holds

- We can get the same solution via the dual problem:

$$\arg\max_{\alpha,\alpha\geq\boldsymbol{0}} \min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{x}\|^2 - \alpha^\top(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b})$$

- Solving $\min_{\boldsymbol{x}} L(\boldsymbol{x},\alpha)$ analytically, we have $\boldsymbol{x}^* = \boldsymbol{A}^\top\alpha$

Example

- Consider a primal problem:

$$\begin{array}{c}\arg\min_{\boldsymbol{x}\in\mathbb{R}^d} \frac{1}{2}\|\boldsymbol{x}\|^2 \\ \text{subject to } \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{A} \in \mathbb{R}^{n\times d}\end{array} = \arg\min_{\boldsymbol{x}} \max_{\alpha,\alpha\geq\boldsymbol{0}} \frac{1}{2}\|\boldsymbol{x}\|^2 - \alpha^\top(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b})$$

  - Convex, so strong duality holds

- We can get the same solution via the dual problem:

$$\arg\max_{\alpha,\alpha\geq\boldsymbol{0}} \min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{x}\|^2 - \alpha^\top(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b})$$

- Solving $\min_{\boldsymbol{x}} L(\boldsymbol{x},\alpha)$ analytically, we have $\boldsymbol{x}^* = \boldsymbol{A}^\top\alpha$
- Substituting this into the dual, we get

$$\arg\max_{\alpha,\alpha\geq\boldsymbol{0}} -\frac{1}{2}\|\boldsymbol{A}^\top\alpha\|^2 + \boldsymbol{b}^\top\alpha$$

- We now solve n variables instead of d (beneficial when $n \ll d$)