Lecture Note on LASSO

This version: November 1, 2023

1 Motivation for LASSO

As we see from last week's note, OLS is not suitable for high-dimensional data. Specifically, when $p > n$, OLS is not well-defined. Even if $p < n$, OLS estimators can be noisy when $\frac{n}{p}$ is small since each parameter only shares a few observations. See the example below.

Example 1 (predicting CLTV with recency)**.** *Suppose that we would like the predict the one-year $CLTV_i$ (say, 1-year sales made by customer i) with the recency variable r_i. The recency variable r_i takes value $1, 2, 3, ..., 90$, where $r_i = r$ means that the last transaction made by customer i is r days ago.*[1] *The best predictor is*

$$E[CLTV|r],$$

which can be estimated by the regression

$$CTLV_i = \beta_1 \mathbb{1}\{r_i = 1\} + \beta_2 \mathbb{1}\{r_i = 2\} + \cdots + \beta_{90} \mathbb{1}\{r_i = 90\}.$$[2][3]

However, the regression can be problematic if we do not have enough observations since we have 90 parameters to estimate. Alternatively, we can run the regression

$$CTLV_i = \gamma_1 \mathbb{1}\{1 \leq r_i \leq 7\} + \gamma_2 \mathbb{1}\{8 \leq r_i \leq 14\} + \cdots + \gamma_{13} \mathbb{1}\{85 \leq r_i \leq 90\},$$

[1] Let's ignore observations with $r_i > 90$ for simplicity.

[2] In economics, we call such regression as "dummy variable" regression; in machine learning communuty, it is called "one-hot encoding".

[3] OLS regression is quite useful as the workhorse for other estimators. In econometrics, many estimators can be expressed as a regression, making computation and inference convenient.

which is easier to estimate (since it has less parameters) but biased, which just means that it is different from the conditional mean, the best predictor. This is the so-called bias-variance trade-off.

High-dimensionality natural arises in certain setups such as

1. DNA microarrays data: each variable is a gene expression and number of observations could be small since DNA sequence was expensive (back in the 90's).
2. text data: think of the example where each observations are articles, and each variable is a dummy indicates whether a specific word in the dictionary appear in that variable.

There are several ways to deal with high-dimensional data. In example 1, we use the **smoothing** technique, in which we exploit the idea that individuals with similar recency should have similar CLTV. The other popular strategy is the "penalization" technique, which we will see in the next section.

2 Penalized Regression

Penalized regression controls the model complexity by introducing penalties on non-zero/ large coefficients, which is useful especially when $n > p$ or when $\frac{n}{p}$ small.

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^{p+1}} \sum_{i=1}^{n} (y - \alpha - \boldsymbol{x}_i\boldsymbol{\beta})^2 + \lambda||\boldsymbol{\beta}||_q,$$

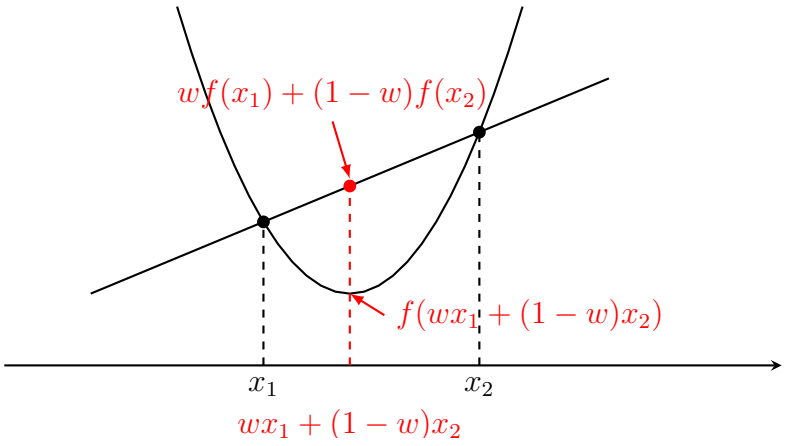
where $||\cdot||_q = (\sum_{j=1}^{p} |\beta_j|^q)^{\frac{1}{q}}$ is the L_q-norm and the parameter $\lambda > 0$ is the **penalty term** chosen by the researcher.[4] In penalized regressions, we not only want the sum of squared errors but also the (absolute value of) coefficients small. If $q = 1$, the method is called the *least absolute shrinkage and selection operato*r (LASSO). When $q = 2$, it is the *ridge regression.*

[4]We will cover how to use choose the optimal λ by *cross-validation* in the next class.

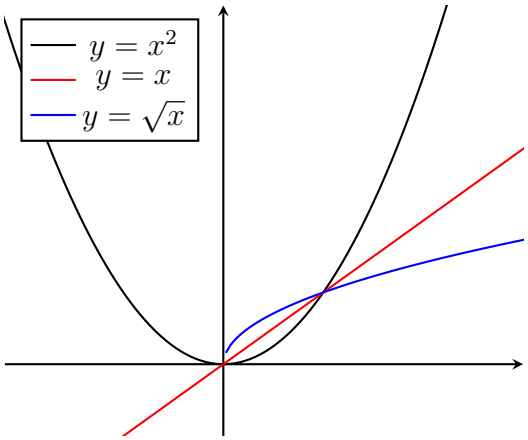
While it may seem like LASSO and ridge regression are similar, LASSO is way more popular these days. What is special about $q = 1$? First, if $q < 1$, the objective function is not convex. Recall that a function $f : \mathbb{R}^p \longrightarrow \mathbb{R}$ is convex if

$$f(w\boldsymbol{x_1} + (1-w)\boldsymbol{x_2}) \leq wf(\boldsymbol{x_1}) + (1-w)f(\boldsymbol{x_2})$$

for all $\boldsymbol{x_1}, \boldsymbol{x_2} \in \mathbb{R}^p$, $w \in [0, 1]$.



Convex functions are much easier to minimize, both theoretically and numerically.



Moreover, when $q = 1$, the coefficients will enjoy the property of *sparsity*. That is, for a

LASSO estimator with proper penalization λ, $\hat{\boldsymbol{\beta}}^{LASSO}$,

$$\hat{\boldsymbol{\beta}}^{LASSO} = \begin{pmatrix} \hat{\beta}_1^{LASSO} \\ \hat{\beta}_2^{LASSO} \\ \vdots \\ \hat{\beta}_p^{LASSO} \end{pmatrix},$$

most of the coefficients $\hat{\beta}_1^{LASSO}, \hat{\beta}_2^{LASSO}, \ldots, \hat{\beta}_p^{LASSO}$ will be zero, and only a few coefficients will be non-zero. Sparsity is useful for two reasons:

1. reduce the number of coefficients needed to be estimated, and
2. the model will be easier to interpret.

The sparsity of L_1 has something to do with the derivative with $||\boldsymbol{\beta}||_1$.

Quiz 1. *What is a "circle" for $q = 1$ and $q = 2$ respectively?*

Quiz 2. *Draw the partial derivatives $\frac{\partial ||\boldsymbol{\beta}||_1}{\partial \beta_1}$ and $\frac{\partial ||\boldsymbol{\beta}||_2}{\partial \beta_1}$ on the same graph.*

3 A Special Case with Closed Form Solution

Unlike the OLS, LASSO estimator does not have a closed form. We generally need numerical optimizations to solve $\boldsymbol{\beta}^{LASSO}$. However, just like what we learned PCA, we will look at a simple, special case of LASSO to help us under stand LASSO.

Example 2 (dummies with no intercept)**.** *$x_{i1}, x_{i2}, \ldots, x_{ip}$ are dummies, and $\sum_{j=1}^{p} x_{ij} = 1$. You can think of x_i as a group membership*

$$x_{ij} = \begin{cases} 1 & \text{if } i \text{ belongs to group } j \\ 0 & \text{o.w.} \end{cases}.$$

Recall that for OLS, $\hat{\beta}_j$ is the group average of y among $x_{ij} = 1$. The objective function

is now

$$L(\beta) = \sum_{i=1}^{n}(y_i - \boldsymbol{x_i'\beta})^2 + \lambda \sum_{j=1}^{p} |\beta_j|.$$

For each i, only one of x_{ij}s is 1, so

$$L(\beta) = \sum_{i:x_{i1}=1} (y_i - \boldsymbol{x_i'\beta})^2 + \sum_{i:x_{i2}=1} (y_i - \boldsymbol{x_i'\beta})^2 + \cdots + \sum_{i:x_{ip}=1} (y_i - \boldsymbol{x_i'\beta})^2 + \lambda \sum_{j=1}^{p} |\beta_j|.$$

For $\{i : x_{ij} = 1\}$, $\boldsymbol{x_i'\beta} = \beta_j$, so

$$\begin{aligned} L(\beta) = &\sum_{i:x_{i1}=1} (y_i - \beta_1)^2 + \lambda|\beta_1| + \cdots \sum_{i:x_{ip}=1} (y_i - \beta_p)^2 + \lambda|\beta_p| \\ =&L_1(\beta_1) + L_2(\beta_2) + \cdots + L_p(\beta_p). \end{aligned}$$

It now suffices to solve $L_1(\beta_1), L_2(\beta_2), \ldots, L_p(\beta_p)$ separately. Also, these are basically the same minimization problem.

Before we proceed, define

$$n_j = \sum_{i=1}^{n} x_{ij} = \textit{number of observations in group } j.$$

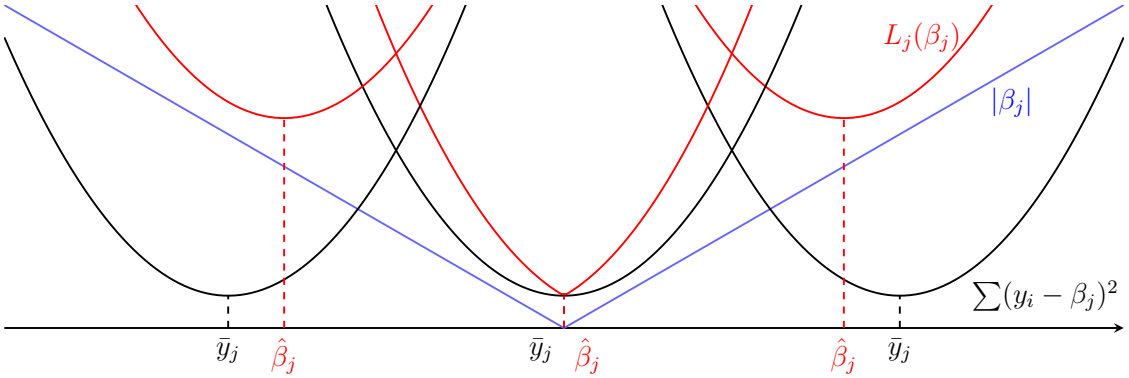
$$\bar{y}_j = \frac{1}{n_j} \sum_{i:x_{ij}=1} y_i = \textit{group average of group } j.$$

Notice that

$$\begin{aligned} \sum_{i:x_{ij}=1} (y_i - \beta_j)^2 &= \sum_{i:x_{ij}=1} \beta_j^2 - 2\beta_j y_i + y_i^2 \\ &= n_j\beta_j^2 - 2n_j\beta_j\bar{y}_j + \ldots \\ &= n_j(\beta_j - \bar{y}_j)^2 + c_j, \end{aligned}$$

for some constant c_j, so $\sum(y_i - \beta_j)^2$ is just a parabola in β_j. Below is a graph of three objective functions (in red) corresponding to three different $\bar{y}_j$. The black and blue lines

are $n_j(\beta_j - \bar{y}_j)^2 + c_j$ *and* $\lambda\beta_j$ *respectively.*



The function $L_j(\beta_j)$ *is not differentiable at* $\beta_j = 0$ *since the absolute function* $|\beta_j|$ *is not differentiable at* $\beta_j = 0$. *We can work this around by dividing the parameter space into*

$$\beta_j < 0, \quad \beta_j = 0, \quad \beta_j > 0$$

and compare the optimum within each partition to find the global maximum.

Finding maximum within $\{\beta_j = 0\}$ *is trivial since it contains only one point.*

$$L_j(\beta_j = 0) = n_j\bar{y}_j^2 + c_j.$$

For $\beta_j > 0$:

$$L_j(\beta_j) = \sum_{i:x_{ij}=1} (y_i - \beta_j)^2 + \lambda\beta_j$$

$$\begin{aligned}\frac{\partial L_j}{\partial \beta_j} &= -\sum_{i:x_{ij}=1} 2(y_i - \beta_j) + \lambda \\ &= -2n_j\bar{y}_j + 2n_j\beta_j + \lambda \Rightarrow \hat{\beta}_j = \bar{y}_j - \frac{\lambda}{2n_j}\end{aligned}$$

So $L_j(\beta_j)$ *attains minimum, within* $\beta_j > 0$ *at* $\beta_j = \bar{y}_j - \frac{\lambda}{2n_j}$ *if* $\bar{y}_j - \frac{\lambda}{2n_j} > 0$. *If* $\bar{y}_j - \frac{\lambda}{2n_j} \leq 0$, *then the minimum happens at boundary. The case for* $\beta_j < 0$ *is similar, which attains*

minimum if $\bar{y}_j - \frac{\lambda}{2n_j} < 0$, *otherwise the minimum happens at boundary.*

To sum up, the LASSO estimator is

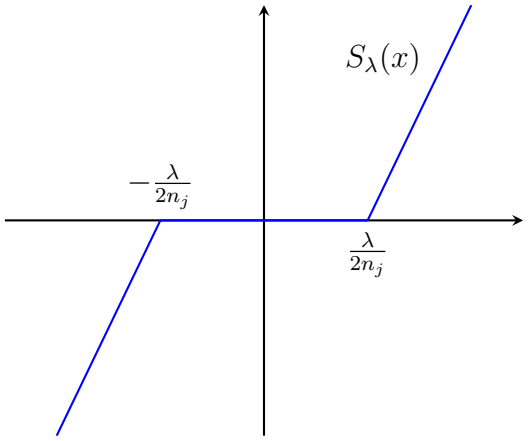
$$\widehat{\beta}_j^L = \begin{cases} \bar{y}_j - \frac{\lambda}{2n_j} & \text{if } y_j > \frac{\lambda}{2n_j} \\ \bar{y}_j + \frac{\lambda}{2n_j} & \text{if } y_j < -\frac{\lambda}{2n_j} \\ 0 & \text{if } |y_j| \leq \frac{\lambda}{2n_j} \end{cases}$$

Compare with $\widehat{\beta}_j^{OLS} = \bar{y}_j$, *we can see that the LASSO estimator is essentially the OLS estimator selected and shrunk by the soft-thresholding operator* S_λ, *that is*

$$\widehat{\beta}_j^L = S_\lambda(\hat{\beta}_j^{OLS}),$$

where

$$S_\lambda(x) = sgn(x)(|x| - \frac{\lambda}{2n_j})_+, \quad sgn(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}, \quad (z)_+ = \begin{cases} z & \text{if } z > 0 \\ 0 & \text{if } z \leq 0 \end{cases}$$



That is, only groups with large enough averages are selected (rest are set to 0); and these selected are still shrunk.

4 Solving LASSO with Gradient Descent

The LASSO regression

$$\arg\min_{\alpha,\beta} \sum_{i=1}^{n}(y_i - \alpha - x_i'\beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

generally has no closed form solutions. However, it could be solved numerically by the *coordinate descent method*. Before we introduce the actual method that can be used for solving LASSO, let us first discuss *gradient descent*, which is one of the most common optimization method used in machine learning.

Gradient: Let $f(x) : \mathbb{R}^p \longrightarrow \mathbb{R}$ be a differentiable function. Its gradient ∇f is defined as

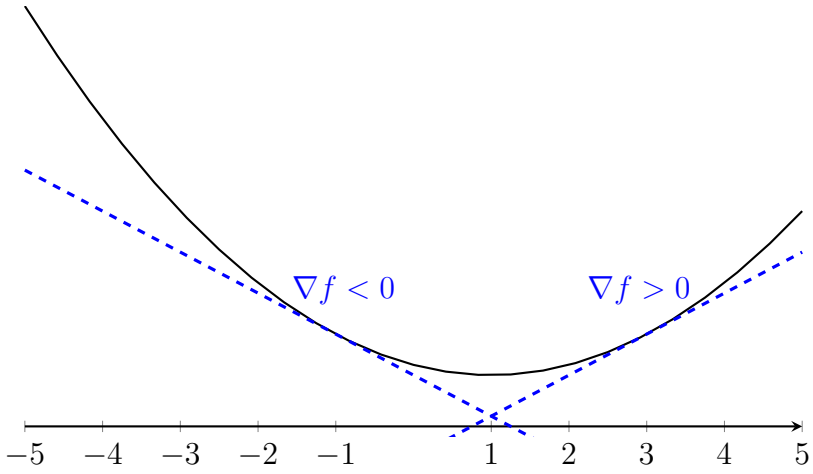
$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \dots \\ \frac{\partial f}{\partial x_p} \end{pmatrix}.$$

Gradient signifies the direction of the steepest increment.

Example 3.

$$f(x) = (x-1)^2 + 5$$

$$\nabla f = \frac{\partial f}{\partial x} = 2x - 2$$



$\nabla f > 0$ *moves right*

$\nabla f < 0$ *moves left*

We could iteratively move in the opposite direction of ∇f to find the minimum of f:

$$x^{(k+1)} = x^{(k)} - \gamma^{(k)} \nabla f(x^{(k)}),$$

where $\gamma^{(k)}$ is the learning rate and $x^{(0)}$ is the initial point chosen by the researcher.

Update direction:

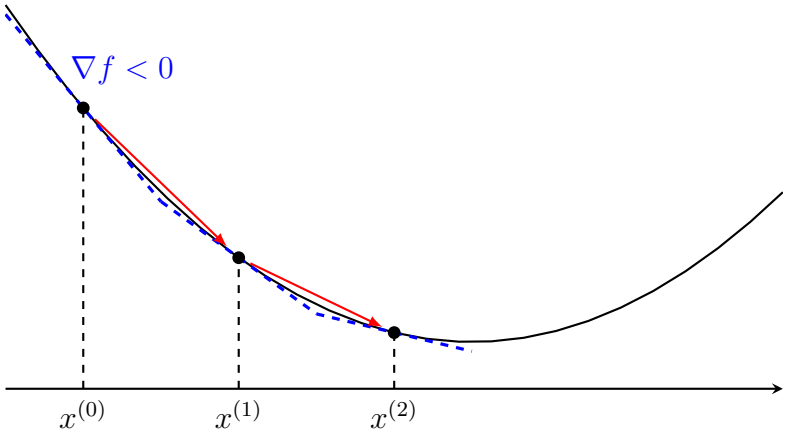
$\nabla f > 0$ moves left

$\nabla f < 0$ moves right

Step size:

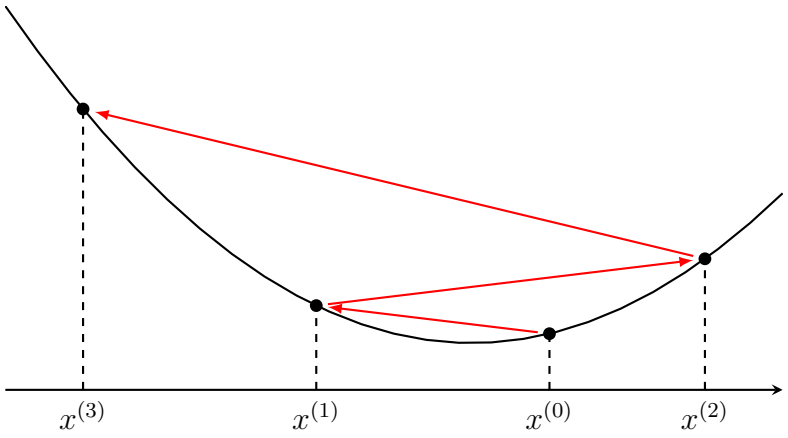
proportional to the steepness ∇f

and learning rate $\gamma^{(k)}$

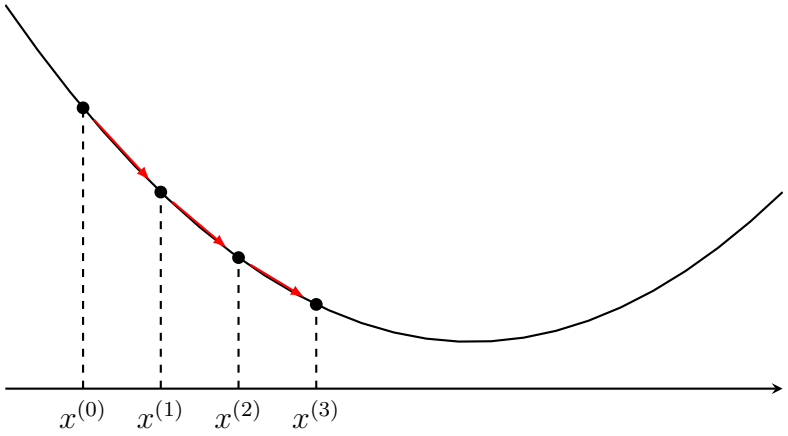


Choosing the learning rate γ

$\gamma^{(k)}$ **too large:** may not converge



$\gamma^{(k)}$ **too small:** Converge too slow



One way to select the learning rate is the *Newton's* Method.

Newton's Method:

$$\gamma^{(k)} = \left(\nabla^2 f(x^{(k)})\right)^{-1} \quad \text{(Inverse Hessian)}$$
$$= \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_p} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_p \partial x_1} & & \frac{\partial^2 f}{\partial x_p^2} \end{pmatrix}^{-1} \Bigg|_{x=x^{(k)}}$$

Recall that Hessian the change rate the of gradient. Therefore, the Newton's Method selects a smaller step size (more conservative) when the hessian is large.

The *Line Search* method might seem more intuitive:

Line search At iteration k, choose γ_k that optimizes

$$f(x_k - \gamma_k \nabla f(x_k)).$$

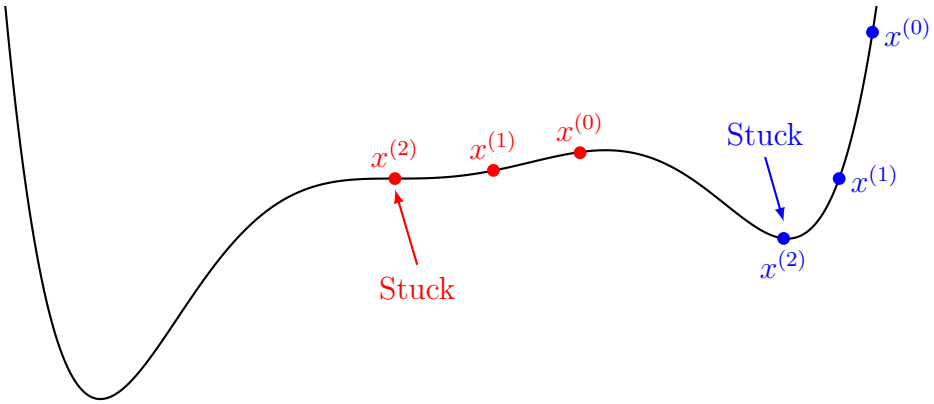
Just like the penalty term in LASSO, learning rate is also a tuning parameter. But we should distinguish the two types of tuning parameters:

1. tuning parameters in numerical optimization, e.g., learning rate in gradient descent, and
2. tuning parameters in model specification, e.g., penalty term in penalized regressions and depth in decision trees.

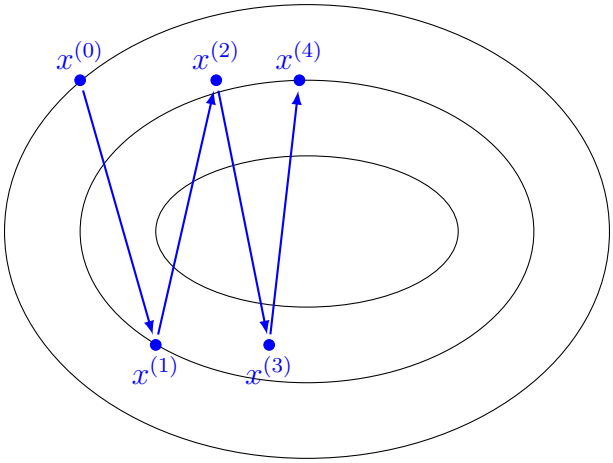
While selecting the tuning parameters (and the algorithm) for numerical optimization can be case-dependent, selecting the tuning parameters for model specifications usually can be done in a systematic way, a topic we will move on to in the next section.

Potential Challenges with 'naive' gradient descent

Non-scrict convex



Zig-zagging



In addition to the Newton's Method, there are many other extensions of gradient descent:

- accelerated gradient descent,
- Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm,
- stochastic gradient descent (a.k.a. SGD), and

L-BFGS and SGD are both widely-used for machine methods, including support vector machines, regression trees, and neuron networks. Although I do not really cover numerical optimization in this course, it is important to know that lots of the recent progress in machine learning are driven by advancement in numerical optimization, that is, becoming better at computing stuff.