Lecture 10. Optimisation

COMP90051 Statistical Machine Learning

Semester 2, 2015
Lecturer: Andrey Kan

THE UNIVERSITY OF MELBOURNE

Weeks 6 – 12

- Lecturer: Andrey Kan
  * MS: Moscow
  * PhD: Melbourne
- Past: software engineer
  * Optimising compilers
  * Multimedia framework
- Present: research officer
  * Computational immunology
  * Medical image analysis

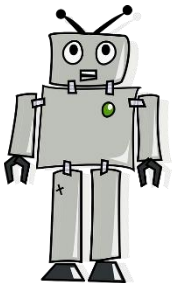
- Email: andrey.kan@unimelb.edu.au
- Office hour (same): Tuesday after lecture, DMD 6.26

Model Parameters

- All types of models involve parameters
  - Example 1: $\hat{y} = w_0 + \sum_{j=1}^{d} w_j x_j$
  - Example 2: Power Plant model with *Pr(HG|HT,FG)*, etc.
- Parameters link models to data
  - Denote parameters of a generic model as $\boldsymbol{\theta}$
  - Model predictions are $f(\boldsymbol{x}, \boldsymbol{\theta})$
- Discrepancy with training data as a function of $\boldsymbol{\theta}$
  - Example 1: $RSS(\boldsymbol{\theta}) = \sum_{i=1}^{n} (y_i - f(x_i, \boldsymbol{\theta}))^2$
  - Example 2: neg. log-likelihood $= -\sum_{i=1}^{n} ln\, f(y_i | x_i, \boldsymbol{\theta})$

Supervised learning

1. Choose/design a model
2. Choose/design discrepancy function



3. Find parameter values that minimize discrepancy with training data

learning ≈ optimisation

Also know as

* Training
* Learning
* Model fitting
* Parameter estimation

art: ClkerFreeVectorImages and OpenClipartVectors at pixabay.com (CC0)

Optimisation

- Find $\hat{\boldsymbol{\theta}} = argmin_{\boldsymbol{\theta}\in\Theta}\ D(tr.data, \boldsymbol{\theta})$
  - E.g., $\hat{\boldsymbol{w}} = \mathrm{argmin}_{\mathbf{w}} \sum_i (y_i - \mathbf{X}_{i}.\mathbf{w})^2$
  - Analytic (aka closed form) solution
  - Iterative approach

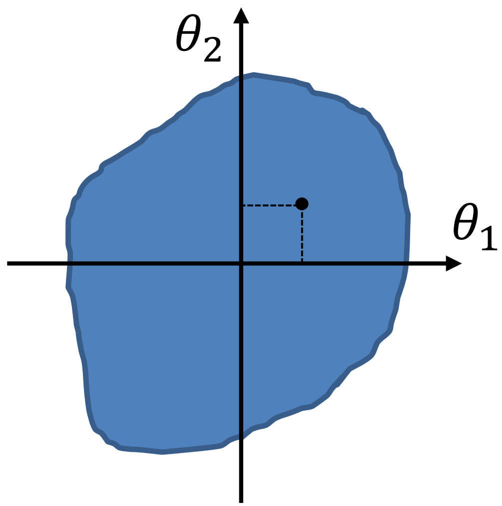
- Analytic solution
  - Known only in limited number of cases
  - Solve $\frac{\partial D}{\partial \theta_1} = \cdots = \frac{\partial D}{\partial \theta_n} = 0$
  - E.g., for linear regression $\hat{\boldsymbol{w}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$

$\boldsymbol{X}^T$ denotes matrix transpose; $\boldsymbol{X}'$ will denote first derivative

Iterative optimisation

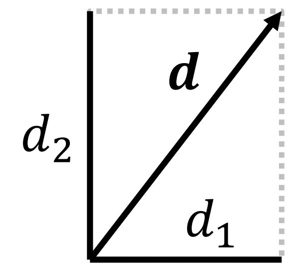
1. Initialisation: choose starting guess $\boldsymbol{\theta}^{(0)}, i = 0$
2. Compute discrepancy $D(tr.data, \boldsymbol{\theta}^{(i)})$
3. Termination: decide whether to stop
4. Update: $\boldsymbol{\theta}^{(i+1)} \leftarrow_{some\ rule} \boldsymbol{\theta}^{(i)}, i \leftarrow i + 1$
5. Go to Step 2

Gradient descent

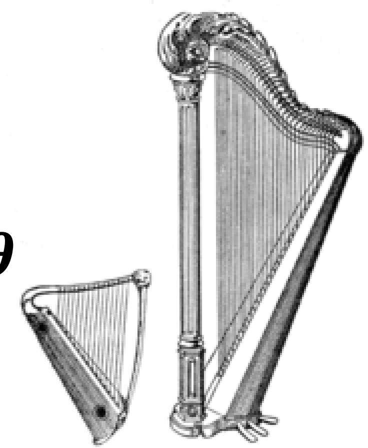


- In this example, a model has 2 parameters
- Each location corresponds to a particular combination of parameter values
- Depth indicates discrepancy between model with those values and data

Gradient



- Direction can be represented by a set of numbers $[d_1, \dots, d_2]$
- <u>Gradient (at point $\boldsymbol{\theta}$)</u> is the direction of maximal change of $D(\boldsymbol{\theta})$ when departing from point $\boldsymbol{\theta}$
  * That is, gradient is $\left[\frac{\partial D}{\partial \theta_1}, \dots, \frac{\partial D}{\partial \theta_n}\right]^T$ each computed at $\boldsymbol{\theta}$
- Shorthand notation
  * $\boldsymbol{\nabla} D \stackrel{\text{def}}{=} \left[\frac{\partial D}{\partial \theta_1}, \dots, \frac{\partial D}{\partial \theta_n}\right]^T$ computed at point $\boldsymbol{\theta}$
  * Here $\boldsymbol{\nabla}$ is the nabla symbol

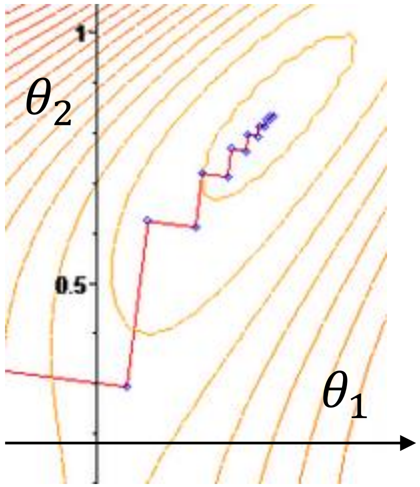

Harps, p. 984.

Gradient descent

We assume D is differentiable

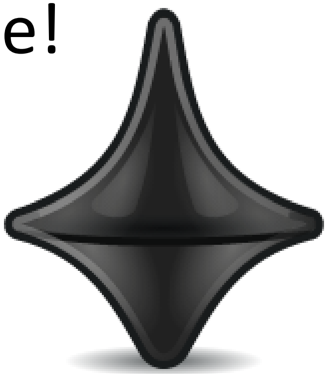
- <u>Update</u>: $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)} - \eta \boldsymbol{\nabla} D(\boldsymbol{\theta}^{(i)})$
- η is usually also updated (next slide)
- Simple method, but …
  * Convergence can be slow
  * Flat regions cause problems

plot adapted from Joris Gillis, Wikimedia Commons



Line search

- Consider optimisation iteration i
  - We first find the direction $\boldsymbol{d}$ of the next step
  - Example, $\boldsymbol{d} = -\nabla D(\boldsymbol{\theta}^{(i)})$ in gradient descent
  - But how far should we go in that direction (coefficient η)?
- Find η that gives sufficiently good decrease!
  - Optimisation within optimisation
- Example: backtracking line search
  1. Start with a large η and c, $\tau \in (0,1)$
  2. If $D\left(\boldsymbol{\theta}^{(i)} + \eta\boldsymbol{d}\right) \le D\left(\boldsymbol{\theta}^{(i)}\right) - c\eta\boldsymbol{d}^T\nabla D(\boldsymbol{\theta}^{(i)})$ then stop
  3. Update $\eta \leftarrow \tau\eta$
  4. Go to Step 2

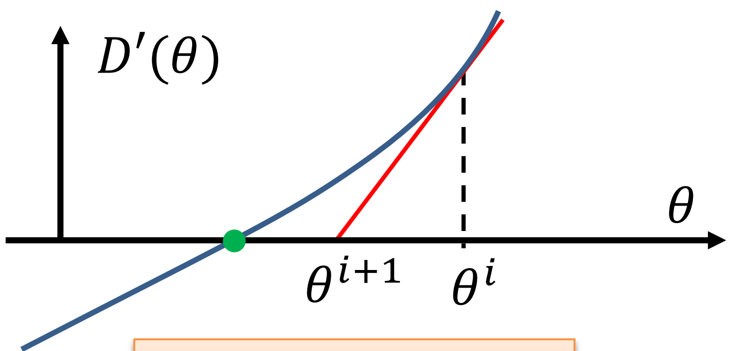


Ensures the gradient will converge to zero

art: DarKobra at Wikimedia Commons (CC3)

Newton-Raphson method

- Method for finding roots (zeros) of a function
  - In our case the function is D' (derivative of discrepancy)



We assume D is <u>twice</u>-differentiable

Tangent line:

$$y = D''(\theta^{(i)})(x - \theta^{(i)}) + D'(\theta^{(i)})$$

Solving for $y = 0$:

$$x = \theta^{(i+1)} = \theta^{(i)} - \frac{D'(\theta^{(i)})}{D''(\theta^{(i)})}$$

- <u>Update</u>: $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)} - [\boldsymbol{H}D(\boldsymbol{\theta}^{(i)})]^{-\mathbf{1}}\nabla D(\boldsymbol{\theta}^{(i)})$
  - Where $\boldsymbol{H}f(\boldsymbol{x})$ is the Hessian matrix (bunch of *second derivatives*)

Newton-Raphson method

- 1D case: $\theta^{(i+1)} = \theta^{(i)} - \frac{D'(\theta^{(i)})}{D''(\theta^{(i)})}$
  - $D(\theta^{(i)} + \Delta\theta) \approx \boxed{D(\theta^{(i)}) + D'\left(\theta^{(i)}\right)\Delta\theta + \frac{1}{2}D''\left(\theta^{(i)}\right)\Delta\theta^2}$
  - Derivative of the RHS w.r.t. $\Delta\theta$: $D'\left(\theta^{(i)}\right) + D''\left(\theta^{(i)}\right)\Delta\theta$
  - This is zero when $\Delta\theta = -\frac{D'(\theta^{(i)})}{D''(\theta^{(i)})}$

Taylor expansion of $D(\theta^{(i)})$ around $\theta^{(i)}$

Ill-posed problems

- Find $\hat{\boldsymbol{\theta}} = argmin_{\boldsymbol{\theta}\in\Omega}\ D(tr.data, \boldsymbol{\theta})$
- Ill-posed if
  * Solution is not unique OR
  * Solution does not exist
- Possible approach
  * $\min_{\mathbf{w}} \sum_i (y_i - \mathbf{X}_i.\mathbf{w})^2 + \lambda\|\mathbf{w}\|_2$
- Setting regularisation parameters require *a separate procedure*
  * E.g., cross-validation

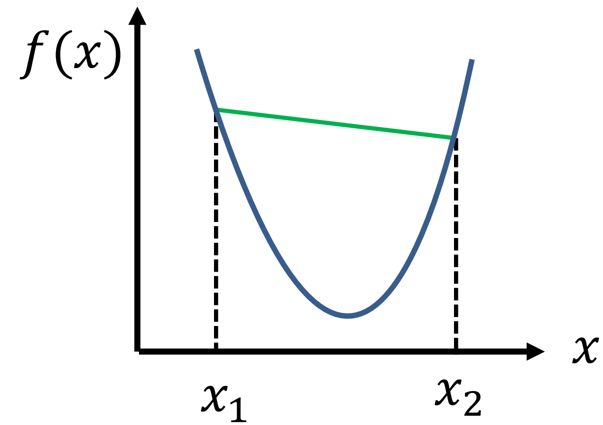
Regularise, baby, regularise

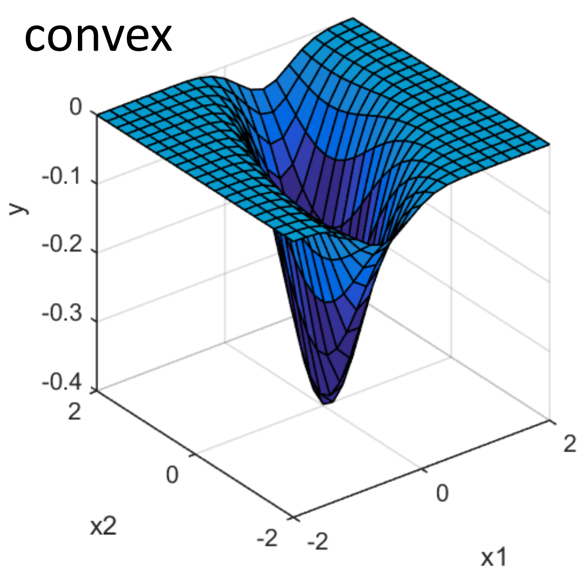
Cross-validation (revisited)

1. Split training data into [training] and [validation]
2. Choose some candidate value $\lambda = \lambda_i$
3. Train model (find parameter values for $\mathbf{w}$) using λ_i
    1. Here λ_i values is <u>fixed</u>
    2. Find $\mathbf{w}$ that minimise discrepancy with training data
4. Test on the validation set
5. Repeat with a different value of $\lambda = \lambda_{i+1}$

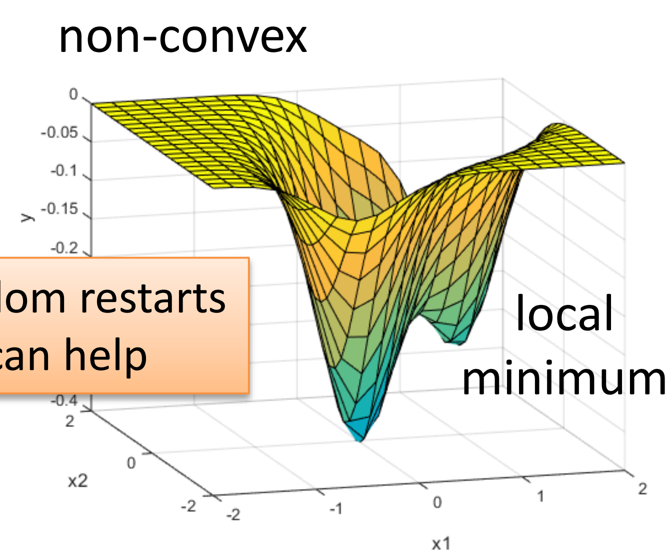
Convex functions

- Funciton is convex if the line segment between any two points lies above the graph
  - More formally, $\forall x_1, x_2$ and $\forall t \in [0,1]$:
  - $f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$
- For convex functions, gradient descent and NR method converge to the global minimum





Random restarts can help



Summary

- Learning ≈ optimisation
  - Find model parameters that minimise discrepancy with training data
- Before doing optimisation
  - Check if the problem is well-posed
  - Check if the problem is convex
- Optimisation is hard in general
  - Particular models can have analytical solution
  - Convexity helps
  - Being able to compute derivatives helps
- *<u>Design of machine learning methods is heavily influenced by optimisation considerations</u>*