Basics of Numerical Optimization for Pattern Recognition and Machine Learning

Heung-Il Suk
hisuk@korea.ac.kr
http://www.ku-milab.org

Department of Brain and Cognitive Engineering,
Korea University

January 20, 2016

Mathematical Optimization

(from Wikipedia)

- alternatively, optimization or mathematical programming
- the selection of a **best element** from some set of available alternatives with regard to **some criteria**
  - *e.g.*, weight coefficients, connection weights, mean, covariance, *etc.*
  - *e.g.*, minimum loss, minimum risk, maximum (log) likelihood

General Scheme in PRML

Given *i.i.d.* samples $X = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$,
the aim is to build a good and useful approximation to y_n

$$\mathbf{x} \quad \Rightarrow \quad f(\mathbf{x}|\theta) \quad \Rightarrow \quad y$$

1. Model: $f(\mathbf{x}_n|\theta) \rightarrow$ enough capacity
2. Loss function: $J(\theta|X) = \sum_n L(y_n, f(\mathbf{x}_n|\theta)) \rightarrow$ sufficient training data
3. Learning: $\theta^* = \underset{\theta}{\operatorname{argmin}} J(\theta|X) \rightarrow$ good optimization method

PRML from an optimization perspective

- formulating a model
- deriving the core optimization problem
- using mathematic programming to solve it

Notations

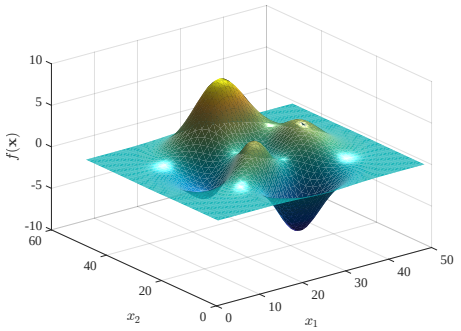
- $\mathbb{R}$: real number
- Scalar: $x \in \mathbb{R}$
- (Column) vector: $\mathbf{x} = [x_i] \in \mathbb{R}^d$
- Matrix: $\mathbf{X} = [X_{ij}] \in \mathbb{R}^{d \times m}$
- Transpose: $\mathbf{X}^T$
- Inverse: $\mathbf{X}^{-1}$

Unconstrained Optimization

Unconstrained Optimization

$$\min_{\mathbf{x}} f(\mathbf{x}) \qquad \max_{\mathbf{x}} f(\mathbf{x})$$

- $f(\mathbf{x})$: objective function
  - cost/loss/error function (minimization)
  - (log) likelihood (maximization)
- "minimization" or "maximization" to make sense: $f : \mathbb{R}^d \to \mathbb{R}$

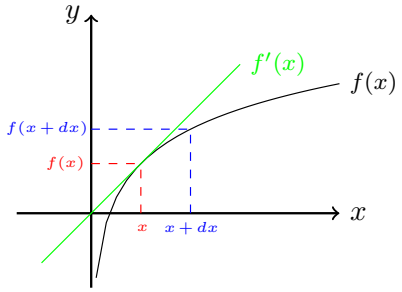


Fundamental meaning of a derivative

Assume that a function $f(x)$ is differentiable (*i.e.*, continuous) at x

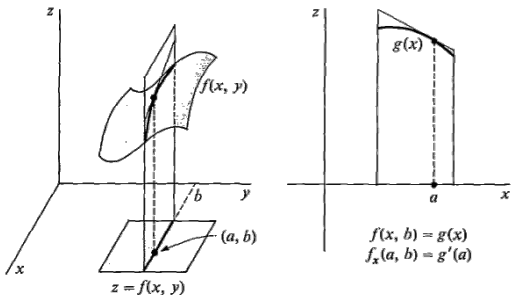
$$f'(x) = \frac{df(x)}{dx} = \lim_{dx \to 0} \frac{f(x+dx) - f(x)}{dx}$$

$$f(x+dx) \approx f(x) + dx f'(x)$$



- Partial derivative

$$\frac{\partial f(x,y)}{\partial x} \quad \frac{\partial f(x,y)}{\partial y}$$

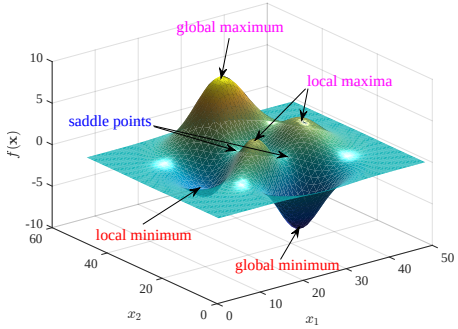


- Gradient vector

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial x_1}, \cdots, \frac{\partial f(\mathbf{x})}{\partial x_i}, \cdots, \frac{\partial f(\mathbf{x})}{\partial x_d}\right]^T$$

$$f(\mathbf{x}+\Delta\mathbf{x}) \approx f(\mathbf{x}) + \Delta\mathbf{x}^T \nabla f_{\mathbf{x}}(\mathbf{x})$$

- $\nabla f_{\mathbf{x}}(\mathbf{x}) = \mathbf{0}$: critical/stationary point
- local/global minimum/maximum, saddle points



Many local minima/maxima, saddle points surrounded by very flat regions make optimization difficult, especially in high-dimension.

Overview: Gradient-based Methods

$$\min_{\mathbf{x}} f(\mathbf{x})$$

1. Starting point $\mathbf{x}_0$
2. **Select a search direction $\mathbf{s}_i$ (★)**
3. Determine the step size η_i for movement along the search direction $\mathbf{s}_i$
4. Set a new point $\mathbf{x}_i = \mathbf{x}_{i-1} + \eta_i \mathbf{s}_i$
5. Check convergence: if not converged, go to step 2

Directional Derivative in Direction $\mathbf{s}$

$$f(\mathbf{x}+\eta\mathbf{s}) = \nabla_{\mathbf{x}} f(\mathbf{x})^T \mathbf{s}$$

$$\left(\because \frac{df}{d\eta} = \sum \left(\frac{\partial f}{\partial x_i}\right)\left(\frac{dx_i}{d\eta}\right) = \nabla_{\mathbf{x}} f(\mathbf{x})^T \mathbf{s}\right)$$

- Change in the function for a small step in the direction $\mathbf{s}$

$$\Delta f(\mathbf{x}) \approx \frac{df}{d\eta}\Delta\lambda$$

  - Representing the expected change in the function for a small step in the direction $\mathbf{s}$
- When locating the minimum exactly

$$\left.\frac{df}{d\eta}\right|_{\eta=\eta^*} = \nabla_{\mathbf{x}} f(\mathbf{x})^T \mathbf{s} = 0$$

To minimize f, we would like to find the direction in which f decreases the fastest

$$\min_{\mathbf{s},\mathbf{s}^T\mathbf{s}=1} \mathbf{s}^T \nabla_{\mathbf{x}} f(\mathbf{x}) = \min_{\mathbf{s},\mathbf{s}^T\mathbf{s}=1} \|\mathbf{s}\|_2 \cdot \|\nabla_{\mathbf{x}} f(\mathbf{x})\|_2 \cos\theta$$

$$\Rightarrow \min_{\mathbf{s}} \cos\theta$$

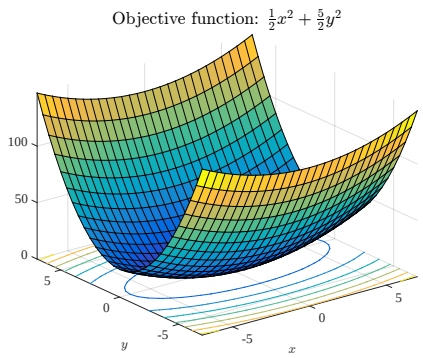
- minimum when s points in the opposite direction as the gradient
- *a.k.a.*, steepest descent or gradient descent

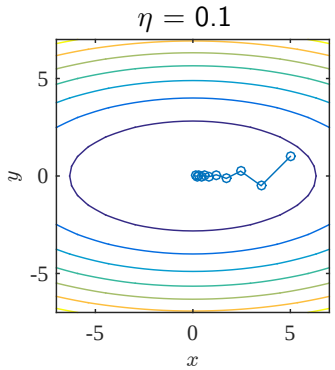
Steepest Gradient Descent

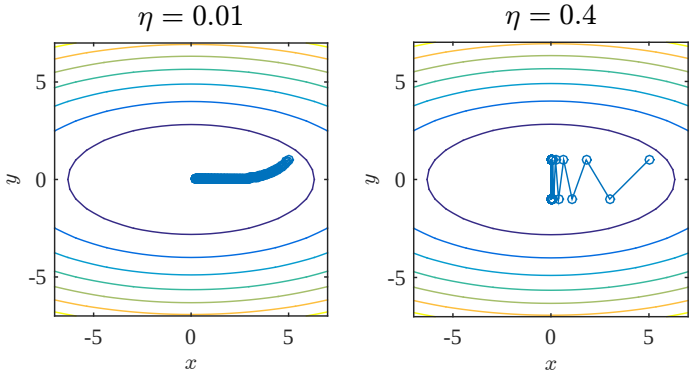
- One of the simplest unconstrained optimization methods
- Given an initial starting point, moves downhill until it can go no further.

$$\mathbf{x}^{\text{new}} = \mathbf{x} - \eta \nabla_{\mathbf{x}} f(\mathbf{x})$$

  - $-\nabla_{\mathbf{x}} f(\mathbf{x})$: search direction
  - η: stepsize, learning rate
    - How to set η: line search method, fixed value (*e.g.*, 10^{-2})







- Small η: long convergence time
- Large η: oscillations and even divergence

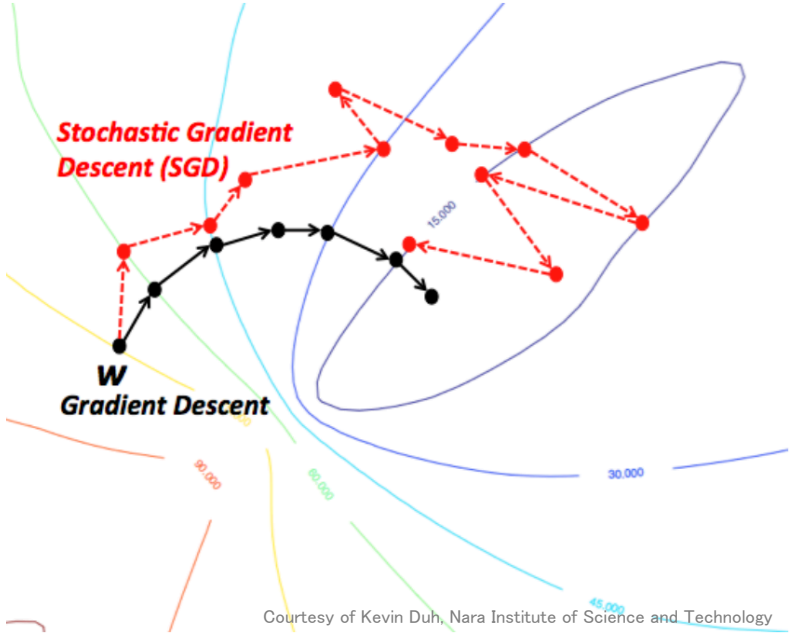
Batch vs. Stochastic (Online)

Batch learning

1. Initialize $\mathbf{w}$
2. Compute $\nabla_{\mathbf{w}} f(X) = \sum_n \nabla_{\mathbf{w}} f(\mathbf{x}_n)$
3. Update $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} f(X)$
4. Repeat steps 2-3 until convergence

Stochastic learning

1. Initialize $\mathbf{w}$
2. For each sample in $\{\mathbf{x}_n\}_{n=1}^N$
   - Compute $\nabla_{\mathbf{w}} f(\mathbf{x}_n)$
   - Update $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} f(\mathbf{x}_n)$
3. Repeat step 2 until convergence



Courtesy of Kevin Duh, Nara Institute of Science and Technology

- Batch learning goes in the direction of steepest descent
  - Slower to compute per iteration for a large dataset
- Stochastic can be considered as noisy descent
  - when stuck in a local optimum, a possibility of getting out of it
  - large η: update depends much on the recent instances (short memory)

Good tradeoff: *mini-batch stochastic gradient descent*
(a bunch of samples at a time)

Pros

- Always goes downhill, *i.e.*, reducing the function value
- Guaranteed to converge to a local optimum when enough steps are taken
- Simple to implement

Cons

- Very slow convergence on elongated functions

Hessian Matrix

$$\mathbf{H}(\mathbf{x}) = \nabla^2 f(\mathbf{x}) = \nabla\left(\nabla f(\mathbf{x})^T\right)$$

$$H_{ij} = \frac{\partial^2}{\partial x_i \partial x_j} f(\mathbf{x})$$

- Anywhere that second derivatives are continuous, the differential operators are commutative

$$H_{ij} = \frac{\partial^2}{\partial x_i \partial x_j} f(\mathbf{x}) = \frac{\partial^2}{\partial x_j \partial x_i} f(\mathbf{x}) = H_{ji} \quad (\mathbf{H}: \text{symmetric})$$

Gives us information about the curvature of a function
and tells us how the gradient is changing

Taylor Expansion

An approximation to a function at a point $\mathbf{x}_o$

$$f(\mathbf{x}) = f(\mathbf{x}_o) + \nabla f(\mathbf{x}_o)^T (\mathbf{x} - \mathbf{x}_o) + \frac{1}{2} (\mathbf{x} - \mathbf{x}_o)^T \nabla^2 f(\mathbf{x}_o) (\mathbf{x} - \mathbf{x}_o) + \cdots$$

By denoting $\mathbf{x} - \mathbf{x}_o$ as $\Delta\mathbf{x}$

$$f(\mathbf{x}) = f(\mathbf{x}_o + \Delta\mathbf{x}) = f(\mathbf{x}_o) + \nabla f(\mathbf{x}_o)^T \Delta\mathbf{x} + \frac{1}{2} \Delta\mathbf{x}^T \nabla^2 f(\mathbf{x}_o) (\Delta\mathbf{x}) + \cdots$$

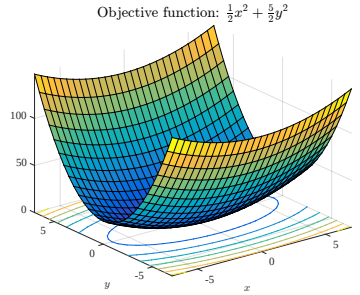
Newton's Method

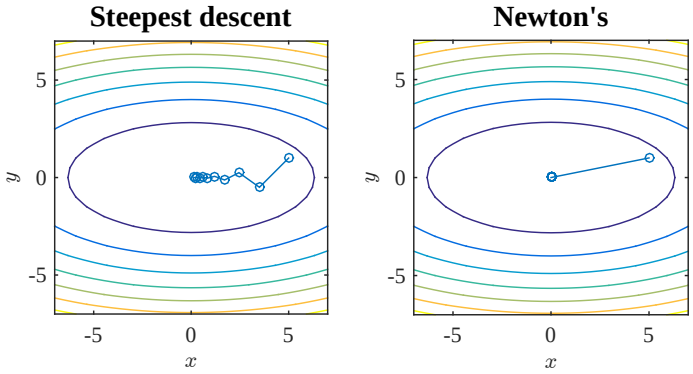
$$f(\mathbf{x}) = f(\mathbf{x}_o) + \nabla f(\mathbf{x}_o)^T \Delta\mathbf{x} + \frac{1}{2} \Delta\mathbf{x}^T \nabla^2 f(\mathbf{x}_o) (\Delta\mathbf{x})$$

- Approximating $f(\mathbf{x})$ near some point $\mathbf{x}_o$ by using second-order Taylor expansion
- If we solve for a critical point of the function by setting $\nabla f(\mathbf{x}) = 0$,

$$\nabla f(\mathbf{x}) = \nabla f(\mathbf{x}_o) + \mathbf{H}(\mathbf{x}_o) \Delta\mathbf{x}$$

$$\Delta\mathbf{x} = -\mathbf{H}(\mathbf{x}_o)^{-1} \nabla_{\mathbf{x}} f(\mathbf{x}_o)$$





Pros

- Solves quadratic in one step
- Very fast when close to optimum on non-quadratic

Cons

- Requiring second derivatives
  - Generally involving numerical computations
  - Computationally expensive
- Finding a critical point, where the gradient is equal to zero
  - No differentiation of minimum, maximum, or saddle point
  - Even can *diverge*

Quasi-Newton Methods

To avoiding the computational problems of requiring second derivatives and its inverse, approximations to $\mathbf{H}^{-1} \equiv \mathbf{G}$ are applied to each iteration.

1. Choose a starting point $\mathbf{x}_0$ and set $\mathbf{G}_0 = \mathbf{I}$
2. for $i = 1, 2, \ldots$ until convergence
   1. Compute a quasi-Newton direction
   $$\mathbf{s}_i = -\mathbf{G}_{i-1} \nabla f(\mathbf{x}_{i-1})$$
   2. Determine the optimal stepsize (*e.g.*, backtracking line search)
   $$f(\mathbf{x}_{i-1} + \eta_i \mathbf{s}_i) = \min_{\eta} f(\mathbf{x}_{i-1} + \eta \mathbf{s}_i)$$
   3. Set a new point $\mathbf{x}_i = \mathbf{x}_{i-1} + \eta_i \mathbf{s}_i$
   4. Update the approximation $\mathbf{G}_i$

Different methods use different rules for updating $\mathbf{G}_i$

- Davidon-Fletcher-Powell (DFP) update

$$\mathbf{G}_i = \mathbf{G}_{i-1} + \frac{\mathbf{v}_i \mathbf{v}_i^T}{\mathbf{v}_i^T \mathbf{y}_i} - \frac{(\mathbf{G}_{i-1}\mathbf{y}_i)(\mathbf{G}_{i-1}\mathbf{y}_i)^T}{\mathbf{y}_i^T \mathbf{G}_{i-1} \mathbf{y}_i}$$

- Broyden-Fletcher-Goldfarb-Shanno (BFGS) update

$$\mathbf{G}_i = \mathbf{G}_{i-1} + \left[1 + \frac{\mathbf{y}_i^T \mathbf{G}_{i-1} \mathbf{y}_i}{\mathbf{v}_i^T \mathbf{y}_i}\right] \left[\frac{\mathbf{v}_i \mathbf{v}_i^T}{\mathbf{v}_i^T \mathbf{y}_i}\right] - \left[\frac{\mathbf{v}_i \mathbf{y}_i^T \mathbf{G}_{i-1} + \mathbf{G}_{i-1} \mathbf{y}_i \mathbf{v}_i^T}{\mathbf{v}_i^T \mathbf{y}_i}\right]$$

where $\mathbf{v}_i = \eta_i \mathbf{s}_i,\ \mathbf{y}_i = \nabla f(\mathbf{x}_i) - \nabla f(\mathbf{x}_{i-1})$

Practice in PRML

Least Square Regression

Given *i.i.d.* samples $X = \{\mathbf{x}_n \in \mathbb{R}^d, y_n \in \mathbb{R}\}_{n=1}^N$,
the aim is to build a good and useful approximation to y_n

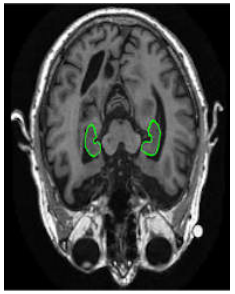
1. Model: $g(\mathbf{x}_n|\theta)$
2. Loss function:
   $J(\theta|X) = \sum_n L(y_n, g(\mathbf{x}_n|\theta))$
3. Optimization:
   $\theta^* = \underset{\theta}{\operatorname{argmin}} J(\theta|X)$

1. Linear regression: $y = \mathbf{x}^T\mathbf{w}$
2. Least square error:
   $J(\mathbf{w}) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}^T\mathbf{w}\|_2^2$
3. Optimization: $\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} J(\mathbf{w})$

Toy example: MRI-based clinical score estimation

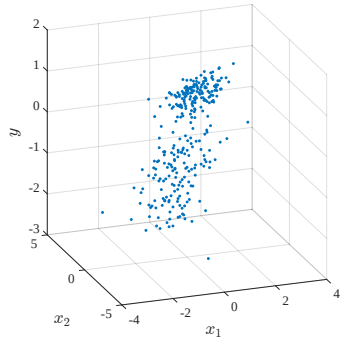
Training data: $X = \{\mathbf{x}_n \in \mathbb{R}^2, y_n \in \mathbb{R}\}_{n=1}^{300}$

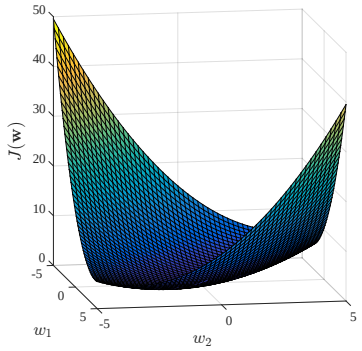
- $\mathbf{x}_n$: volume of left/right hippocampal formation in MRI
- y_n: clinical score of Mini-Mental State Examination (MMSE)



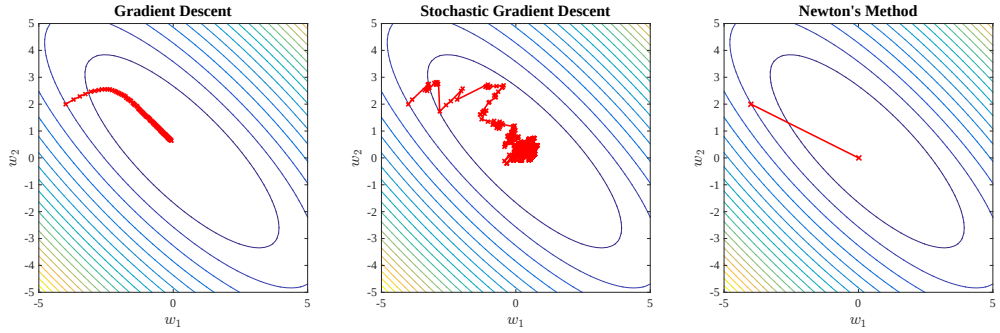
MMSE

- 30 question screening test for signs of dementia
- Assessing cognitive function; not for making a diagnosis
- Orientation to time and place, registration (immediate memory), attention and calculation, short-term memory, language skills, and complex commands



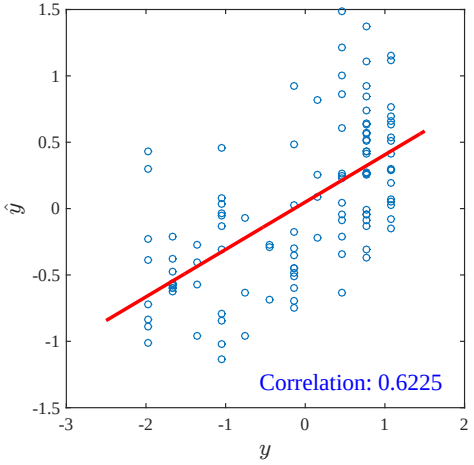


$$\nabla_{\mathbf{w}} J(\mathbf{w}) = -\mathbf{X}(\mathbf{y} - \mathbf{X}^T\mathbf{w})$$
$$\nabla_{\mathbf{w}}^2 J(\mathbf{w}) = \mathbf{X}\mathbf{X}^T$$



$$\hat{y} = \mathbf{x}^T\mathbf{w}^*$$

Test data: $\bar{X} = \{\mathbf{x}_n \in \mathbb{R}^2, y_n \in \mathbb{R}\}_{n=1}^{112}$



Logistic Regression

Given *i.i.d.* samples $X = \left\{\mathbf{x}_n \in \mathbb{R}^d, y_n \in \{0,1\}\right\}_{n=1}^{N}$,
the aim is to build a good and useful approximation to y_n

1. Logistic regression

$$y = \frac{1}{1+\exp\left(-\mathbf{x}^T\mathbf{w}\right)} = h(\mathbf{x})$$

2. Cross-entropy function

$$J(\mathbf{w}) = -\sum_{n=1}^{N} y_n \log\left(h(\mathbf{x}_n)\right) - (1-y_n)\log\left(1-h(\mathbf{x}_n)\right)$$

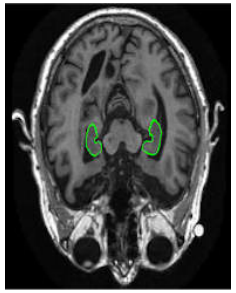
3. Optimization

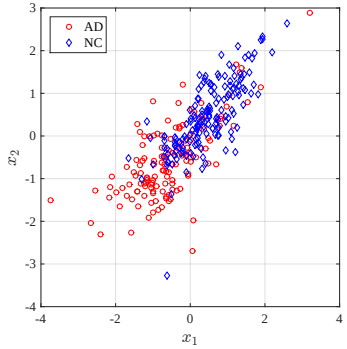
$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}}\, J(\mathbf{w})$$

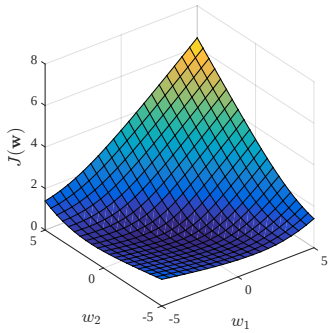
Toy example: MRI-based Alzheimer's disease diagnosis

Training data: $X = \left\{\mathbf{x}_n \in \mathbb{R}^2, y_n \in \{0,1\}\right\}_{n=1}^{300}$

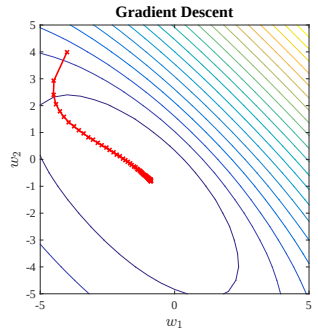
- $\mathbf{x}_n$: volume of left/right hippocampal formation in MRI
- y_n: clinical label (patient: 1, normal: 0)

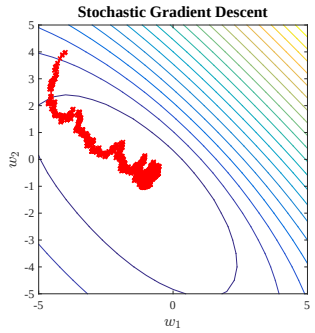






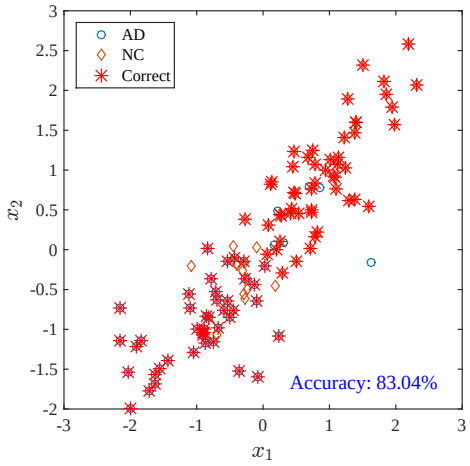
$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \mathbf{X}^T\left(h(\mathbf{X}) - \mathbf{y}\right)$$





$$\hat{y} = \left(\frac{1}{1+\exp\left(-\mathbf{x}^T\mathbf{w}^*\right)} > 0.5\right)$$

Test data: $\bar{X} = \left\{\mathbf{x}_n \in \mathbb{R}^2, y_n \in \{0,1\}\right\}_{n=1}^{112}$



Interim Summary (I)

- PRML from an optimization perspective
- Gradient vector, Hessian matrix, directional derivative
- Gradient-based Search Methods
  - First-order: steepest gradient descent
  - Second-order: Newton's method
- Least square/logistic regression

Constrained Optimization

Constrained Optimization

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad \text{subject to } \mathbf{x} \in \mathbb{S}$$

(where $\mathbb{S}$: feasible point set)

Formal statement of constrained optimization problem

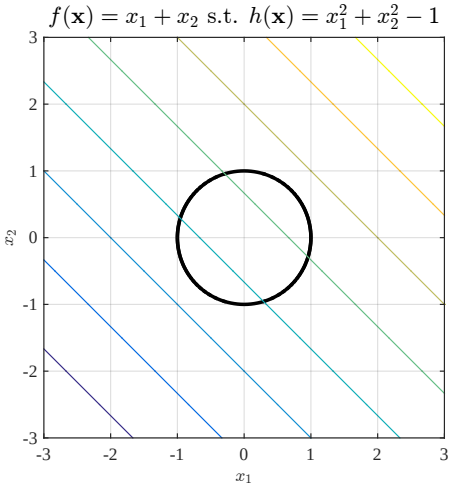
Given functions $f(\mathbf{x})$, $g_i(\mathbf{x})$, and $h_j(\mathbf{x})$ defined on some domain $\Psi \subset \mathbb{R}^n$, the optimization problem has the form:

$$\min_{\mathbf{x}\in\Psi} f(\mathbf{x}) \quad \text{subject to} \begin{cases} g_i(\mathbf{x}) \leq 0 & i = 1, 2, \ldots, k \\ h_j(\mathbf{x}) = 0 & j = 1, 2, \ldots, m \end{cases}$$

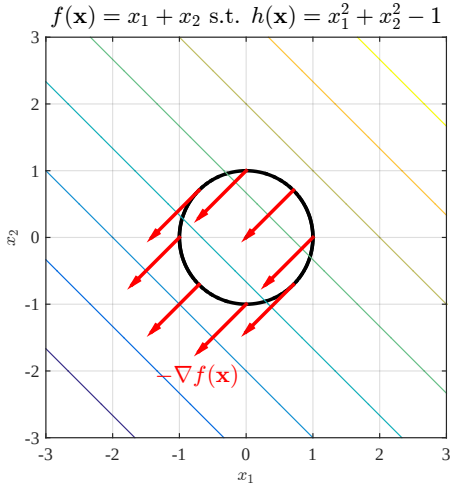
Equality Constraints

$$\min_{\mathbf{x}\in\Psi} f(\mathbf{x}) \text{ subject to } h_j(\mathbf{x}) = 0 \ \forall j$$

Example)

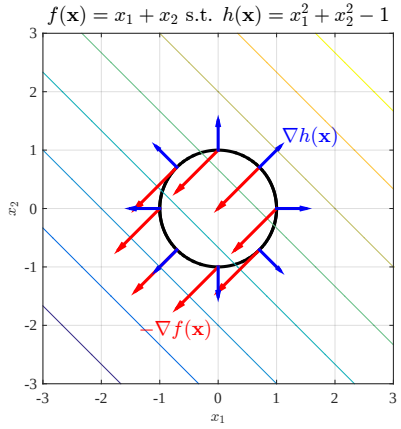


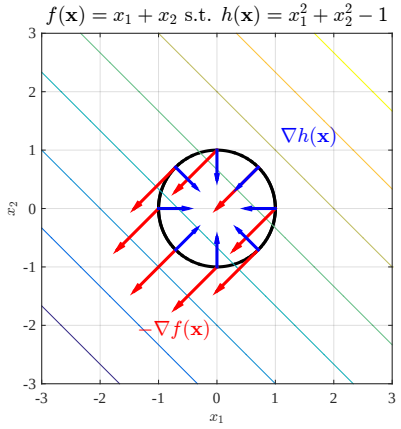
Direction of the steepest descent: $-\nabla_{\mathbf{x}} f(\mathbf{x})$



For $f(\mathbf{x}+\Delta\mathbf{x}) < f(\mathbf{x})$, $\Delta\mathbf{x}^T(-\nabla f(\mathbf{x})) > 0$

Normals to the constraint surface are given by $\nabla_{\mathbf{x}} h(\mathbf{x})$





To move a small $\Delta\mathbf{x}$ from $\mathbf{x}$ and remain on the constraint surface,
i.e., $h(\mathbf{x}) = 0$ or $-h(\mathbf{x}) = 0$,
we have to move in a direction orthogonal to $\nabla_{\mathbf{x}} h(\mathbf{x})$.

$\Delta\mathbf{x}^T(-\nabla f(\mathbf{x})) > 0$ and $\Delta\mathbf{x}^T\nabla_{\mathbf{x}} h(\mathbf{x}) = 0$

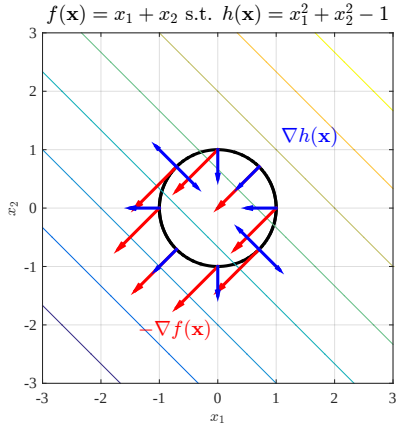
Consider the case when

$$\nabla f(\mathbf{x}) = \alpha \nabla h(\mathbf{x}).$$

Then

$$\Delta\mathbf{x}^T(-\nabla f(\mathbf{x})) = -\Delta\mathbf{x}^T\alpha\nabla h(\mathbf{x}) = 0$$

- Critical point, *constrained local optimum*



A constrained local optimum occurs at $\mathbf{x}^*$
when $\nabla f(\mathbf{x})$ is parallel to $\nabla h(\mathbf{x})$, *i.e.*, $\nabla f(\mathbf{x}) = \alpha\nabla h(\mathbf{x})$.

$\min_{\mathbf{x}\in\Psi} f(\mathbf{x})$ subject to $h_j(\mathbf{x}) = 0$ for $j = 1, \ldots, m$

Lagrangian Multipliers

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\alpha}) = f(\mathbf{x}) + \sum_{j=1}^{m} \alpha_j h_j(\mathbf{x}) = f(\mathbf{x}) + \boldsymbol{\alpha}^T \mathbf{h}(\mathbf{x})$$

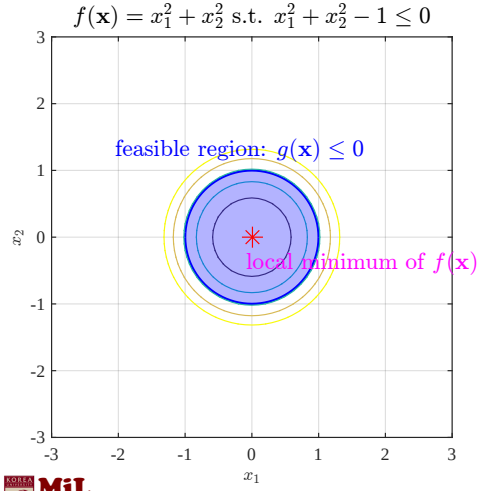
$\mathbf{x}^*$ is a local minimum $\Leftrightarrow$ there exists a unique $\boldsymbol{\alpha}^*$

1. $\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\alpha}^*) = \mathbf{0}$ $\leftarrow \nabla f(\mathbf{x}^*) = \boldsymbol{\alpha}^* \nabla \mathbf{h}(\mathbf{x}^*)$
2. $\nabla_{\boldsymbol{\alpha}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\alpha}^*) = \mathbf{0}$ $\leftarrow \mathbf{h}(\mathbf{x}) = \mathbf{0}$
3. $\mathbf{s}^T \nabla^2_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\alpha}^*) \mathbf{s} \geq 0$, $\forall \mathbf{s}$ s.t. $\mathbf{s}^T \nabla_{\mathbf{x}} h(\mathbf{x}) = 0$ $\leftarrow$ Positive definite Hessian

Inequality Constraints

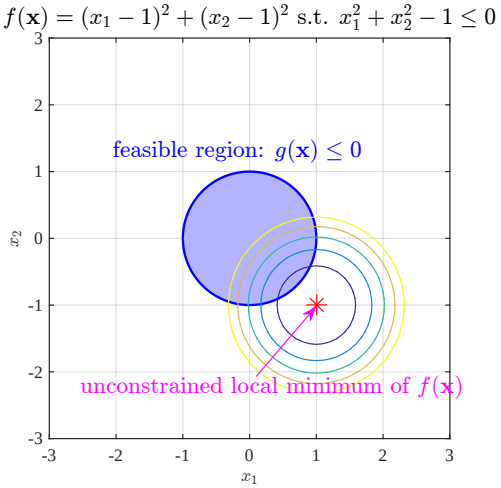
$\min_{\mathbf{x}\in\Psi} f(\mathbf{x})$ subject to $g_i(\mathbf{x}) \leq 0 \ \forall i$

Example)

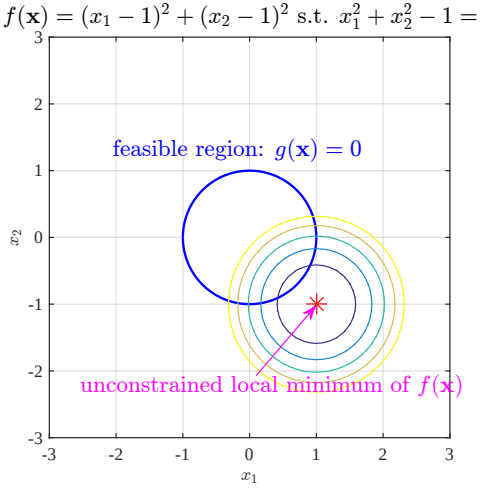


- Constraint is not active at the local minimum ($g(\mathbf{x}^*) < 0$).
- The local minimum is identified by the same conditions as in the unconstrained case.

Example)



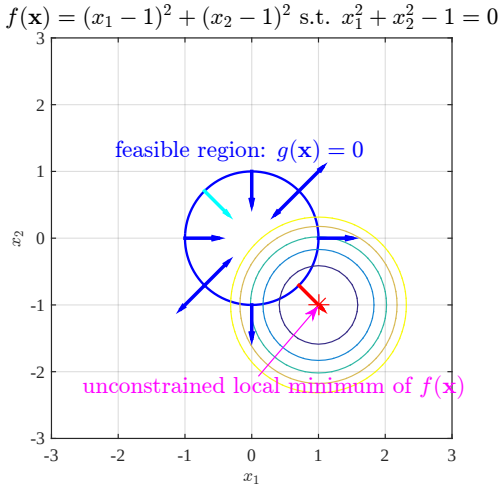
- The constrained local minimum: on the surface of the constraint surface
- An optimization problem with an equality constraint: $g(\mathbf{x}) = 0$



A constrained local optimum:

$-\nabla_{\mathbf{x}} f(\mathbf{x}) = \lambda \nabla_{\mathbf{x}} g(\mathbf{x})$ and $\lambda > 0$

(*i.e.*, $-\nabla_{\mathbf{x}} f(\mathbf{x})$ and $\nabla_{\mathbf{x}} g(\mathbf{x})$ in the same direction)



$-\nabla_{\mathbf{x}} f(\mathbf{x})$ points away from the feasible region

$$\min_{\mathbf{x}\in\Psi} f(\mathbf{x}) \text{ subject to } g_i(\mathbf{x}) \le 0 \text{ for } i = 1,\dots,k$$

Lagrangian Multipliers

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{i=1}^{k} \lambda_i g_i(\mathbf{x}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x})$$

$\mathbf{x}^*$ is a local minimum $\Leftrightarrow$ there exists a unique $\boldsymbol{\lambda}^*$

1. $\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) = \mathbf{0}$
2. $\lambda_i^* \ge 0$ for $i = 1,\dots,k$
3. $g_i(\mathbf{x}^*) \le 0$ for $i = 1,\dots,k$
4. $\lambda_i^* g_i(\mathbf{x}^*) = 0$ for $i = 1,\dots,k$
5. $\mathbf{s}^T \nabla_{\mathbf{x}}^2 \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) \mathbf{s} \ge 0$, $\forall \mathbf{s}$ s.t. $\mathbf{s}^T \nabla_{\mathbf{x}} g(\mathbf{x}) = 0$

Karush-Kuhn-Tucker (KKT) Conditions

$$\min_{\mathbf{x}\in\Psi} f(\mathbf{x}) \quad \text{s.t.} \begin{cases} h_j(\mathbf{x}) = 0 \text{ for } j = 1,\dots,m \\ g_i(\mathbf{x}) \le 0 \text{ for } i = 1,\dots,k \end{cases}$$

Lagrangian Multipliers

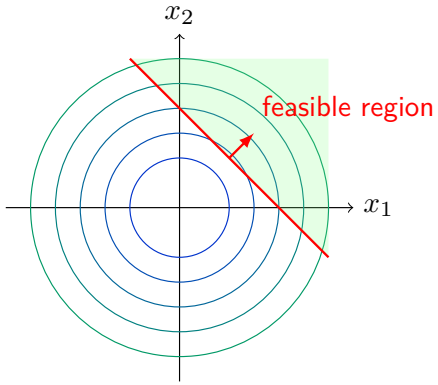
$$\mathcal{L}(\mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\alpha}^T \mathbf{h}(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x})$$

$\mathbf{x}^*$ is a local minimum $\Leftrightarrow$ there exists a unique $(\boldsymbol{\lambda}^*, \boldsymbol{\alpha}^*)$

1. $\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\alpha}^*, \boldsymbol{\lambda}^*) = \mathbf{0}$
2. $\lambda_i^* \ge 0$ for $i = 1,\dots,k$
3. $g_i(\mathbf{x}^*) \le 0$ for $i = 1,\dots,k$
4. $\lambda_i^* g_i(\mathbf{x}^*) = 0$ for $i = 1,\dots,k$
5. $\mathbf{h}(\mathbf{x}^*) = \mathbf{0}$
6. $\mathbf{s}^T \nabla_{\mathbf{x}}^2 \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) \mathbf{s} \ge 0$

Duality

$$\min_{\mathbf{x}} f(\mathbf{x}) = x_1^2 + x_2^2$$
$$\text{subject to } x_1 + x_2 \ge 1$$



- Transform the constraint into a standard form
  $$g(\mathbf{x}) = 1 - x_1 - x_2 \le 0$$

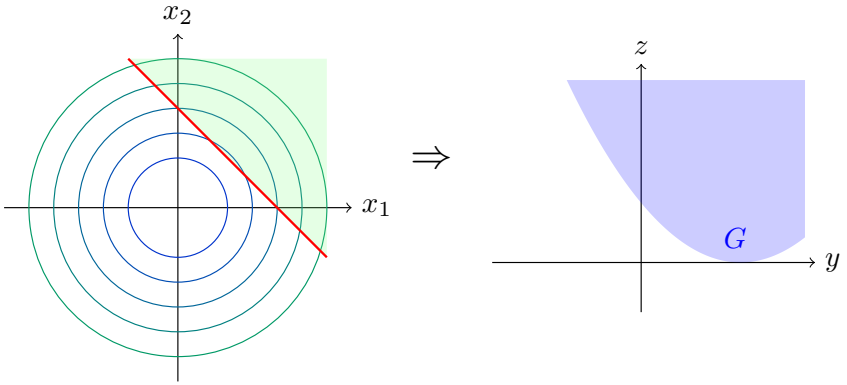
Lagrangian multipliers

$$\mathcal{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})$$

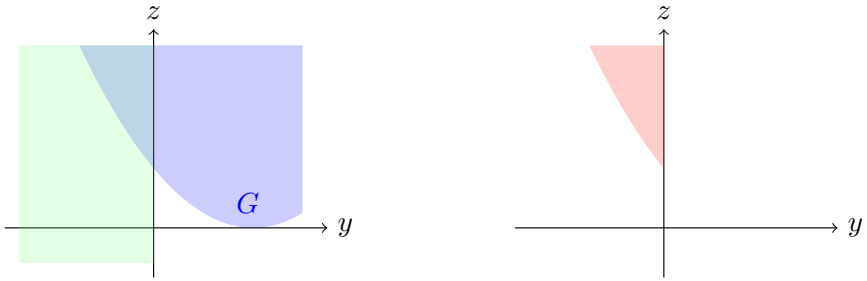
- Map each point $\mathbf{x} \in \mathbb{R}^2$ to $(g(\mathbf{x}), f(\mathbf{x})) \in \mathbb{R}^2$
- Then define the set

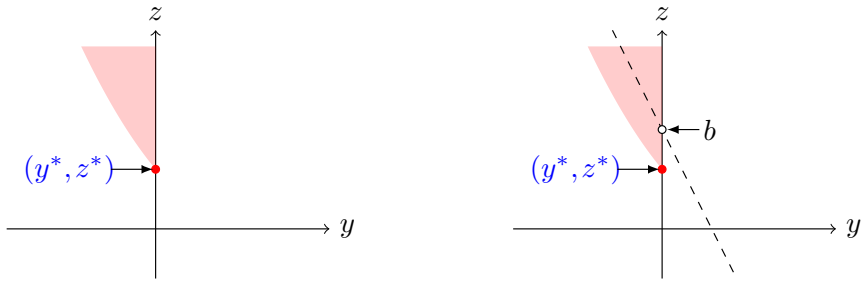
$$G = \{(y, z) \,|\, y = g(\mathbf{x}), z = f(\mathbf{x}) \text{ for } \mathbf{x} \in \mathbb{R}^2\}$$

$$\mathcal{L}(\mathbf{x}, \lambda) = z + \lambda y$$



Inequality constraint: $y = g(\mathbf{x}) \leq 0$

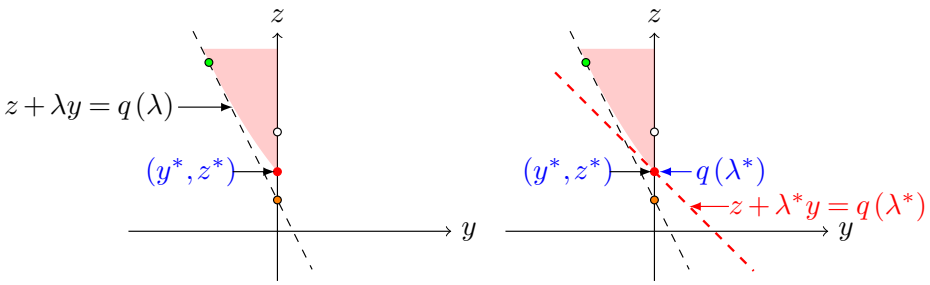




- (left) Obviously the optimal point: (y^*, z^*)
- (right) Given $\lambda \geq 0$, $\mathcal{L}(\mathbf{x}, \lambda) = z + \lambda y$

For a given $\lambda \geq 0$, $\min_{(y,z) \in G} (z + \lambda y)$

Move the line $z+\lambda y$ while contacting with G



Dual problem: $\max_{\lambda} q(\lambda)$

Generalization of Duality

$$\mathbb{S}=\left\{\mathbf{x}|\underbrace{\forall i,\ h_i(\mathbf{x})=0}_{\text{equality}}\ \text{and}\ \underbrace{\forall j,\ g_j(\mathbf{x})\le 0}_{\text{inequatlity}}\right\}:\ \text{standard form}$$

$$\mathcal{L}(\mathbf{x},\boldsymbol{\lambda},\boldsymbol{\alpha})=f(\mathbf{x})+\sum_i \lambda_i g_i(\mathbf{x})+\sum_j \alpha_j h_j(\mathbf{x})$$

$$\Rightarrow \min_{\mathbf{x}}\max_{\boldsymbol{\lambda}}\max_{\boldsymbol{\alpha}\ge\mathbf{0}}\mathcal{L}(\mathbf{x},\boldsymbol{\lambda},\boldsymbol{\alpha})$$

Significance of dual functions

- The dual approach guarantees to succeed for a certain class of functions.
- In such cases, it leads to a simpler optimization problem.
  - When the dimension of $\mathbf{x}$ is much larger than the number of constraints.
- The expression of $\mathbf{w}^*$ in terms of the Lagrangian multipliers may give some insight into the optimal solution.
  - *e.g.*, Support Vector Machine (SVM)

Primal: $\begin{cases} \min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_n \xi_n \\ \text{s.t. } y_n\left(\mathbf{w}^T\Phi(\mathbf{x}_n)+w_0\right)\ge 1-\xi_n,\ \xi_n\ge 0,\ \forall n \end{cases}$

Dual: $\begin{cases} \max_{\boldsymbol{\alpha}} -\frac{1}{2}\sum_n\sum_m \alpha_n\alpha_m y_n y_m K(\mathbf{x}_n,\mathbf{x}_m)+\sum_n \alpha_n \\ \text{s.t. } \sum_n \alpha_n y_n = 0 \text{ and } 0\le\alpha_n\le C,\ \forall n \end{cases}$

Practice in PRML

Classical Regularization

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) + \lambda \Omega(\boldsymbol{\theta})$$

- $\Omega(\mathbf{x})$: penalty or regularization term; $\lambda \geq 0$: weight control parameter
- Encode specific kinds of prior knowledge (Bayesian perspective)
- For a generic preference for a simpler model class in order to promote generalization
- To limit the capacity of models or to account for limitations of the training data
  - *e.g.*, linear regression, logistic regression, (deep) neural networks

Classical Regularization as Constrained Optimization

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) + \lambda \Omega(\boldsymbol{\theta})$$

Assume that we wanted to constrain $\Omega(\boldsymbol{\theta}) \leq k$, ($k$: some constant)

$$\mathcal{L}(\boldsymbol{\theta}, \lambda; \mathbf{X}, \mathbf{y}) = J(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) + \lambda (\Omega(\boldsymbol{\theta}) - k)$$

Solution to this constrained problem

$$\boldsymbol{\theta}^* = \min_{\boldsymbol{\theta}} \max_{\lambda \geq 0} \mathcal{L}(\boldsymbol{\theta}, \lambda)$$

Once finding the optimal value of λ^*, then the problem becomes

$$\boldsymbol{\theta}^* = \min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \lambda^*) = J(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) + \lambda^* \Omega(\boldsymbol{\theta})$$

$$\boldsymbol{\theta}^* = \min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \lambda^*) = J(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) + \lambda^* \Omega(\boldsymbol{\theta})$$

- The value of λ^* does not directly tell us the value of k.
- The relationship between k and λ^* depends on the form of J.
- Larger/smaller λ will result in a smaller/larger constraint region.

Regularized Least Square Regression

Given *i.i.d.* samples $X = \left\{\mathbf{x}_n \in \mathbb{R}^d, y_n \in \mathbb{R}\right\}_{n=1}^N$,
the aim is to build a good and useful approximation to y_n

1. Linear regression

$$y = \mathbf{x}^T \mathbf{w}$$

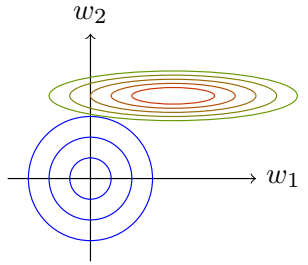
2. Regularized least square error

$$J(\mathbf{w}) = \frac{1}{2} \|\mathbf{y} - \mathbf{X}^T \mathbf{w}\|_2^2 + \frac{\lambda}{2} \|\mathbf{w}\|_2^2$$

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = -\mathbf{X} \left(\mathbf{y} - \mathbf{X}^T \mathbf{w}\right) + \lambda \mathbf{w}$$

3. Optimization

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} J(\mathbf{w})$$



Regularized Logistic Regression

Given *i.i.d.* samples $X = \left\{\mathbf{x}_n \in \mathbb{R}^d, y_n \in \{0,1\}\right\}_{n=1}^N$,
the aim is to build a good and useful approximation to y_n

1. Logistic regression

$$y = \frac{1}{1+\exp\left(-\mathbf{x}^T\mathbf{w}\right)} = h(\mathbf{x})$$

2. Cross-entropy function

$$J(\mathbf{w}) = -\sum_{n=1}^N y_n \log\left(h(\mathbf{x}_n)\right) - (1-y_n)\log\left(1-h(\mathbf{x}_n)\right) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

$$\nabla_{\mathbf{w}} J\left(\mathbf{w}\right) = \mathbf{X}^T\left(h(\mathbf{X}) - \mathbf{y}\right) + \lambda\mathbf{w}$$

3. Optimization

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}}\, J(\mathbf{w})$$

Interim Summary (II)

- Equality/inequality constraints
- Lagrangian multipliers
- Duality
- Regularization: norm penalty
  - Least square/logistic regression with L_2-norm regularization

Take-home Messages

- PRML as an optimization problem
- Unconstrained optimization
  - Steeptest gradient descent
  - (Quasi-)Newton's method
- Constrained optimization
  - Lagrangian multiplier
  - KKT conditions
  - Duality
  - Regularization

References

- Y. Bengio, I. Goodfellow, and A. Courville, *Deep Learning*, MIT Press (in Preparation)
- C. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006
- T. Minka, "A Comparison of Numerical Optimizers for Logistic Regression," 2003
  (Compared different optimization algorithms for computing the MAP parameter estimate)
- A. Parkinson, R. Balling, and J. Hedengren, *Optimization Methods for Engineering Design*, Brigham Young University, 2013
- J. Snyman, *Practical Mathematical Optimization*, Springer, 2005
- J. Sullivan, Elements of Statistical Learning Course (DD3364), 2012