Optimization in engineering applications

Jitkomut Songsiri

Department of Electrical Engineering
Faculty of Engineering
Chulalongkorn University

CUEE

February 1, 2024

Outline

Application outlines

selected topics

- power system
  - energy management system (EMS)
  - unit commitment
  - economic dispatch
- traffic network
- portfolio optimization
- regression
  - linear leasts-quares and its variants
  - nonlinear least-squares: e.g. data fitting
  - neural networks
- classification: logistic regression, SVM, ANN
- regularization techniques: see separate handouts (optim_regularization.pdf)

Optimization in power systems

Optimization in power system

optimal load scheduling of generating plants involves 2 problems:

- unit commitment: select generating units to meet the demand and provide a reserve (design over a time period)
- economic dispatch: allocate power generation from different units to minimize the cost of supply under necessary constraints

two possible ways to formulate a problem

1. **basic setting:** neglect power system network equations (considered here)
   - optimization variables are power generations by n units (x)
2. **realistic setting:** there are relations among bus voltage (v), power line flow p, power generations (x), and power demanded (d) by loads (as differential equations)
   - variables are v, p, x, d

Battery management system

- how to design a command to charge/discharge EV battery
- how to manage a power consumption according to TOU
- information of available power generations is given, *e.g.*, day-ahead solar irradiance forecasts is obtained first (as a problem parameter)

Economic dispatch (ED)

setting: there are n generating units and each is indexed by i

objective function: operating cost of power plant (generator)

$$\begin{aligned} f(x_1, x_2, \ldots, x_n) &= \sum_{i=1}^{n} [\ a_i + b_i x_i + (1/2) c_i x_i^2 \ ] \ \ (\$/\text{hour}) \\ &\triangleq (1/2) x^T C x + b^T x + \mathbf{1}^T a \end{aligned}$$

- often assumed as a **quadratic** function of power output (mainly the cost of fuel)
- $x = (x_1, x_2, \ldots, x_n)$ is the power output of n units
- a_i, b_i, c_i are positive coefficients of the cost function of ith unit
- actual unit operating cost can be nonlinear in x_i

incremental fuel cost of power plant: $\frac{df}{dx_i} = b_i + c_i x_i$ (\$/MWh)

Constraints in economic dispatch

variables and parameters

- power output of n generators $x = (x_1, x_2, \ldots, x_n)$ (variable)
- power demanded by m loads $d = (d_1, d_2, \ldots, d_m)$ (given parameters)

1. power flow equation: generated = demand + loss

$$\sum_{i=1}^{n} x_i - \sum_{i=1}^{m} d_i - \ell(x_1, x_2, \ldots, x_n) = 0$$

2. generation limit: $x_{\min} \preceq x \preceq x_{\max}$

ED1: neglect line loss and generator limit

minimize the cost of operating subject to power flow equation

$$\begin{array}{ll} \text{minimize} & f(x) := \sum_{i=1}^{n} [\ a_i + b_i x_i + (1/2) c_i x_i^2\ ] \\ \text{subject to} & \sum_{i=1}^{n} x_i - \sum_{i=1}^{m} d_i = 0 \end{array}$$

vector formulation: given C, b, a, d (cost coef: $C \succ 0, a, b, d \succ 0$)

$$\underset{x}{\text{minimize}}\ \ f(x) := (1/2) x^T C x + b^T x + \mathbf{1}^T a \ \ \text{subject to}\ \ \mathbf{1}^T x = \mathbf{1}^T d$$

where

$$C = \begin{bmatrix} c_1 & & & \\ & c_2 & & \\ & & \ddots & \\ & & & c_n \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}, \quad a = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$$

quadratic programming (QP) with a linear equality constraint (solve KKT system)

ED2: neglect line loss

minimize the cost of operating subject to power flow equation and generator limits

$$
\begin{array}{ll}
\text{minimize} & f(x) := (1/2)x^T C x + b^T x + \mathbf{1}^T a \\
\text{subject to} & \mathbf{1}^T x = \mathbf{1}^T d \\
& x_{\min} \preceq x \preceq x_{\max}
\end{array}
$$

given parameters: $C, b, a, d, x_{\min}, x_{\max}$

- constraint set is smaller than ED problem on page 9, so the optimal value is higher
- can be cast as a QP where the inequality constraints can be wrapped up as

$$
\begin{bmatrix} I \\ -I \end{bmatrix} x \preceq \begin{bmatrix} x_{\min} \\ x_{\max} \end{bmatrix}
$$

- the inequality constraint is a **box constraint** and MATLAB has an option to accept this form directly

ED3: include line loss and generator limits

minimize the cost of operating subject to power flow equation and generator limits

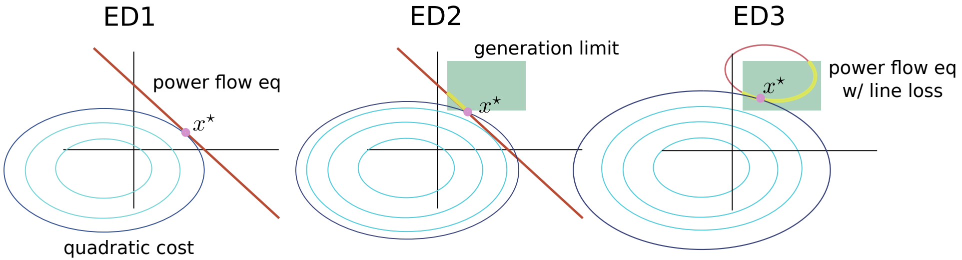
$$\begin{array}{ll} \text{minimize} & f(x) := (1/2)x^T C x + b^T x + \mathbf{1}^T a \\ \text{subject to} & \mathbf{1}^T x - \ell(x) - \mathbf{1}^T d = 0 \\ & x_{\min} \preceq x \preceq x_{\max} \end{array}$$

- $\ell(x)$ can be modeled as a quadratic function of x (more details in power system on this assumption)

$$\ell(x) = (1/2)x^T L x = (1/2)\sum_i l_{ii} x_i^2 + \sum_{i \neq j} l_i l_j x_i x_j$$

- the problem is nonlinear optimization (due to nonlinearity in $\ell(x)$)
- the problem is more nonlinear when $f(x)$ and $\ell(x)$ are any nonlinear functions
- `fmincon` in MATLAB solves the problem using the interior-point method

Economic dispatch optimization



- as the constraint is more stringent, the minimized cost is higher (optimal level set is bigger)
- as long as the cost is quadratic, the ED1 and ED2 are simple quadratic programming, while ED1 does not require an iterative method, just solve linear KKT system
- the complication of ED3 solely depends on the line loss function

Optimization in finance

Markowitz portfolio optimization

setting:

- $r = (r_1, r_2, \ldots, r_n) \in \mathbf{R}^n$; r_i is the (random) return of asset i
- the return has the mean $\bar{r}$ and covariance Σ

optimization variable: $x \in \mathbf{R}^n$ where x_i is the portion to invest in asset i

problem parameters: $\Sigma \succeq 0, \bar{r} \in \mathbf{R}^n, \gamma > 0$

$$\begin{array}{ll} \text{minimize} & -\bar{r}^T x + \gamma x^T \Sigma x \\ \text{subject to} & x \succeq 0, \quad \mathbf{1}^T x = 1 \end{array}$$

- $\mathbf{var}(r^T x) = x^T \Sigma x$ is the risk of the portfolio
- the goal is to maximize the expected return while minimize the risk
- γ is the risk-aversion parameter controlling the trade-off

Risk minimization with fixed return

setting: consider returns of n assets in T periods

- $R \in \mathbf{R}^{T \times n}$: R_{ij} is the gain of asset j in period i (%)
- $w \in \mathbf{R}^n$: asset allocation (or weight) where $\mathbf{1}^T w = 1$
- $r \in \mathbf{R}^T$: r_i is the return (of all assets) in period i, so $r = Rw$
- total portfolio value in period t is

$$V_t = V_1(1 + r_1)(1 + r_2) \cdots (1 + r_{t-1})$$

  and can be approximated when r_t is small as $V_{T+1} \approx V_1 + T\,\mathbf{avg}(r)V_1$
- unlike Markowitz that used statistical property of the returns, here we use a set of actual (or realized) returns
- as seen in Markowitz formulation, w that minimize risk for a given return is called **Pareto optimal**

Risk minimization with fixed return

goal: fix the return to a value ρ and minimize the risk over all portfolios

- the portfolio return is given by $\mathbf{avg}(r) = (1/T)\mathbf{1}^T(Rw) \triangleq \mu^T w = \rho$
- the risk is $\mathbf{var}[r] = (1/T)\|r - \mathbf{avg}(r)\|^2 = (1/T)\|r - \rho\mathbf{1}\|^2$

the problem of minimizing the risk with return ρ is

$$\begin{array}{ll} \text{minimize} & \|Rw - \rho\mathbf{1}\|^2 \\ \text{subject to} & \begin{bmatrix} \mathbf{1}^T \\ \mu^T \end{bmatrix} w = \begin{bmatrix} 1 \\ \rho \end{bmatrix} \end{array}$$

with variable $w \in \mathbf{R}^n$ and parameters R, ρ, μ

(no non-negative constraint in w – this gives quadratic programming with linear equality)

Traffic network optimization

Traffic network problem

setting: $i = 1, 2, \ldots, n$ is an intersection node; road network topology is given

problem parameters:

t_{ij}	travel time when traffice is light
α_{ij}	the rate at which travel time increases as the traffic gets heavier
c_{ij}	road capacity (a maximum number of cars per hour)
N	a road network (of interest) has a volume of N cars per hour

optimization variable: x_{ij} is the number of cars entering the road per hour

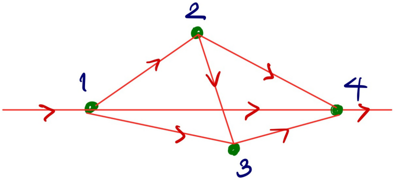
prior knowledge: T_{ij}, the travel time between node i, j can be modelled by

$$T_{ij} = t_{ij} + \alpha_{ij} \frac{x_{ij}}{1 - x_{ij}/c_{ij}}$$

- we wish to minimize the total travel time for a volume of N cars per hour
- constraints can be derived from physical structures of road network

Example of traffic network optimization

$$
\begin{array}{ll}
\text{minimize} & f_0(x) := \sum_{ij} x_{ij} T_{ij}(x_{ij}) \\
\text{subject to} & x_{12} + x_{13} + x_{14} = N \\
& x_{12} - x_{23} - x_{24} = 0 \\
& x_{13} + x_{23} - x_{34} = 0 \\
& x_{14} + x_{24} + x_{34} = N \\
& 0 \leq x_{ij} \leq c_{ij} - \epsilon, \quad 1 \leq i, j \leq n
\end{array}
$$



- the objective is to minimize the sum of travel times for all cars
- the constraints indicate that all cars entering intersection also leave the intersection
- $\epsilon > 0$ is introduced to avoid the objective function undefined
- the objective function is nonlinear in x_{ij}
- the constraints are linear equalities and inequalities

Classification problems

Linear binary classification

- logistic regression using cross-entropy cost $(y = 0, 1)$
- logistic regression using softmax cost $(y = \pm 1)$
- hyberbolic tangent $(y = \pm 1)$

the **logistic sigmoid function**

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad \Rightarrow \quad \sigma(wx) = \frac{1}{1+e^{-wx}}, \quad \sigma^{-1}(x) = \log\left(\frac{x}{1-x}\right)$$

- a cdf of logistic distribution (so the value ranges from 0 to 1)
- differentiable and a good approximation for the step function (by varying w)

Logistic regression

modeling: $\{(x_i, y_i)\}_{i=1}^N$ are explained and response variables where $y_i = 0, 1$

- use the logistic function to define $P(y=1)$; hence, the logit (log odds) is

$$\log \frac{P(y=1)}{P(y=0)} = \sigma^{-1}(P(y=1)) = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n \triangleq z^T \beta \quad \text{where } z = (1, x)$$

- the log-likelihood of y_i given x_i is

$$\ell(y_i | x_i; \beta) = y_i \log(\sigma(z_i^T \beta)) + (1 - y_i) \log(1 - \sigma(z_i^T \beta))$$

the **logistic regression** with the **cross-entropy loss** is

$$\text{minimize } f_0(\beta) := -\sum_{i=1}^N y_i \log(\sigma(z_i^T \beta)) + (1 - y_i) \log(1 - \sigma(z_i^T \beta))$$

with variable $\beta \in \mathbf{R}^{n+1}$ and parameters $y_i \in \mathbf{R}$, $z_i = (1, x_i) \in \mathbf{R}^{n+1}$ for $i = 1, \ldots, N$

Logistic regression

the gradient and Hessian of $f_0(\beta)$ are (please verify)

$$\begin{aligned}\nabla_\beta f_0(\beta) &= -\sum_{i=1}^N (y_i - \sigma(z_i^T\beta))z_i, \\ \nabla_\beta^2 f_0(\beta) &= \sum_{i=1}^N z_i z_i^T \sigma(z_i^T\beta)(1-\sigma(z_i^T w))\end{aligned}$$

the Hessian is positive semidefinite; hence $f_0(\beta)$ is convex

Soft-max cost

soft-max and cross-entropy cost are equivalent upon the change of label to $y_i = \pm 1$

- we employ the point-wise **log error** cost

$$g(\beta) = -\log(\sigma(x^T\beta)), \text{ if } y = 1, \quad \text{and} \quad g(\beta) = -\log(1 - \sigma(x^T\beta)), \text{ if } y = -1$$

- use that $1 - \sigma(x) = \sigma(-x)$ and combine the two cases of g into a single one

$$g(\beta) = -\log(\sigma(yx^T\beta))$$

- use the definition of sigmoid function to obtain the **soft-max** cost

$$\text{minimize} \quad f_0(\beta) = \sum_{i=1}^{N} \log(1 + e^{-y_i x_i^T \beta})$$

(it can be shown that $f_0(\beta)$ is convex)

Hyperbolic tangent cost

adjust the version of using sigmoid to approximate the step function

- the **hyperbolic tangent function** is just a transform of $\sigma(x)$

$$\tanh(x) = 2\sigma(x) - 1 = \frac{1 - e^{-x}}{1 + e^{-x}} \quad \text{(values range between -1 and 1)}$$

- the regression problem (used with labels $y_i = \pm 1$) is

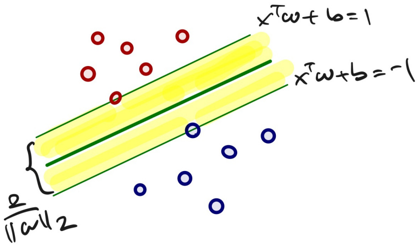
$$\text{minimize} \quad \sum_{i=1}^{N} (\tanh(x_i^T \beta) - y_i)^2$$

with variable β and problem parameters (x_i, y_i)

consider the shape of $\tanh(x)$ function (and then squared) – is the problem convex ?

Support vector machine

setting: given $\{(x_i, y_i)\}_{i=1}^N$ where $x_i \in \mathbf{R}^n$ are data with label $y_i \in \{1, -1\}$



modeling:

- the goal is to find a hyperplane $x^T w + b$ to classify data into two classes
- the distance between two hyperplanes $x^T w + b = \pm 1$ is $2/\|w\|_2$
- for $i = 1, 2, \ldots, N$ data from each class satisfy

$$y_i = 1 : x_i^T w + b \geq 1, \text{ and } y_i = -1 : x_i^T w + b \leq -1 \quad \Rightarrow \quad y_i(x_i^T w + b) \geq 1$$

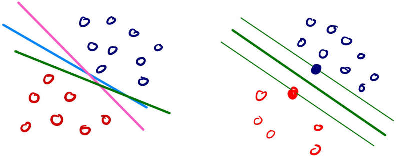
Hard-margin SVM

problem parameters: $x_i \in \mathbf{R}^n$ and $y_i \in \mathbf{R}$ for $i = 1, \ldots, N$

optimization variables: $w \in \mathbf{R}^n, b \in \mathbf{R}$

$$\text{minimize} \quad \|w\|_2^2 \quad \text{subject to} \quad y_i(x_i^T w + b) \geq 1, \; i = 1, 2, \ldots, N$$

- data are classified by separating hyperplane with maximized margin (right figure)
- if feasible, the data from two classes are separated perfectly
- the decision boundary pass through points from both classes– these points are called **support vectors**

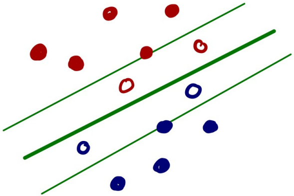


Soft-margin SVM

problem parameters: $x_i \in \mathbf{R}^n$ and $y_i \in \mathbf{R}$ for $i = 1, \ldots, N, \lambda > 0$

optimization variables: $w \in \mathbf{R}^n, b \in \mathbf{R}, z \in \mathbf{R}^N$

$$
\begin{array}{ll}
\text{minimize} & (1/2)\|w\|_2^2 + \lambda \mathbf{1}^T z \\
\text{subject to} & y_i(x_i^T w + b) \geq 1 - z_i, \quad i = 1, 2, \ldots, N \\
& z \succeq 0
\end{array}
$$



- z_i is called a *slack variable*, allowing some of the hard constraints to be relaxed
- if $z_i > 0$ at optimum, the ith data point is relaxed to lie inside the buffer zone
- the *regularization (penalty) parameter* λ controls the trade-off between maximizing the margin and the number of points in the tube

Soft-margin SVM

- the original hard constraint relates to the **margin-perceptron cost**

$$y_i(x_i^T w + b) \geq 1 \quad \iff \quad \max(0, 1 - y_i(x_i^T w + b)) = 0$$

- another formulation of soft-margin SVM is to use the hinge loss

$$y_i(x_i^T w + b) \geq 1 - z_i, \quad \mathbf{1}^T z = \sum_{i=1}^N \max(0, 1 - y_i(x_i^T w + b))$$

  and put the formulation as a single cost function (aka hinge primal problem)

$$\text{minimize} \quad (1/2)\|w\|_2^2 + \lambda \sum_{i=1}^N \max(0, 1 - y_i(x_i^T w + b))$$

- note that max function is non-differentiable at zero

Sparse SVM

use $\|w\|_1$ in the objective

$$\text{minimize}_{w,b} \quad \lambda\|w\|_1 + \frac{1}{N}\sum_{i=1}^N \max(0, 1 - y_i(x_i^T w + b))$$

- the ℓ_1-norm encourages sparsity of the optimal w
- for such a sparse w, the product $w^T x$ involves only a few entries in x (use less feature)

Dual of soft-margin SVM

derived from duality theory, the dual problem of soft-margin SVM is

$$\begin{array}{ll} \text{maximize} & \mathbf{1}^T\alpha - (1/2)\sum_{i=1}^N \sum_{j=1}^N \alpha_i y_i x_i^T x_j \alpha_j y_j \\ \text{subject to} & \alpha \succeq 0, \quad \alpha^T y = 0, \\ & \alpha_i \leq \lambda, \quad i = 1, 2, \ldots, N \end{array}$$

with variable $\alpha \in \mathbf{R}^N$

the dual can be recognized as having a quadratic cost because

$$\sum_i \sum_j \alpha_i y_i x_i^T x_j \alpha_j y_j = \alpha^T G \alpha, \quad G_{ij} = \langle y_i x_i, y_j x_j \rangle$$

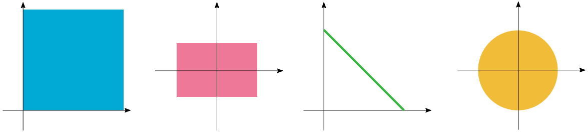
G is a Gram matrix (which is positive definite)

Regression problems in ML

Linear least-squares with constraints

regression coefficients are restricted to lie in a set

$$\text{minimize} \quad \|Ax - y\|_2^2 \quad \text{subject to} \quad x \in \mathcal{C}$$



- nonnegativity: $\mathcal{C} = \{ x \mid x \succeq 0 \}$
- variable bound: $\mathcal{C} = \{ x \mid l \preceq x \preceq u \}$
- probability distribution: $\mathcal{C} = \{ x \mid x \succeq 0, \quad \mathbf{1}^T x = 1 \}$
- norm ball constraint: $\mathcal{C} = \{ x \mid \|x - x_0\| \leq d \}$

Nonlinear least-squares: data fitting

given data $\{x_i, y_i\}_{i=1}^N$, fit $g(x;\theta)$ to y

$$\underset{\theta}{\text{minimize}} \quad (1/2)\sum_{i=1}^N (y_i - g(x_i;\theta))^2$$

- polynomial: $g(x) = a(x-b)^n$ where $\theta = (a, b, n)$
- Gaussian: $g(x) = ae^{-\frac{(x-b)^2}{c^2}} + d$ where $\theta = (a, b, c, d)$
- sum of exponential: $g(x) = ae^{bx} + ce^{dx}$ where $\theta = (a, b, c, d)$
- Fourier series: $g(x) = a_0 + \sum_{k=1}^m [a_k \cos(k\omega x) + b_k \sin(k\omega x)]$ where $\theta = (a_0, a_1, \ldots, a_m, b_1, \ldots, b_m, \omega)$
- rational model: $g(x) = \frac{p_1 x^2 + p_2 x + p_3}{x^3 + q_1 x^2 + q_2 x + q_3}$ where $\theta = (p_1, p_2, p_3, q_1, q_2, q_3)$

Robust least-squares

consider the LS problem

$$\underset{x}{\text{minimize}} \quad \|Ax - b\|_2$$

but A may have variation or some uncertainty

we can treat the uncertainty in A in different ways

- A is deterministic but belongs to a set
- A is stochastic

Worst-case robust least-squares

describe the uncertainty by a set of possible values for A:

$$A \in \mathcal{A} \subseteq \mathbf{R}^{m \times n}$$

the problem is to minimize the worst-case error:

$$\underset{x}{\text{minimize}} \quad \sup_{A} \; \{\|Ax - y\|_2 \mid A \in \mathcal{A}\}$$

- always a convex problem
- its tractablity depends on the description of $\mathcal{A}$

Example: worst-case robust LS

given $\mathcal{A} = \{\bar{A} + E \mid \|E\|_F \leq e\}$

- meaning: each column in A corresponds to measurements of a variable recorded thru a sensor given with noise RMS
- define $w = \bar{A}x - y$, the worst-case norm-2 can be calculated by

$$
\begin{aligned}
\|Ax - y\|^2 = \|Ex + w\|^2 &= x^T E^T E x + 2 w^T E x + \|w\|^2 \\
&\leq \lambda_{\max}(E^T E)\|x\|^2 + 2\,\mathbf{tr}((wx^T)^T E) + \|w\|^2 && (1) \\
&\leq \|E\|_F^2 \|x\|^2 + 2\|wx^T\|_F \|E\|_F + \|w\|^2 && (2) \\
&\leq e^2\|x\|^2 + 2e\|w\|\|x\| + \|w\|^2 = (e\|x\| + \|w\|)^2 && (3)
\end{aligned}
$$

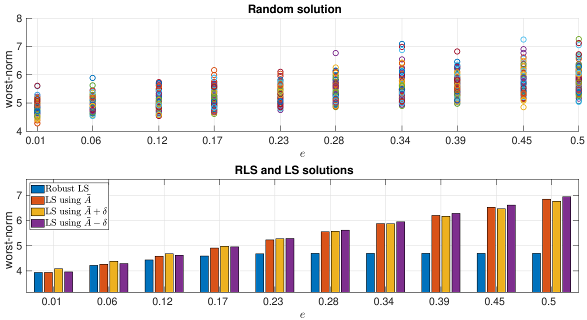
- the worst-case norm is attained when $E = \alpha w x^T$ where $\alpha = e/\|w\|\|x\|$

$$
\sup_{A \in \mathcal{A}} \|Ax - y\|_2 = \|\bar{A}x - y\|_2 + e\|x\|_2
$$

it is a second-order cone programming

Simulation of robust LS

: $\bar{A} \in \mathbf{R}^{20\times 5}$ and $e = 0.1$ (used for RLS estimation)



- compare robust LS (RLS) with LS using $\bar{A}, \bar{A}-\delta, \bar{A}+\delta$ for $\delta = 0.01$
- compute worst norm $\|\bar{A}x - y\|_2 + e\|x\|_2$ as e varies using x from various methods
- (top) worst-norms of random solution x are high and widely spread
- (bottom) worst-norms of RLS are relatively low and are not sensitive to e

Example 2:

let $U = \begin{bmatrix} u_1 & u_2 & \cdots & u_n \end{bmatrix}$

uncertainty in A is prescribed as upper bounds of 2-norm of each columns in U

$$\mathcal{A} = \{\bar{A} + U \mid \|u_j\|_2 \leq a_j,\ j = 1, 2, \ldots, n\}$$

it can be shown that

$$\sup_{\|u_j\|_2 \leq a_j} \|\bar{A}x - y + Ux\|_2 = a^T|x| + \|\bar{A}x - y\|_2$$

where the supremum is attained when each column of U is selected as

$$u_j = \frac{c_j \mathbf{sign}(x_j)}{\|\bar{A}x - y\|_2} \cdot (\bar{A}x - y), \quad j = 1, 2, \ldots, n$$

- the robust LS can be cast as a second-order cone programming
- the term $a^T|x|$ can be viewed as a weighted ℓ_1-regularization

Worst-case Chebyshev approximation

setting: find $\sup_U \|(Ax - y\|_\infty$ where uncertainty in A is prescribed as upper bounds of ∞-norm of each columns in U

$$\mathcal{A} = \{\bar{A} + U \mid \|u_j\|_\infty \leq a_j,\ j = 1, 2, \ldots, n\}$$

it can be shown that

$$\sup_{\|u_j\|_\infty \leq a_j} \|\bar{A}x - y + Ux\|_\infty = a^T|x| + \|\bar{A}x - y\|_\infty$$

where the supremum is attained when

- let j be the index for which $\|w\|_\infty = |w_j|$
- for each column u_k, for $k = 1, \ldots, n$, set all entries as zero, except the jth as

$$(u_k)_j = \mathbf{sign}(x_k w_j) \cdot a_k = \begin{cases} a_k, & \text{if } x_k \text{ and } w_j \text{ has the same sign} \\ -a_k, & \text{otherwise} \end{cases}$$

Stochastic robust least-squares

when A is a random variable, so we can describe A as

$$A = \bar{A} + U,$$

where $\bar{A}$ is the average value of A and U is a random matrix

use the expected value of $\|Ax - y\|$ as the objective:

$$\underset{x}{\text{minimize}} \quad \mathbf{E}\|Ax - y\|_2^2$$

expanding the objective gives

$$\begin{aligned} \mathbf{E}\|Ax - y\|_2^2 &= (\bar{A}x - y)^T(\bar{A}x - y) + \mathbf{E}x^T U^T U x \\ &= \|\bar{A}x - y\|_2^2 + x^T P x \end{aligned}$$

where $P = \mathbf{E}[U^T U]$

this problem is equivalent to

$$\underset{x}{\text{minimize}} \quad \|\bar{A}x - y\|_2^2 + \|P^{1/2}x\|_2^2$$

with solution $x = (\bar{A}^T\bar{A} + P)^{-1}\bar{A}^T y$

- a form of a regularized least-squares
- balance making $\bar{A}x - y$ small with aiming to get a small x (so that the variation in Ax is small)
- Tikhonov regularization is a special case of robust least-squares:

  when U has zero mean and uncorrelated variables, *i.e.*, $\mathbf{E}[U^TU] = \delta I$

example: $u_{ij} \sim \mathcal{U}(-a_j, a_j)$ and assume columns of U are uncorrelated

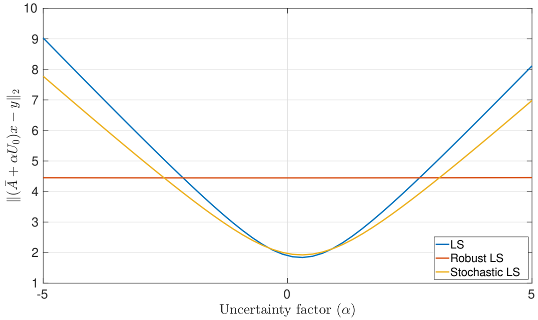
$$P_{ij} = 0, \quad P_{ii} = \mathbf{E}[u_i^T u_i] = \mathbf{E}\left[\sum_{k=1}^m u_{ki}^2\right] = m\,\mathbf{var}[u_{ki}] = ma_j^2/3$$

Comparison between robust and stochastic LS

two comparable formulations

- robust LS: $A \in \mathcal{A} = \{\bar{A} + U \mid |u_{ij}| \leq a_j,\ j = 1, 2, \ldots, n\}$
- stochastic LS: $A = \bar{A} + U$ where $u_{ij} \sim \mathcal{U}(-a_j, a_j)$
- the ground truth A follows $A = \bar{A} + \alpha U_0$
  - columns of U_0 is perturbed from $\bar{A}$ by 1-3%
  - vary $\alpha \in [-5, 5]$
- we plot $\|(\bar{A} + \alpha U_0)x - y\|_2$ where x are solutions from
  - ordinary LS
  - robust LS
  - stochastic LS

Comparison between robust and stochastic LS



- OLS is certainly the best when $\alpha = 0$ (no uncertainty)
- robust LS solution is most robust to uncertainty while LS solution is most sensitive to α and stochastic LS performance lies in between

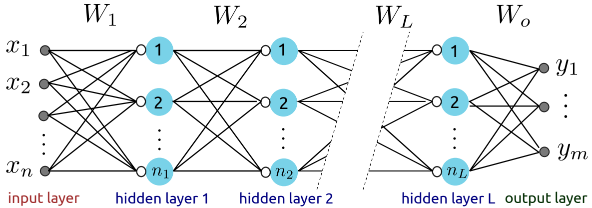
Feedforward neural networks

Feedforward neural networks

- structure and parameters
- mathematical relations
- loss functions

Feedforward NN structure

fully connected L-hidden layers; each of which has n_i units and the weight matrix W_i



- $x = (x_1, x_2, \dots, x_p)$ is the input (assume the first element is constant)
- $y = (y_1, y_2, \dots, y_m)$ is the output (or target)
- hidden-layer weight matrices: $W_1 \in \mathbf{R}^{n_1 \times p}$ and $W_j \in \mathbf{R}^{n_j \times n_{j-1}}$, $j = 2, \dots, L$
- output-layer weight: $W_0 \in \mathbf{R}^{m \times (n_L + 1)}$
- $h : \mathbf{R}^d \to \mathbf{R}^d$ is an activation function for units in hidden layer
- $g : \mathbf{R}^m \to \mathbf{R}^m$ is a transformation for output layer

Compact mathematical representations

linear transform of input and pass through a nonlinear activation function

- $(W_k)_{ij}$ is the weight of the kth layer that maps input i to output j (assume $x_1 = 1$, so $(W_k)_{i1}$ is a bias term)
- the functions h and g are element-wise operations
- activation function examples: step (heaviside), sigmoid, ReLU, tanh, RBF
- example: single hidden-layer of n units; tanh activation:

$$h(W_1 x) = \begin{bmatrix} \tanh[(W_1)_{11} \cdot 1 + (W_1)_{12} x_2 + \cdots (W_1)_{1p} x_p] \\ \tanh[(W_1)_{21} \cdot 1 + (W_1)_{22} x_2 + \cdots (W_1)_{2p} x_p] \\ \vdots \\ \tanh[(W_1)_{n1} \cdot 1 + (W_1)_{n2} x_2 + \cdots (W_1)_{np} x_p] \end{bmatrix} \triangleq \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_n \end{bmatrix}$$

$$z = (W_o)_0 \cdot 1 + (W_o)_1 h_1 + (W_o)_2 h_2 + \cdots + (W_o)_n h_n \ \in \mathbf{R}^m$$

Tasks of NN

the transformation of output unit depends on the task of NN

- **regression:** g is linear; $y = z = W_o h(W_1 x)$
- **multi-class classification:** g is softmax function: $g_k(z) = \frac{e^{z_k}}{\sum_{i=1}^m e^{z_k}}$, $k = 1, \ldots, m$

$$y = g(z) = g(W_o(h(W_1 x)))$$

  (y_k is the probability of classifying the input to class k)
- **binary classification:** y has a single node; g reduces to the sigmoid function

Feedforward NN as composites of nonlinear functions

example of L hidden-layer: $y = g(W_o h(W_L h(W_{L-1} h(\cdots h(W_1 x)))))$

to differentiate the notation of NN output from the true description y, we often use

$$\hat{y} = f(x; \Theta)$$

as the output of NN

- conceptually, a nonlinear function of x, parametrized by $\Theta = (W_1, \ldots, W_L, W_o)$
- nonlinearity of a model is introduced via a choice of activation function
- the overall number of parameters is specified by the **depth** (number of hidden layers) and number of **units**

Regression task of NN

let $\hat{y} = f(x; \Theta)$ be the output of neural network using input data x

$\{x_i, y_i\}_{i=1}^N$ are N-sample of input/output data; $\hat{y}_i$ is a model output from sample i

regression: loss functions that are tied with the regression task

- MSE: $(1/N)\sum_{i=1}^N \|y_i - \hat{y}_i\|_2^2$
- MAE: $(1/N)\sum_{i=1}^N \|y_i - \hat{y}_i\|_1$
- huber: $(1/N)\sum_{i=1}^N \text{huber}(r_i)$ where $r_i = y - \hat{y}_i$;

$$\text{huber}(x) = \begin{cases} (1/2)x^2, & |x| \leq M \\ M(|x| - M/2), & x > M \end{cases}$$

Binary classification task

the output unit predicts the probability of one class

- class labels have two choices: $y \in \{1, -1\}$ or $y \in \{1, 0\}$
- y is modeled to have a Bernoulli distribution: $p(y|x) = \pi^y (1-\pi)^{1-y}$
- the negative loglikelihood is aka cross-entropy:
  -$\log p(y|x) = -[y \log \pi + (1-y)(1-\pi)]$
- modeling: predict $\pi = P(y = 1|x)$ using NN (or other models); replace π by $\hat{\pi} = \hat{y}(x; \Theta)$

loss functions used to train NN for binary classification

- cross-entropy: labels are $0, 1$; $\hat{y}_i \triangleq \hat{y}_i(x_i; \Theta) = P(y_i = 1|x_i)$ (classify to class 1)

$$\text{loss} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)$$

Binary classification task

- hinge loss (or ReLU, perceptron cost): labels are $1, -1$; normalize $\hat{y}_i$ to $(-1, 1)$

$$\text{loss} = \sum_{i=1}^{N} \max(0, 1 - y_i \cdot \hat{y}_i), \quad \text{(when } \hat{y}_i \neq y_i \text{ the loss is 2)}$$

- scores motivated from F1 or **dice similarity coefficient**

$$\text{F1} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}, \quad \text{(no TN, predicting majority samples correctly)}$$

meaning: $\text{TP} = \sum_i y_i \hat{y}_i$, $\text{FP} = \sum_i (1 - y_i)\hat{y}_i$, and $\text{FN} = \sum_i y_i(1 - \hat{y}_i)$

- minimizing these losses is similar to maximizing F1 score

$$\text{soft-dice loss} = 1 - \frac{2\sum_{i=1}^{N} y_i \hat{y}_i}{\sum_{i=1}^{N} (y_i + \hat{y}_i)}, \qquad \text{squared-dice loss} = 1 - \frac{2\sum_{i=1}^{N} y_i \hat{y}_i}{\sum_{i=1}^{N} (y_i^2 + \hat{y}_i^2)}$$

K-class classification using NN

label y is a standard unit vector in $\mathbf{R}^K$

$$y = (y_1, y_2, \ldots, y_K)$$

(only one of $y_1, y_2, \ldots, y_K$ has value of 1; the rest is all zero)

- denote π_k the probability that $y = (0, 0, \underbrace{1}_{k\text{th}}, 0, \ldots, 0)$ where $\sum_{i=1}^K \pi_i = 1$
- generalize Bernoulli distribution to an K-dimensional binary variable y

$$p(y|x) = \pi_1^{y_1} \pi_2^{y_2} \cdots \pi_K^{y_K}$$

- the (conditional) loglikelihood is called **(multi-class) cross entropy**

$$\log p(y|x) = y_1 \log \pi_1 + y_2 \log \pi_2 + \cdots + y_K \log \pi_K$$

- modeling: NN has K-dimensional output units that predictds π_k's

$$\hat{y}_k = \hat{\pi}_k \approx \pi_k, \quad k = 1, \ldots, K$$

K-class classification using NN

let i be a sample index, $i = 1, \ldots, N$

cross-entropy loss: $\hat{y}_i$ is the output of the **softmax function**

$$\text{loss} = -\sum_{i=1}^{N} y_{i1}\log(\hat{y}_{i1}) + y_{i2}\log(\hat{y}_{i2}) + \cdots + y_{iK}\log(\hat{y}_{iK})$$

$$= -\sum_{i=1}^{N} \log(\hat{y}_{i,\text{correct class}}) = -\sum_{i=1}^{N} \log\left(\frac{e^{z_{i,\text{correct class}}}}{\sum_{k=1}^{K} e^{z_{ik}}}\right)$$

$z_i \in \mathbf{R}^K$ is predicted output from a model; before being mapped to probabilities

(alo referred to multi-class softmax cost, softplus cost, multi-class cross entropy loss)

Considerations in learning NN

- hidden units: properties and recent choices of activation functions (leaky/parametric ReLU, softplus, etc.)
- architecture design: determine overall structure of the network (theoretical result: **universal approximation theorem**
- recent advances in proposing new choices of **loss functions**
- model training
  - gradient-based learning requires computing derivatives of the composition: concept of **backprogation** based on chain rule in calculus
  - how a learning algorithm in optimization process affects a model capacity (which are the **effective capacity**, and representational capacity; the latter defined by the family of model)
  - computation: automatic differentiation, justification of non-differentiability of some activation functions by numerical point of view, batch/**mini-batch optim**
- regularization: ℓ_1 and ℓ_2, dropout

References

- Wijarn Wangdee, *Chapter 8: Economic Operation of Power Systems*, Lecture note on Modern Power Grid Operation and Control.
- Robust least-square is taken from Chapter 6 in Jitkomut Songsiri, *System Identification*, Chula Press, 2022
- C.C. Aggarwal, *Linear Algebra and Optimization for Machine Learning: A Textbook*, Springer, 2020
- G. Calafiore and L. El Ghaoui, *Optimization Models*, Cambridge University Press, 2014