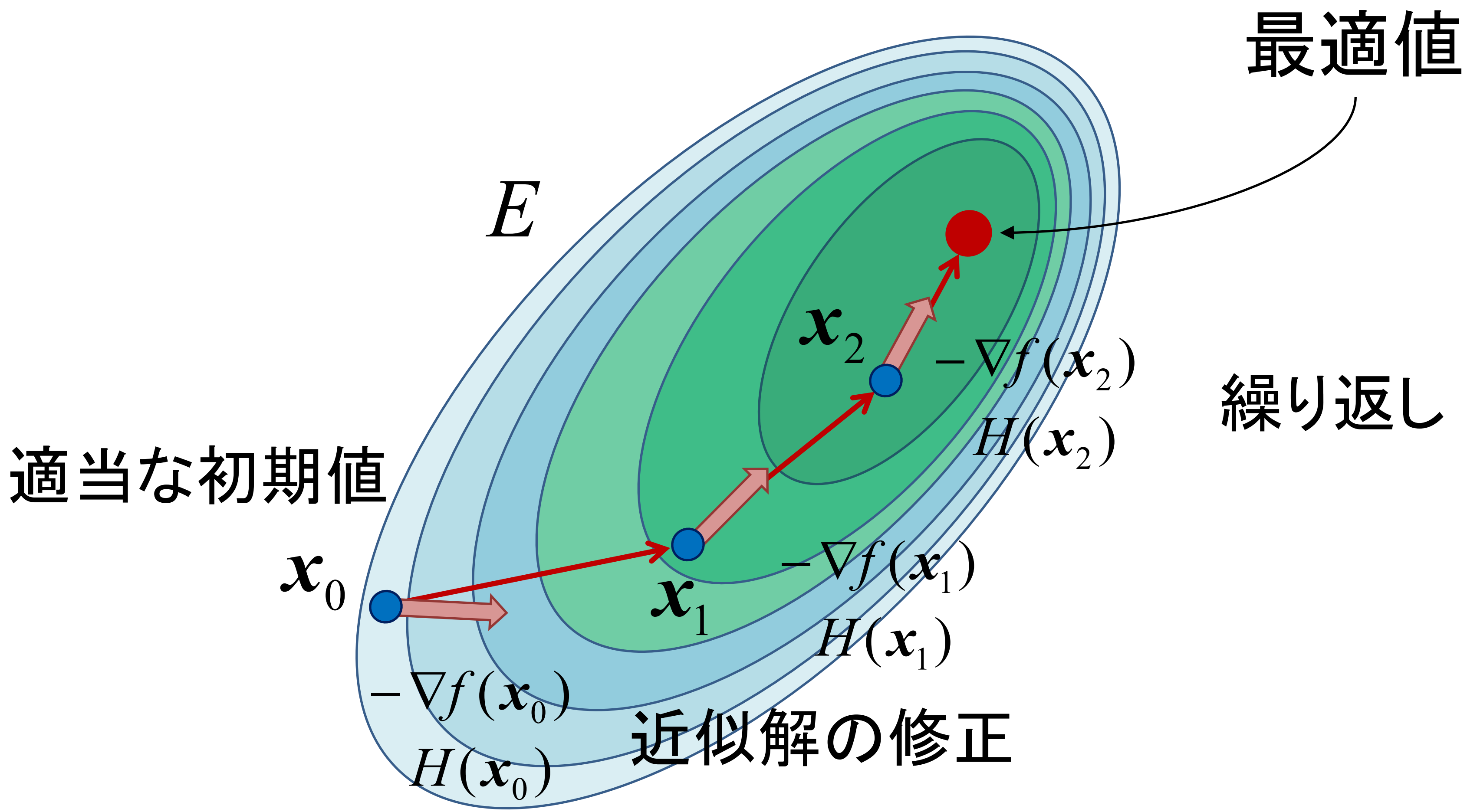
勾配法

最急降下法
共役勾配法
ニュートン法
正則化（L1,L2）

§ 勾配法

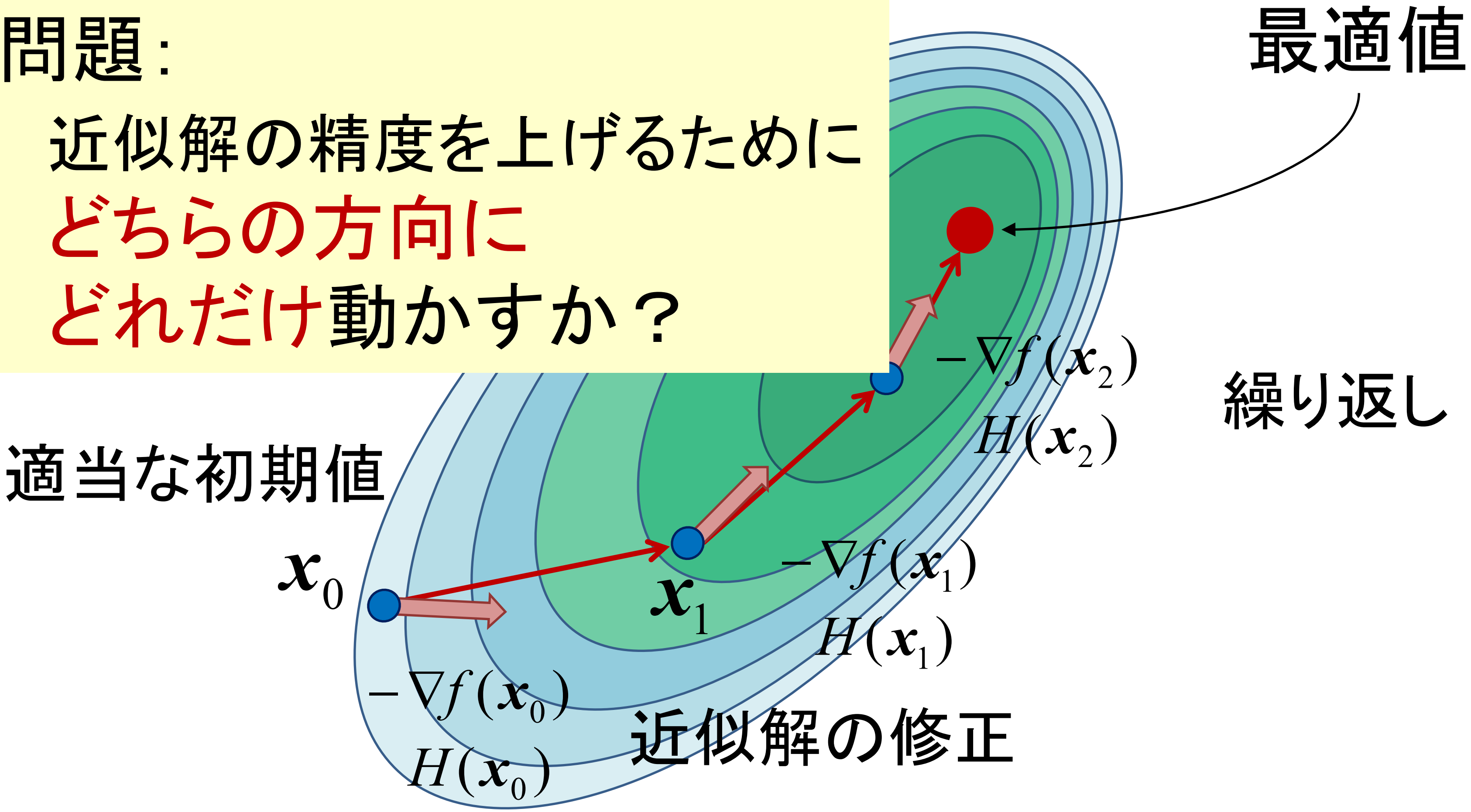


勾配(，ヘシアン)(Eのx_0における1次(，2次)の微分)

勾配法

問題：

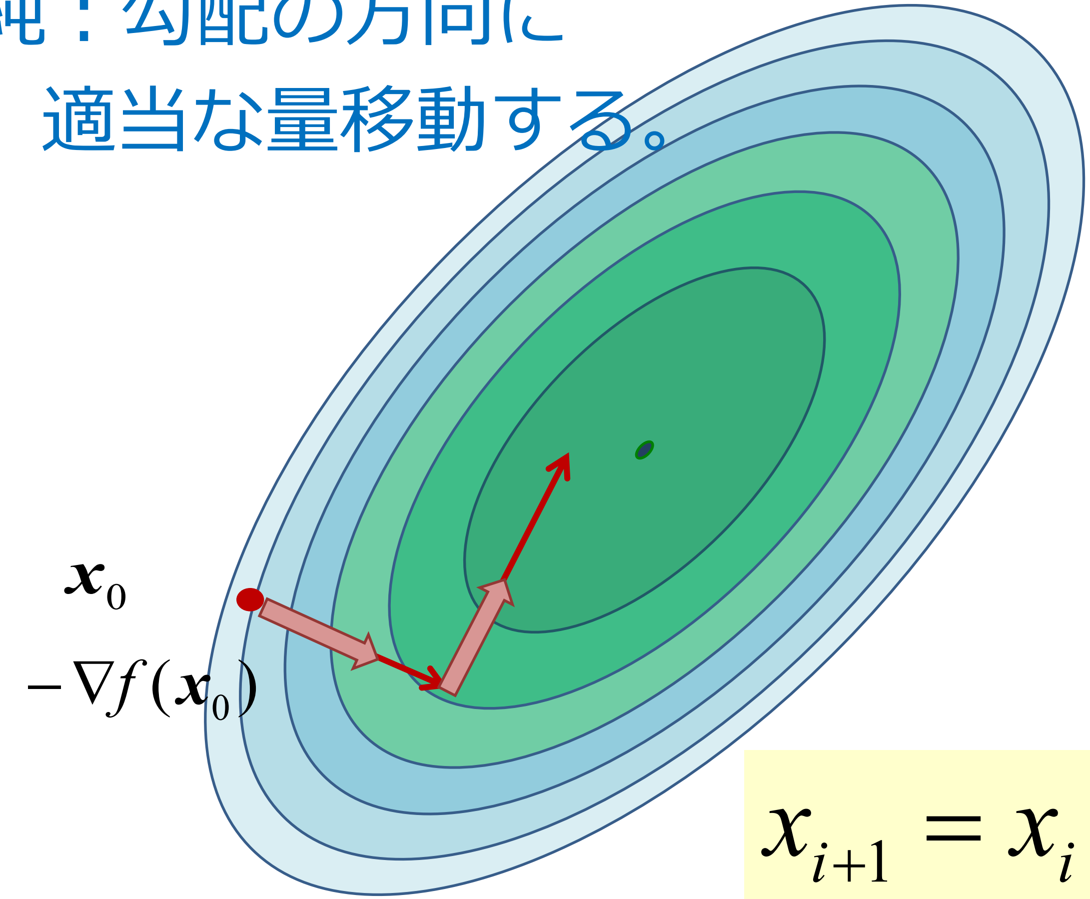
近似解の精度を上げるために
どちらの方向に
どれだけ動かすか？



勾配（, ヘシアン）（Eのx_0における1次（, 2次）の微分）

§ 最急降下法

- 最急降下法：
  - ・最も単純：勾配の方向に
    適当な量移動する。



$$x_{i+1} = x_i - \alpha \nabla f(\boldsymbol{x}_i)$$

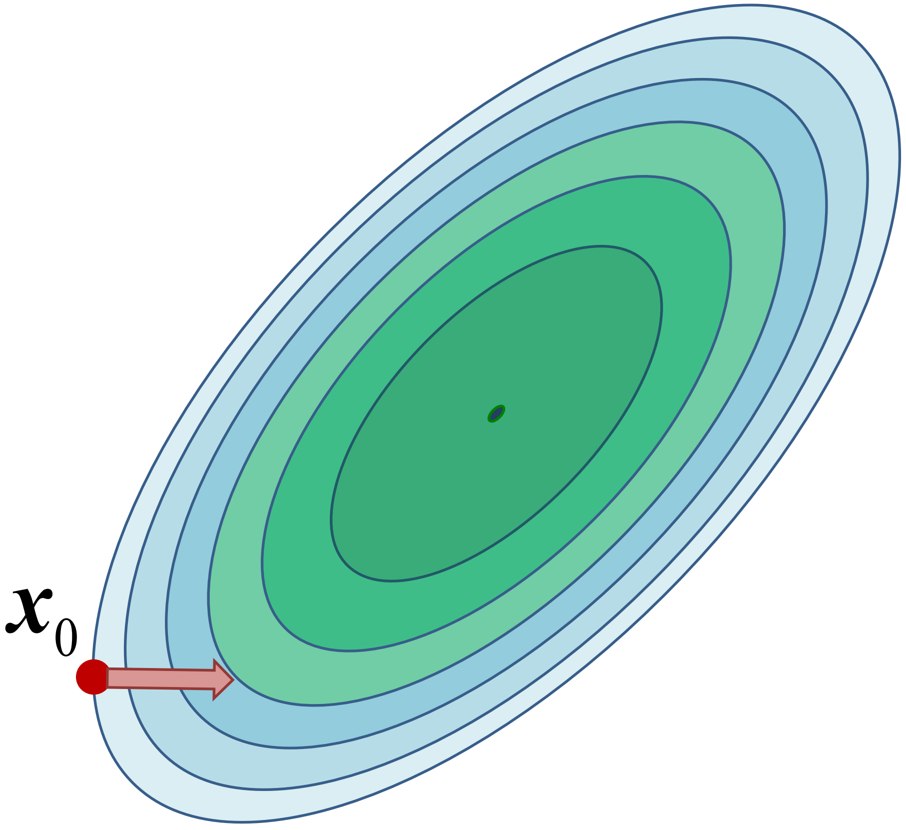
最急降下法

i=0，X_0を決める，
収束するまで繰り返し {

$$x_{i+1} = x_i - \alpha \nabla f(\boldsymbol{x}_i)$$

i = i+1

}



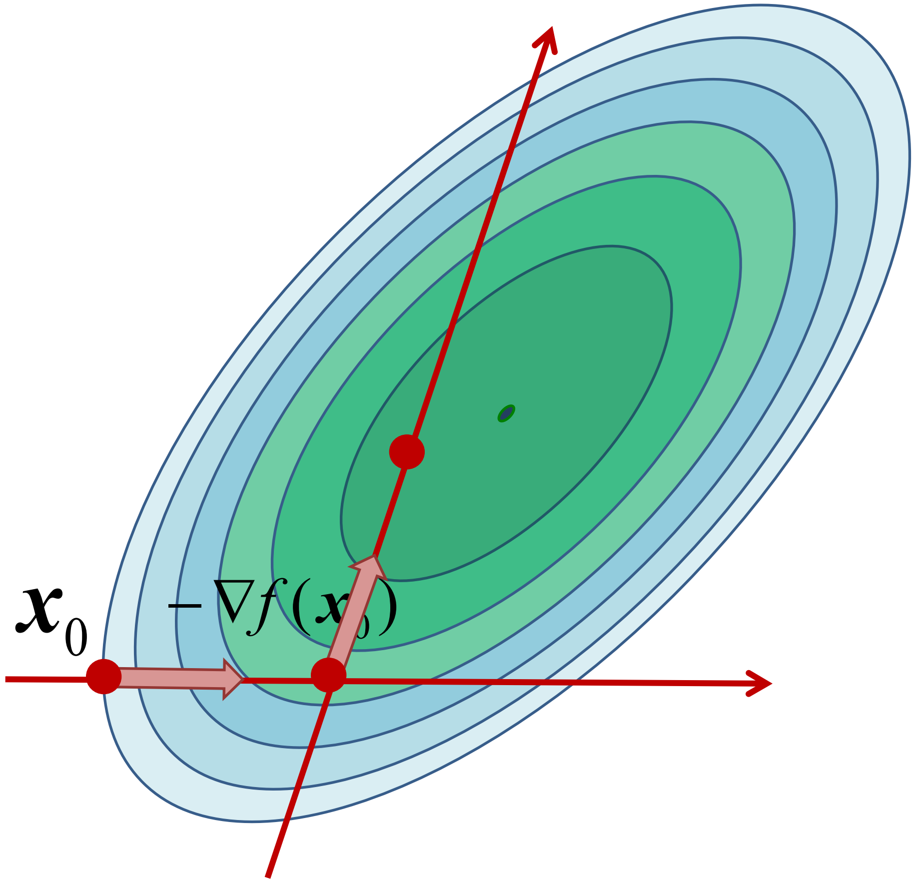
最急降下法

i=0，x_0を決める，
収束するまで繰り返し {

$$x_{i+1} = x_i - \alpha \nabla f(\boldsymbol{x}_i)$$

i = i+1

}



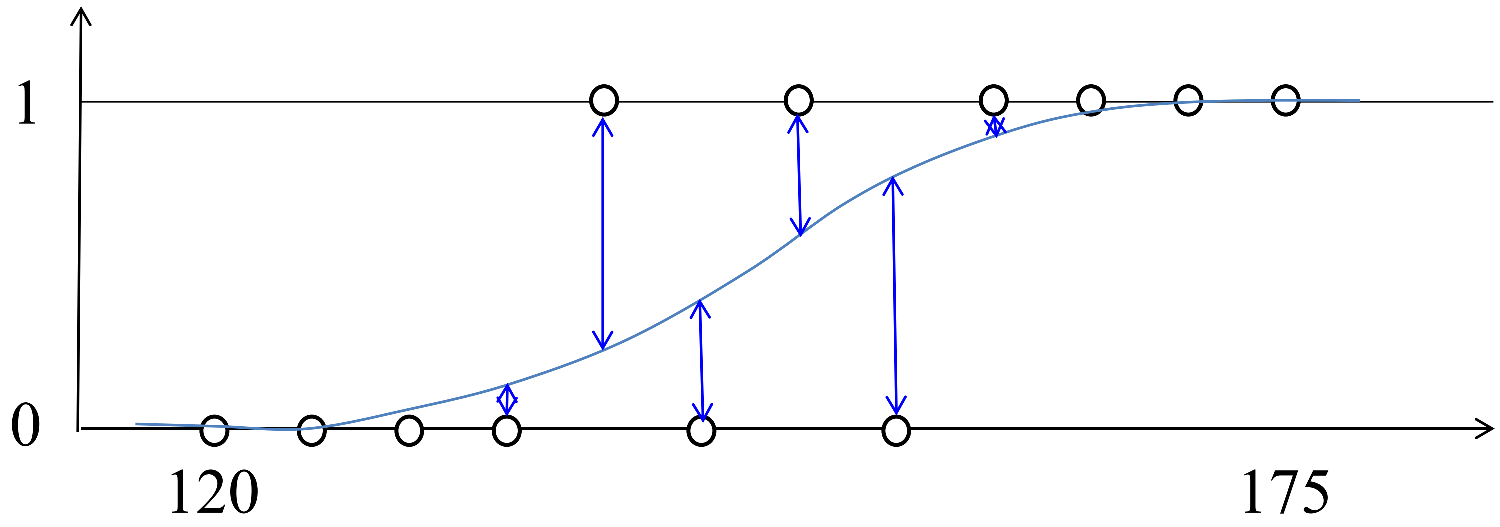
αの決め方は様々

・-∇f(x)　の最適値まで移動
（解析的に求める，ラインサーチで求める）
・決まった量移動
・過去の勾配の大きさの履歴に応じて移動量を決定

例題

下記の学習データに対し，ロジスティック関数（シグモイド関数）を当てはめる。

（1.20，0）（1.25，0）（1.30，0）（1.35，0）
（1.40，1）（1.45，0）（1.50，1）（1.55，0）
（1.60，1）（1.65，1）（1.70，1）（1.75，1）



$$e_n{}^2 = \frac{1}{2}(\tilde{y}_n - y_n)^2 = \frac{1}{2}\left(\frac{1}{1+\exp(-\boldsymbol{w}^T\boldsymbol{x})} - y_n\right)^2$$

$$\frac{\partial e_n{}^2}{\partial \boldsymbol{w}} = (\tilde{y}_n - y_n)\tilde{y}_n(1-\tilde{y}_n)\boldsymbol{x}$$

$$\frac{\partial E}{\partial \boldsymbol{w}} = \sum_{n=1}^{N}(\tilde{y}_n - y_n)\tilde{y}_n(1-\tilde{y}_n)\boldsymbol{x}$$

- On-line 学習： サンプル毎にパラメタを更新

$$\boldsymbol{w}_{new} = \boldsymbol{w}_{prv} - \alpha\frac{\partial e_n{}^2}{\partial \boldsymbol{w}}$$

- バッチ学習 ：学習データ一括でパラメタを更新

$$\boldsymbol{w}_{new} = \boldsymbol{w}_{prv} - \alpha\frac{\partial E}{\partial \boldsymbol{w}}$$

疑似コーディングの例

```
パラメタの初期値を w に与える。
収束するまで繰り返し {
        j=1 から順に，学習データの数 N まで繰り返し {
                grad = データ xj における傾き;          // 修正の方向
                学習率 α を決める;                       // 修正の量
                w = w - α  * grad ;                     // wの更新
        }
        収束していたら， wを解として終了;
}
```

プログラムの例 1

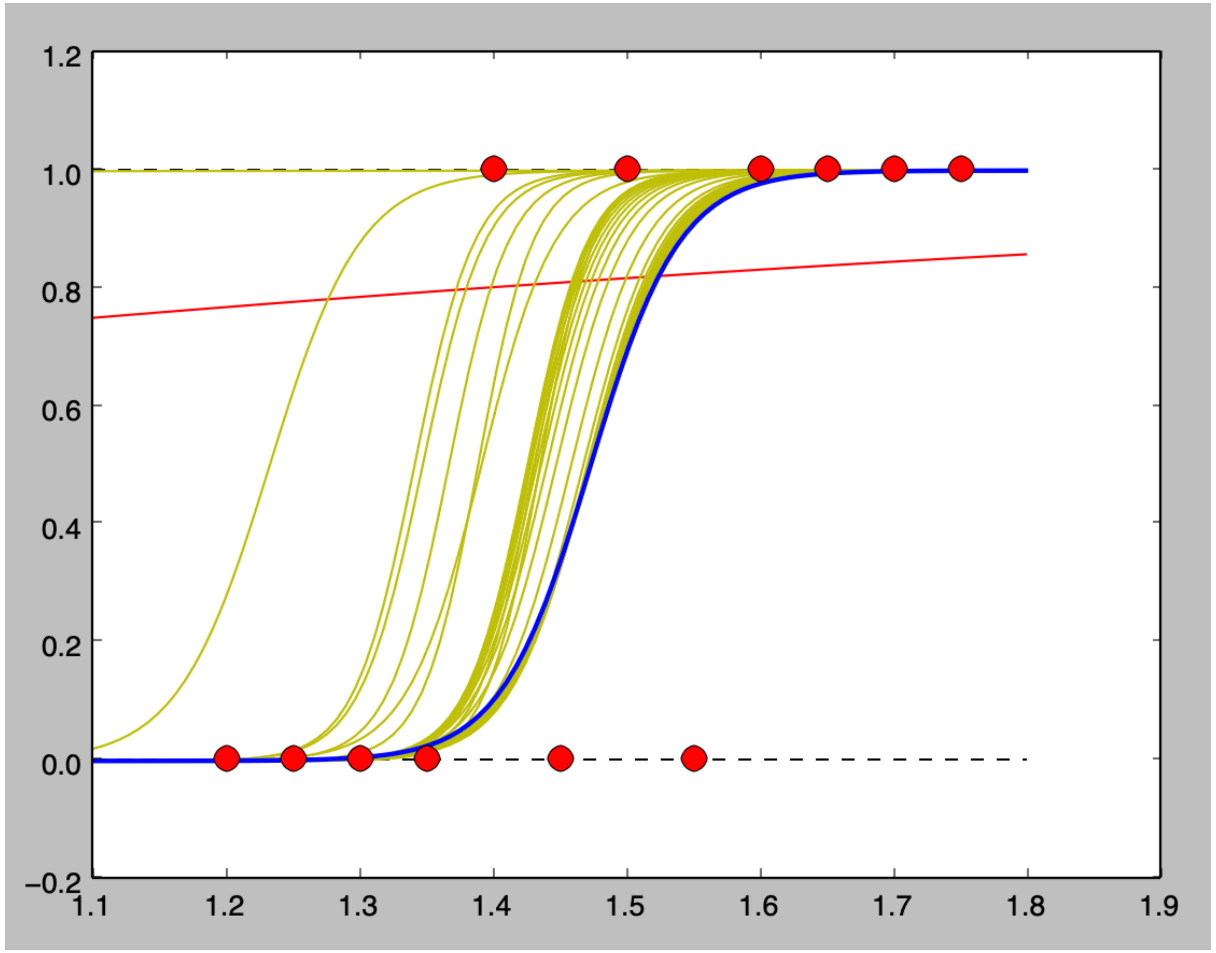
```
import numpy as np
from scipy import linalg as la
import matplotlib.pyplot as plt
import copy as cp

def sigm(z):
    return 1/(1+np.exp(-z))

trainingData =
np.array([[1.20,0],[1.25,0],[1.30,0],[1.35,0],[1.40,1],[1.45,0],[1.50,1
],[1.55,0],[1.60,1],[1.65,1],[1.70,1],[1.75,1]])

N,nd = trainingData.shape      # N : number of samples
                               # nd: dimension
```

```
true    = trainingData[:,1]        # true value
D = cp.copy(trainingData)
D[:,nd-1] = np.ones(N)             # Data Matrix

TH = 1.0e-7          # threshold for iteration
alpha = 5.0          # learning rate
alpDec = 0.9999      # parameter for alpha decay

nPlot = 100                              # plot resolution
pltXaxis = np.linspace(1.1,1.8,nPlot)# x-data for plot
xx = np.ones((nPlot,nd))                 # data for plot
xx[:,0] = pltXaxis                       # data for plot

w = np.array([1.0,0.0])      # initial model parameters
plt.plot(pltXaxis, sigm(xx.dot(w)),color="r",linewidth=1)
```

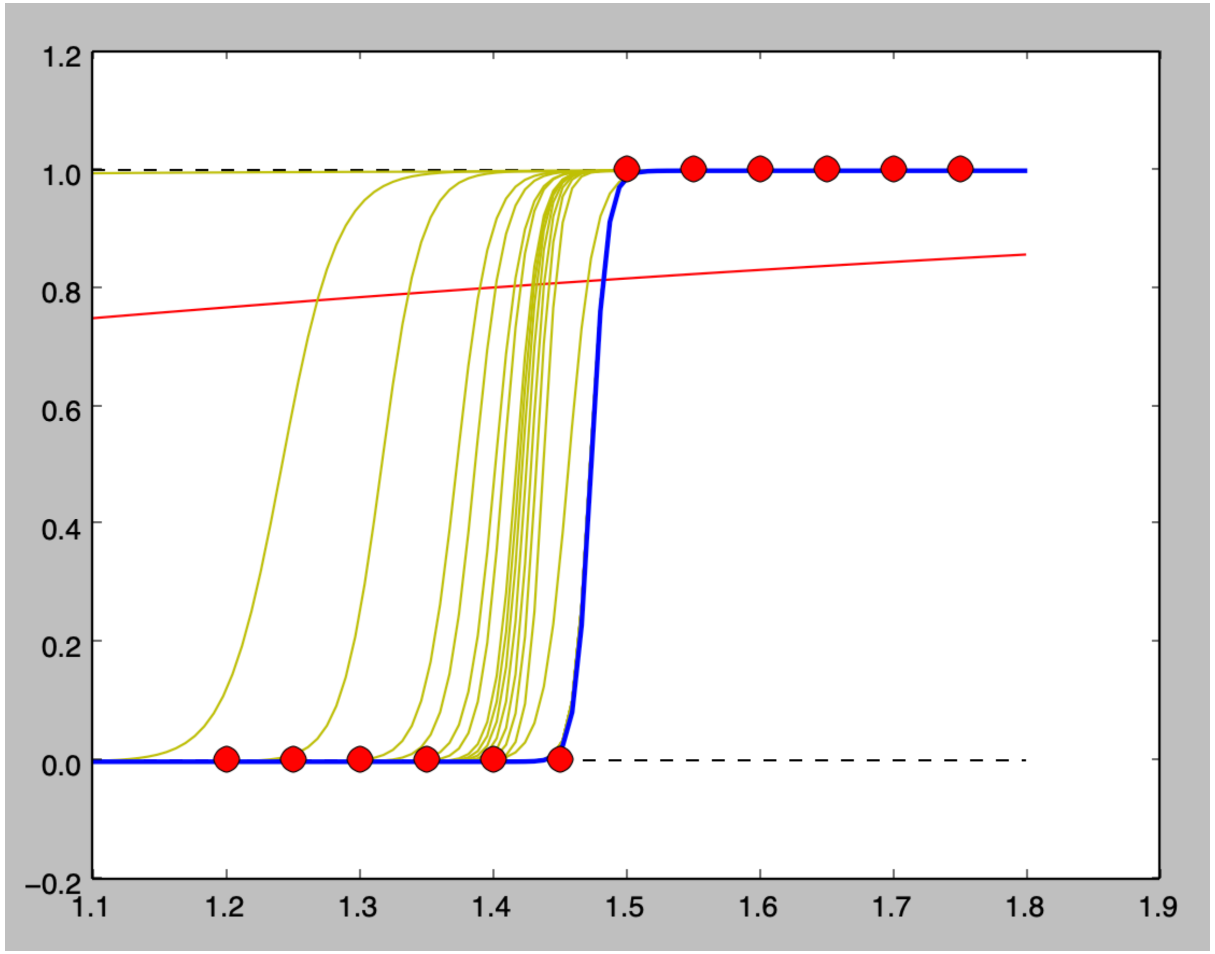
$$\frac{\partial {e_n}^2}{\partial \boldsymbol{w}} = (\tilde{y}_n - y_n)\tilde{y}_n(1 - \tilde{y}_n)\boldsymbol{x}$$

```
errPrv = 10.0
for i in range(100000):
    for j in range(N):
        zj = sigm(D[j,:].dot(w))
        grad = (zj-true[j])*zj*(1-zj)*D[j,:]
        w = w - alpha*grad
        alpha = alpha*alpDec

    estimate = sigm(D.dot(w))
    errNew = np.sqrt((true-estimate).dot((true-estimate)))
    if (abs(errNew-errPrv) < TH): break
    errPrv = errNew

    if (i%100 == 0):
        plt.plot(pltXaxis, sigm(xx.dot(w)),color="y")
        print i,w,errNew
```

```
plt.plot(pltXaxis, sigm(xx.dot(w)),color="b",linewidth=2)

plt.ylim(-0.2, 1.2)
plt.hlines([0, 1], 1.1, 1.8, linestyles="dashed")
plt.plot(trainingData[:,0],true,"ro",markersize=10)
plt.show()
```

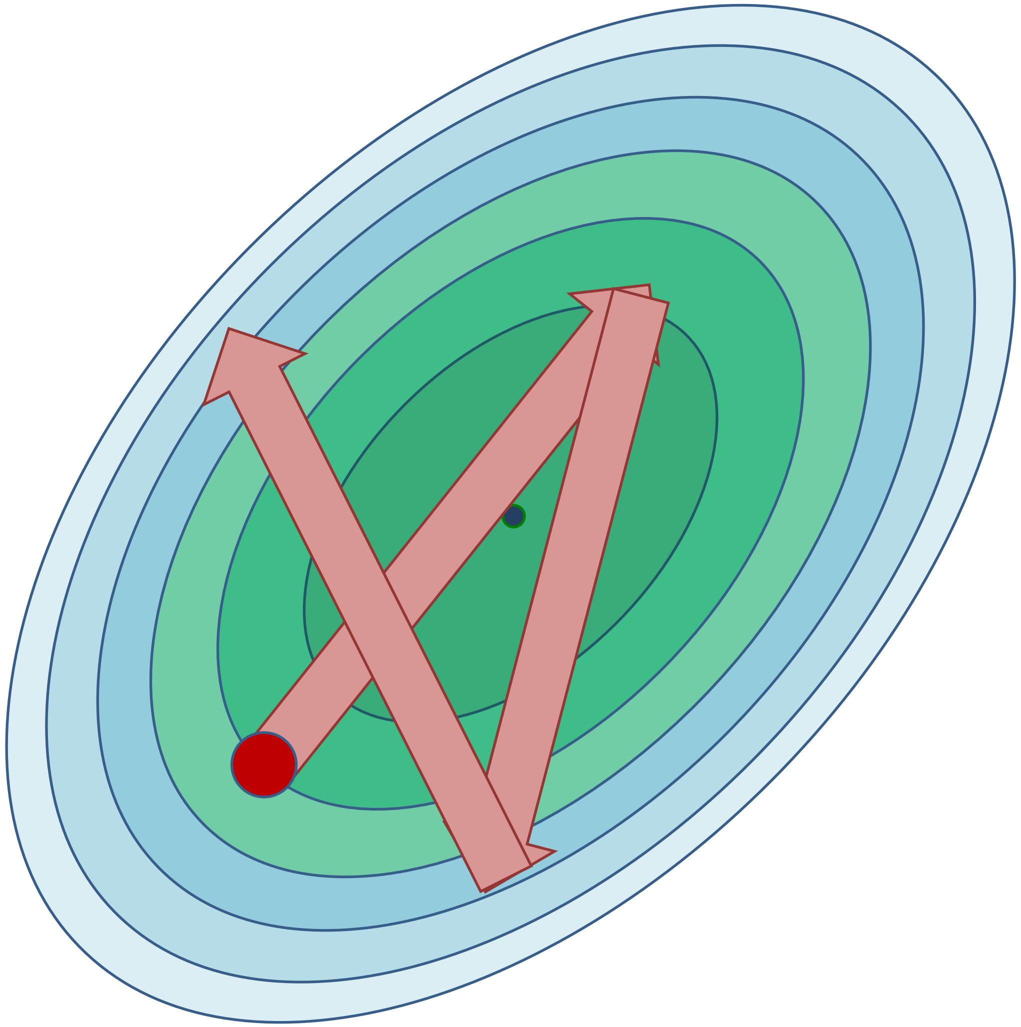


iteration:6901, error: 1.35668



iteration:6162, error: 0.17588

最急降下法の問題点 と 解決法1



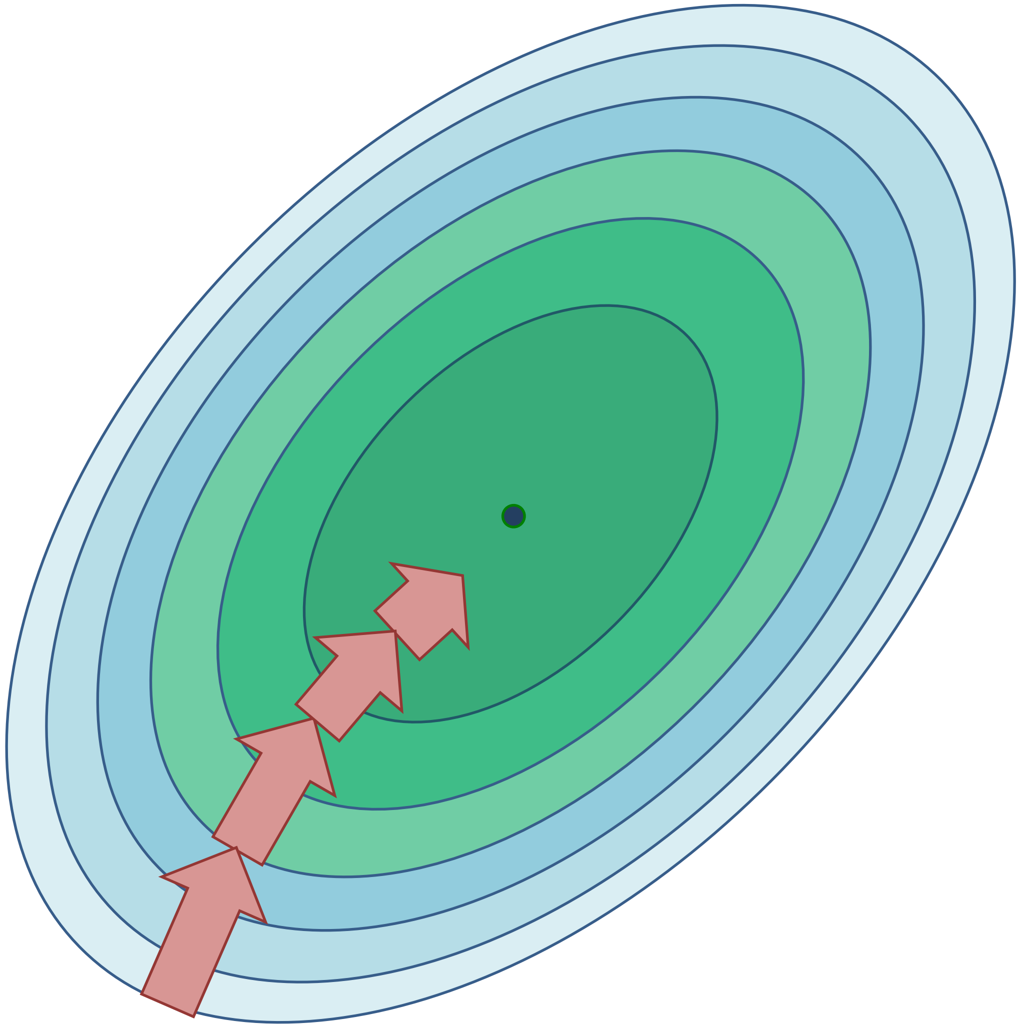
- 学習率αが大きいと，解の近傍で振動する。

⇒　移動方向に
傾きの指数平滑移動平均
を用いる

$$g_{new} = \beta g_{prv} + (1 - \beta)\nabla E$$

$$\boldsymbol{w}_{new} = \boldsymbol{w}_{prv} - \alpha g_{new}$$

最急降下法の問題点 と 解決法2



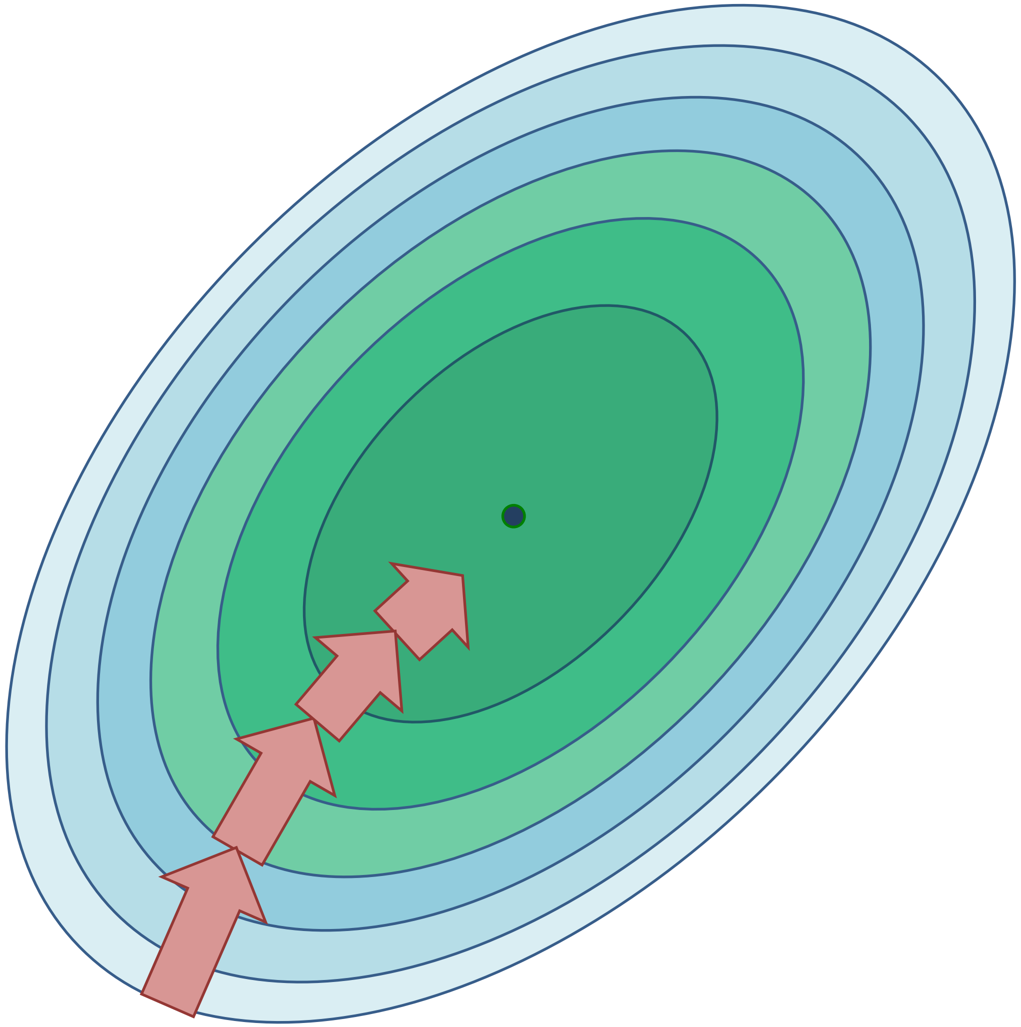
- 更新式を

$$\boldsymbol{w}_{new} = \boldsymbol{w}_{prv} - \frac{\eta}{c}\frac{\nabla E}{\|\nabla E\|}$$

とおいて考える

- 傾きが小さくなってきたら，解は近い
  - ⇒　更新量を小さくしたい
  - ⇒　c を大きくしたい
- 傾きが大きくなってきたら，解は遠い
  - ⇒　更新量を大きくしたい
  - ⇒　c を小さくしたい

最急降下法の問題点 と 解決法2



- 更新式を

$$\boldsymbol{w}_{new} = \boldsymbol{w}_{prv} - \frac{\eta\,\nabla E}{\sqrt{s_{new}}}$$

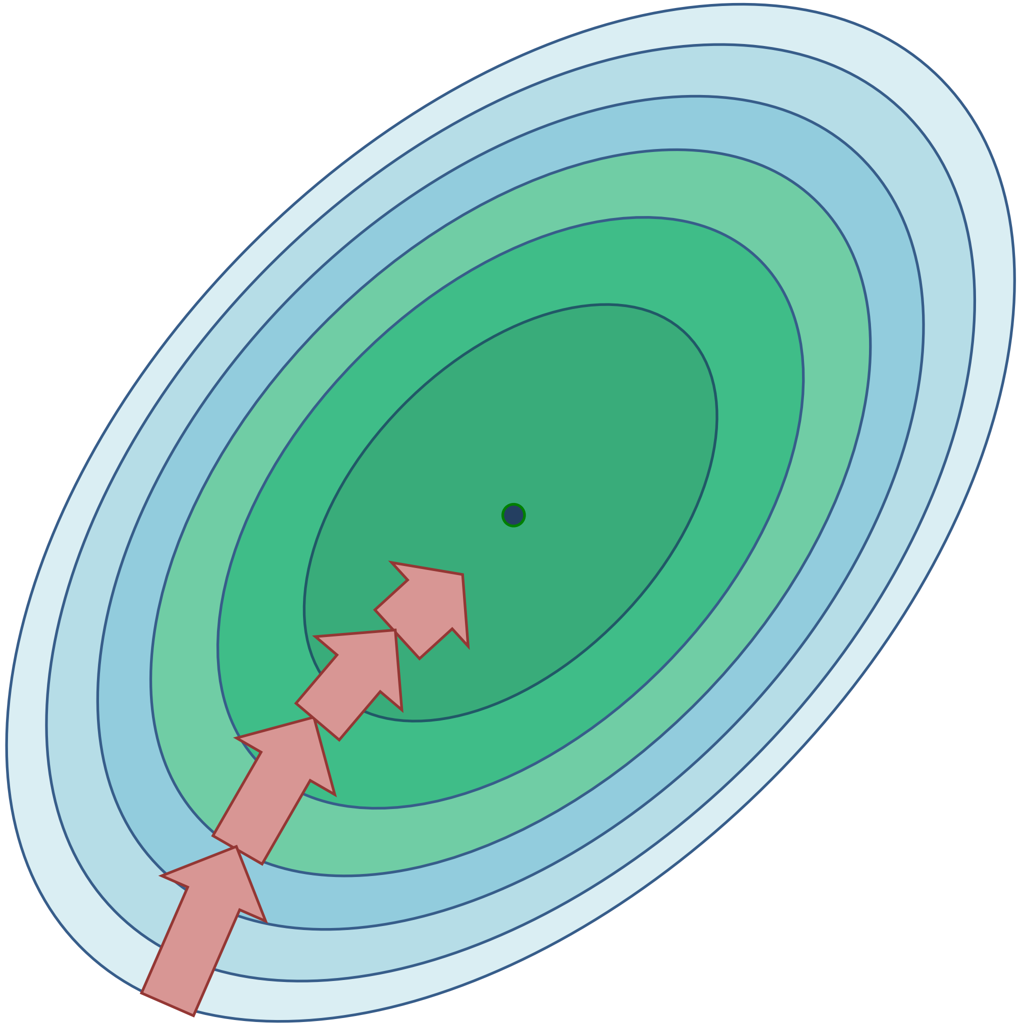
とおく。　ただし，s_{new}は傾きのノルムの二乗の指数平滑移動平均とする

$$s_{new} = \gamma s_{prv} + (1-\gamma)\|\nabla E\|^2$$

- 傾きが小さく(大きく)なってきたとき，傾きはその移動平均より小さい(大きい)

  $\Rightarrow$ $\sqrt{s_{new}}$で割ることは，
  前頁の更新式 の c を，
  大きく(小さく)することと等価

最急降下法の問題点 と 解決法3



- 解決法1,2の組み合わせ

$$g_{new} = \beta g_{prv} + (1-\beta)\nabla E$$

$$s_{new} = \gamma s_{prv} + (1-\gamma)\|\nabla E\|^2$$

$$\boldsymbol{w}_{new} = \boldsymbol{w}_{prv} - \eta \frac{g_{new}}{\sqrt{s_{new}}}$$

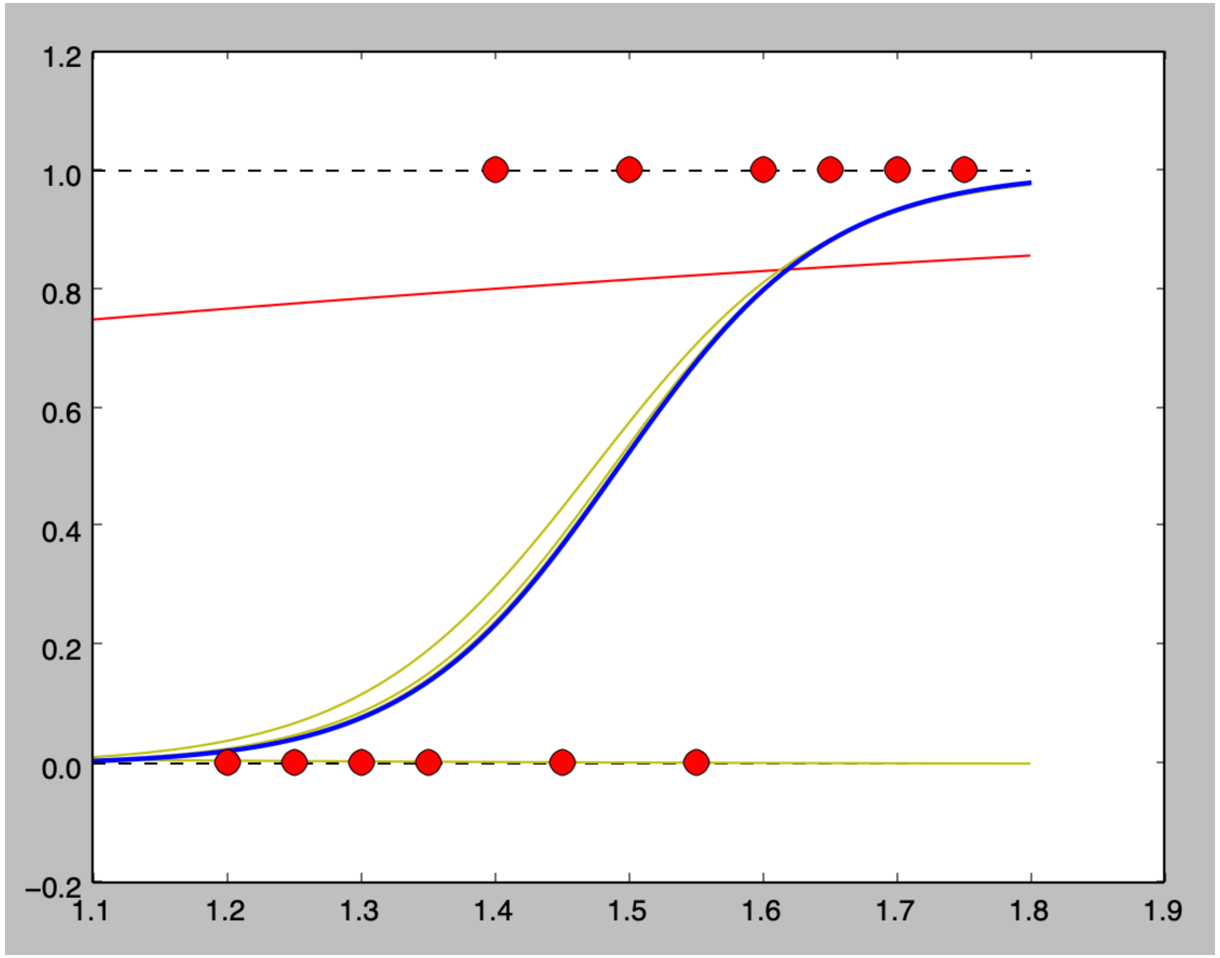
プログラムの例 2

```
import numpy as np
from scipy import linalg as la
import matplotlib.pyplot as plt
import copy as cp

def sigm(z):
    return 1/(1+np.exp(-z))

trainingData =
np.array([[1.20,0],[1.25,0],[1.30,0],[1.35,0],[1.40,1],[1.45,0],[1.50,1
],[1.55,0],[1.60,1],[1.65,1],[1.70,1],[1.75,1]])

N,nd = trainingData.shape      # N : number of samples
                               # nd: dimension
```

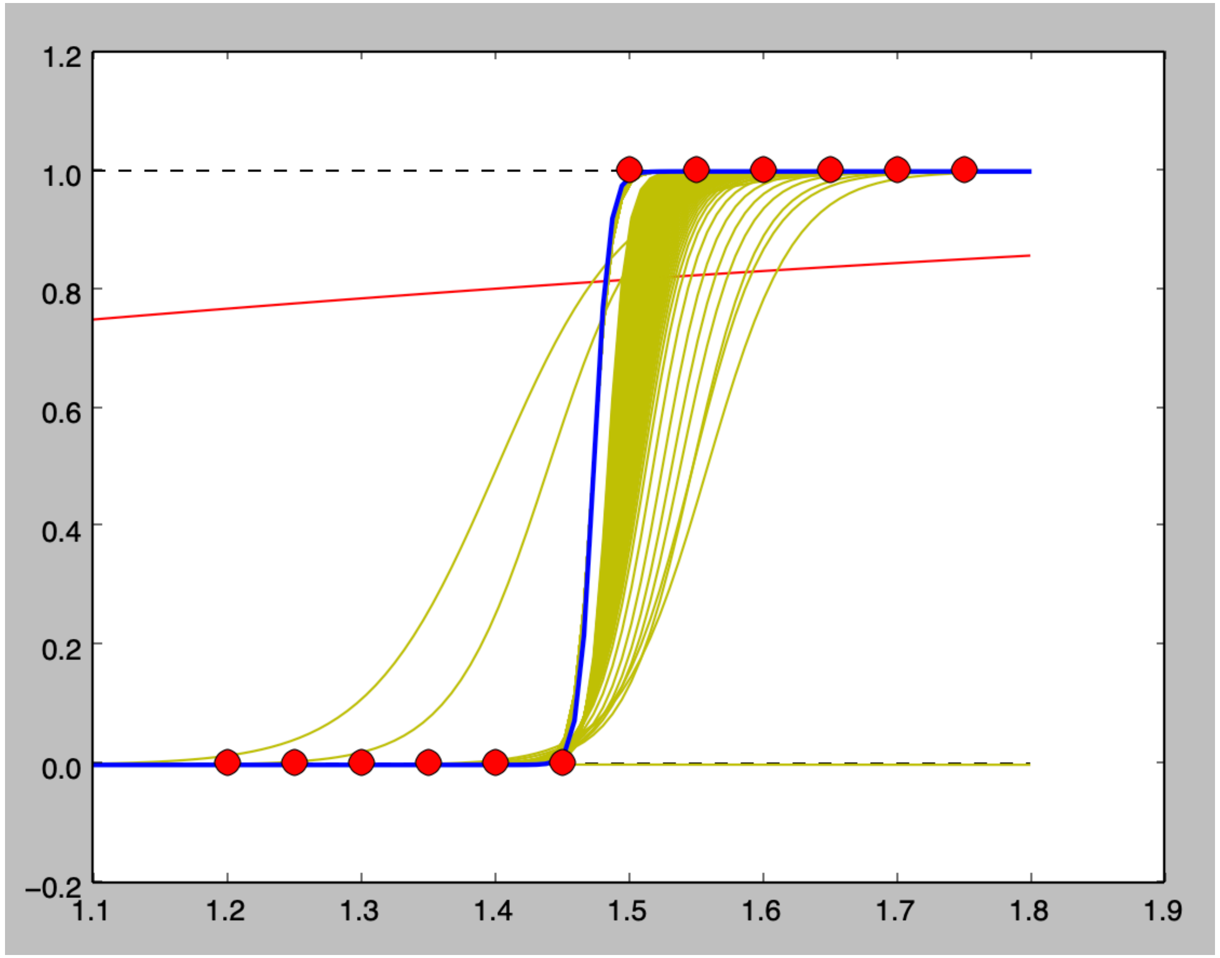
```
true    = trainingData[:,1]          # true value
D = cp.copy(trainingData)
D[:,nd-1] = np.ones(N)               # Data Matrix

TH = 1.0e-7           # threshold for iteration
eta = 1.0             # constant balancing update parameter
beta = 0.7            # weight for averaging v
gamma = 0.7           # weight for averaginf s
epsilon = 1.0e-4      # constant preventing zero divide

nPlot = 100                                  # plot resolution
pltXaxis = np.linspace(1.1,1.8,nPlot)# x-data for plot
xx = np.ones((nPlot,nd))                     # data for plot
xx[:,0] = pltXaxis                           # data for plot

w = np.array([1.0,0.0])     # initial model parameters
plt.plot(pltXaxis, sigm(xx.dot(w)),color="r",linewidth=1)
```

```python
errPrv = 10.0
vPrv = np.array([0.0,0.0])
sPrv = 0
for i in range(100000):
    for j in range(N):
        zj = sigm(D[j,:].dot(w))
        grad = (zj-true[j])*zj*(1-zj))*D[j,:]
        vNew = beta*vPrv + (1-beta)*grad
        sNew = gamma*sPrv + (1-gamma)*grad.dot(grad)
        w = w - eta*vNew/np.sqrt(sNew+epsilon)
        vPrv = vNew
        sPrv = sNew
    estimate = sigm(D.dot(w))
    errNew = np.sqrt((true-estimate).dot((true-estimate)))
    if (abs(errNew-errPrv) < TH): break
    errPrv = errNew
```

```
    if (i%100 == 0):
        plt.plot(pltXaxis, sigm(xx.dot(w)),color="y")
        print i,w,errNew
plt.plot(pltXaxis, sigm(xx.dot(w)),color="b",linewidth=2)

plt.ylim(-0.2, 1.2)
plt.hlines([0, 1], 1.1, 1.8, linestyles="dashed")
plt.plot(trainingData[:,0],true,"ro",markersize=10)
plt.show()
```



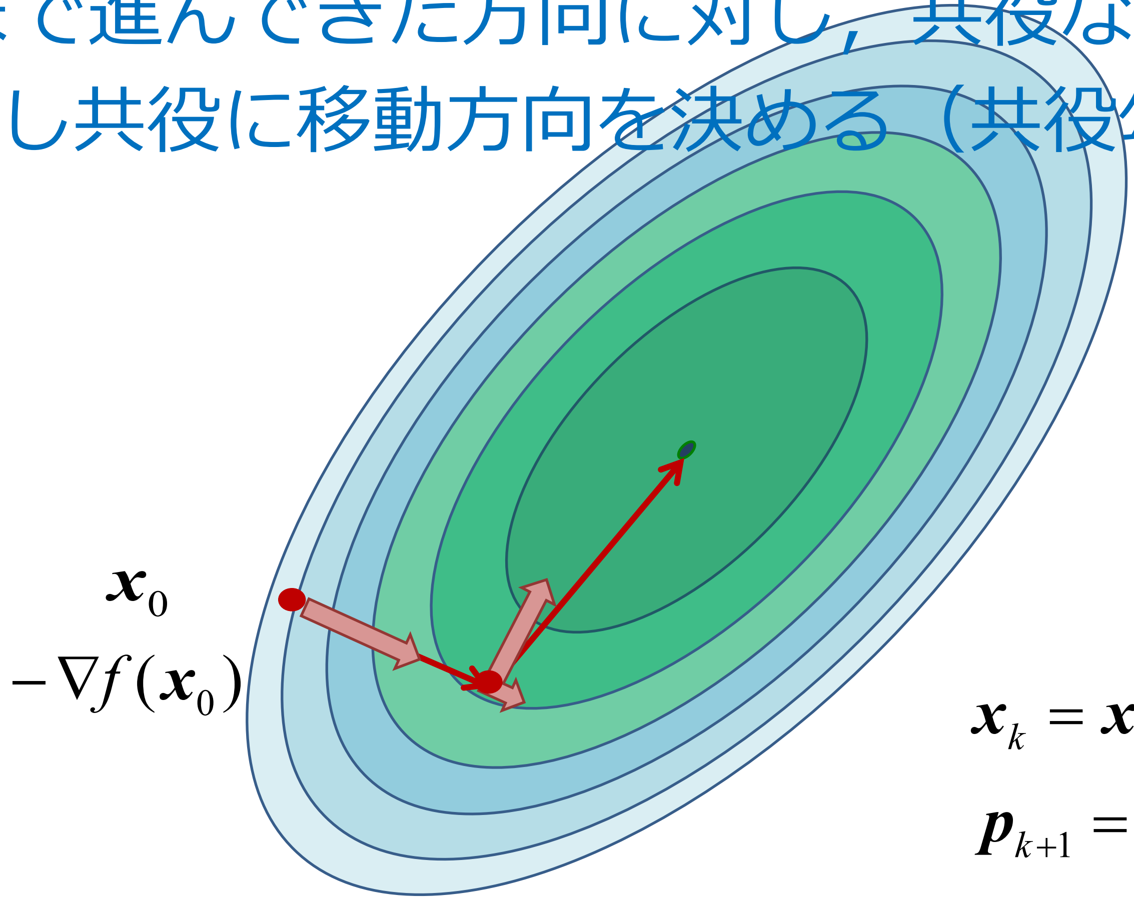
iteration:796, error: 1.22206



iteration:77548, error: 0.01725

§ 共役勾配法

- 最急降下法では，直線的に最適解に向かわない
  ⇒　今まで進んできた方向に対し，共役な方向に
  対し共役に移動方向を決める（共役勾配法）



$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + c_k \boldsymbol{p}_k$$

$$\boldsymbol{p}_{k+1} = -\nabla f(\boldsymbol{x}_k) + \alpha_k \boldsymbol{p}_k$$

共役勾配法

共役勾配法とは?

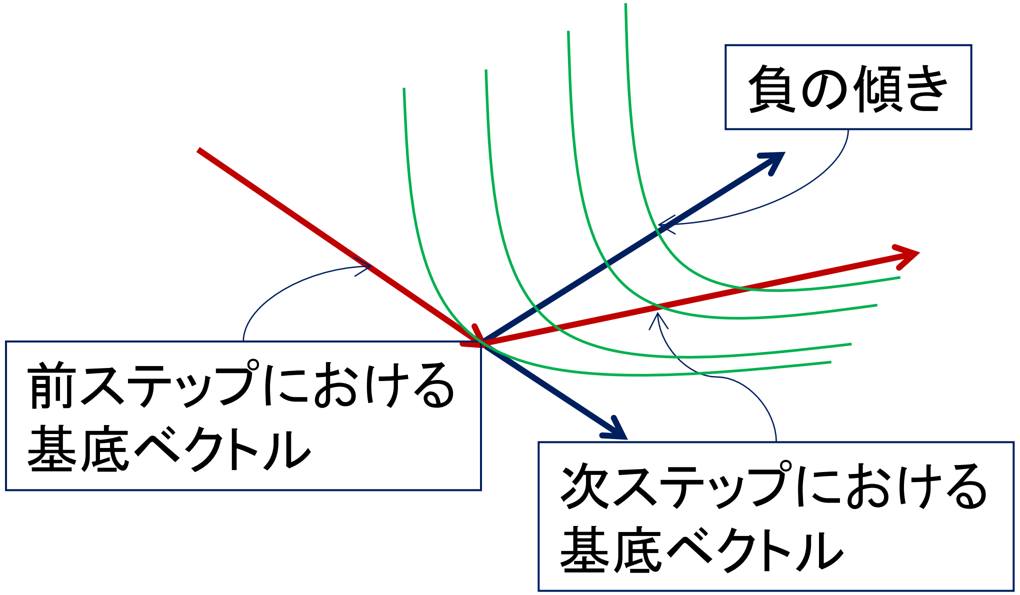
- 勾配法の一種
- 勾配だけでなく，今まで進んできた方向も考慮して探索方向を決める。
  ⇒　効率化
- 「今までの進んできた方向の考慮」にあたり，ベクトルの「共役」という性質を利用する。

「共役」　とは?

ベクトル　$\boldsymbol{x}$, $\boldsymbol{y}$　が

$$\boldsymbol{x}^T A \boldsymbol{y} = <\boldsymbol{x}, \boldsymbol{y}>_A = 0$$

を満たすこと。



固有ベクトルによる対称行列の対角化

n 次対称行列 A は,
固有ベクトル $\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_n$ を並べて作る行列

$$V = \begin{pmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \cdots & \boldsymbol{v}_n \end{pmatrix}$$

を用いて $V^T A V = \Lambda,$

$$\Lambda = \begin{pmatrix} \lambda_1 & & & \boldsymbol{0} \\ & \lambda_2 & & \\ & & \ddots & \\ \boldsymbol{0} & & & \lambda_n \end{pmatrix}$$

$\lambda_1, \lambda_2, \cdots, \lambda_n$ は固有値

と対角化できる。

行列の分解

$$V^T A V = \Lambda$$

ここで，$V^T V = V V^T = I$　であるから，

$$V V^T A V V^T = V \Lambda V^T, \quad A = V \Lambda V^T$$

対称行列は，固有値を対角要素に持つ行列 Λ と　固有ベクトルをならべて作る行列 V で表すことができる。

ベクトルの共役性

ベクトルの共役性：

定義： $\boldsymbol{x}_1^T A \boldsymbol{x}_2 = 0$ のとき， $\boldsymbol{x}_1$ と $\boldsymbol{x}_2$ は共役

解釈： $\boldsymbol{x}_1^T A \boldsymbol{x}_2 = \boldsymbol{x}_1^T V \Lambda V^T \boldsymbol{x}_2$

$$= \boldsymbol{x}_1^T \left(\Lambda^{\frac{1}{2}} V^T \right)^T \Lambda^{\frac{1}{2}} V^T \boldsymbol{x}_2 = \left(\Lambda^{\frac{1}{2}} V^T \boldsymbol{x}_1 \right)^T \Lambda^{\frac{1}{2}} V^T \boldsymbol{x}_2$$

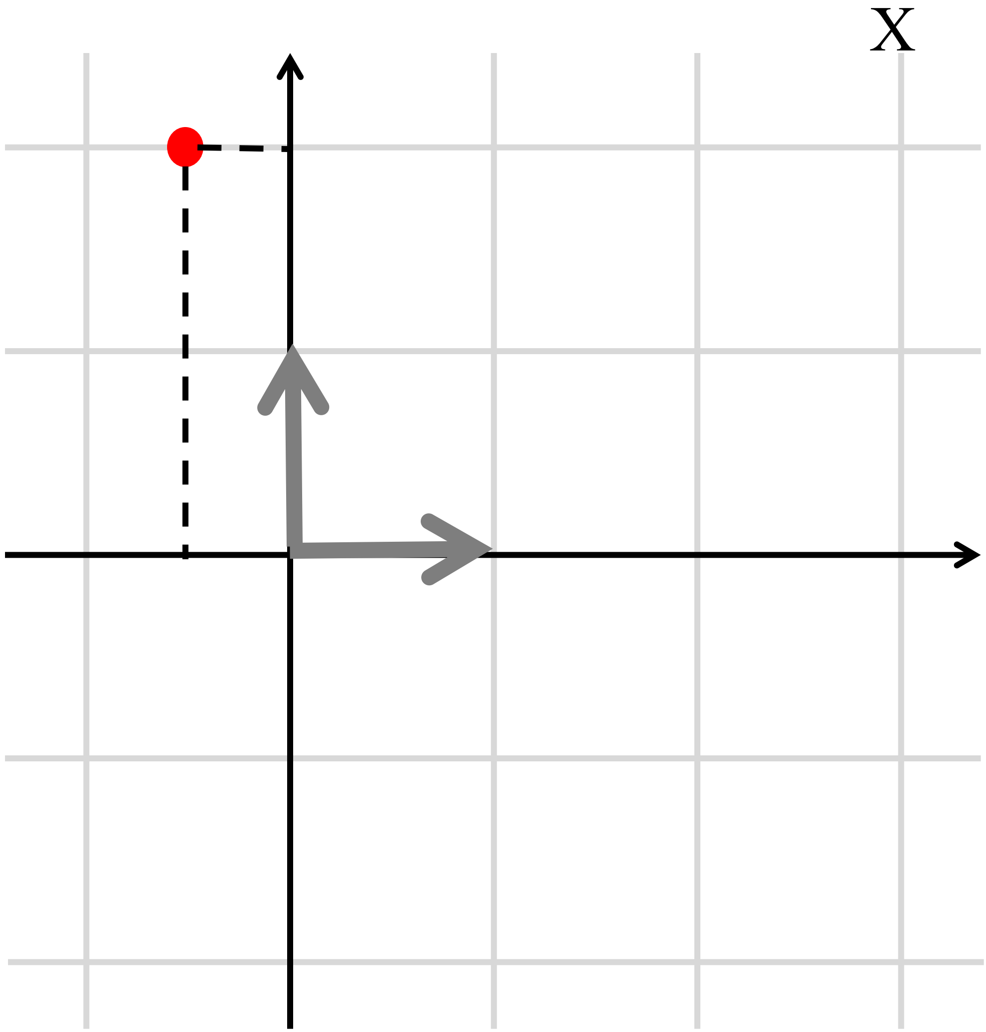
$$\boldsymbol{z} = \Lambda^{\frac{1}{2}} V^T \boldsymbol{x}$$

なる座標変換をした後のベクトルの内積

$$z = \Lambda^{\frac{1}{2}} V^T \boldsymbol{x}$$

$$z = \Lambda^{\frac{1}{2}} \boldsymbol{y}$$

$$\boldsymbol{y} = V^T \boldsymbol{x}$$

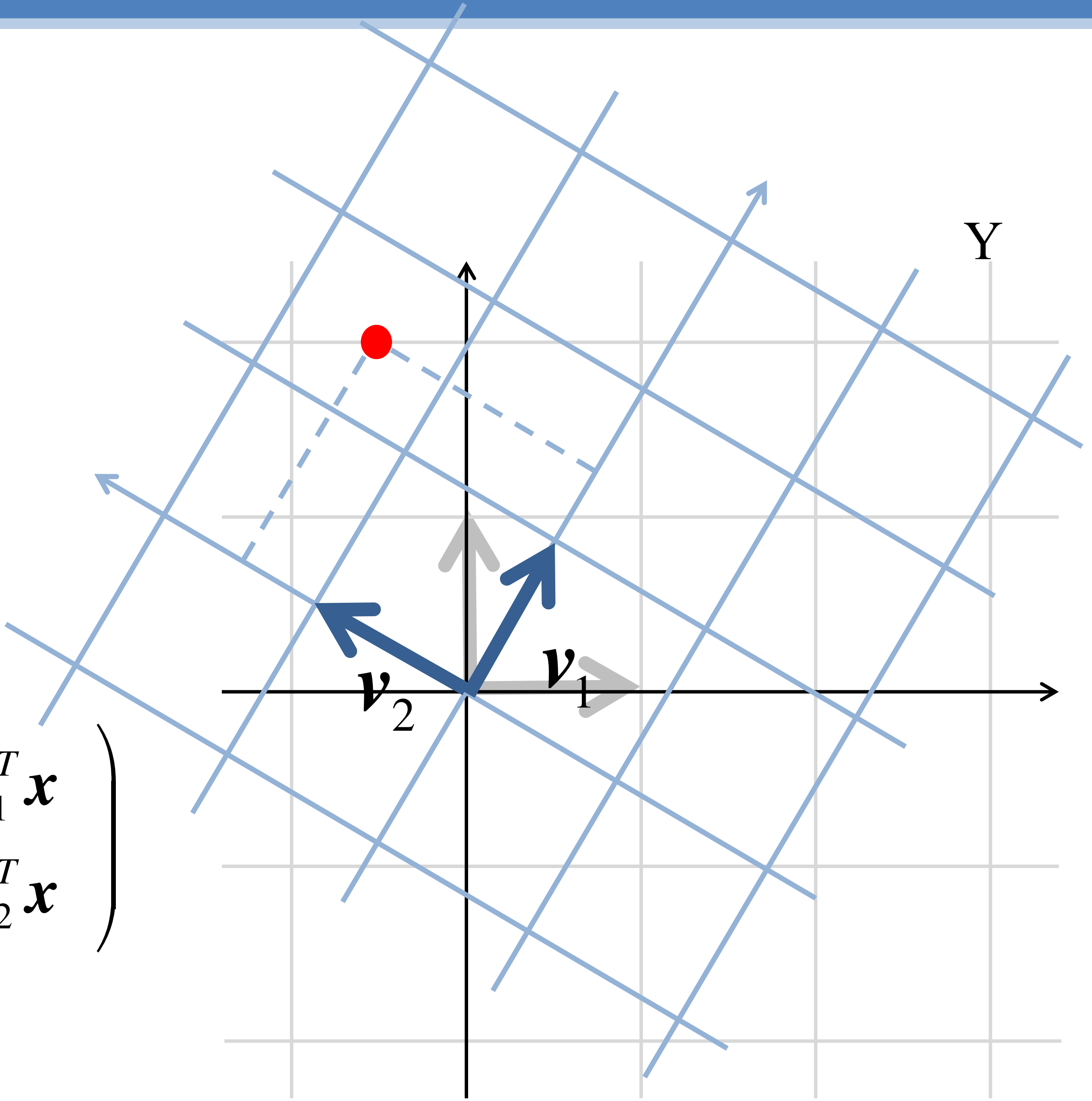


$$z = \Lambda^{\frac{1}{2}} V^T \boldsymbol{x}$$

$$z = \Lambda^{\frac{1}{2}} \boldsymbol{y}$$

$$\boldsymbol{y} = V^T \boldsymbol{x}$$

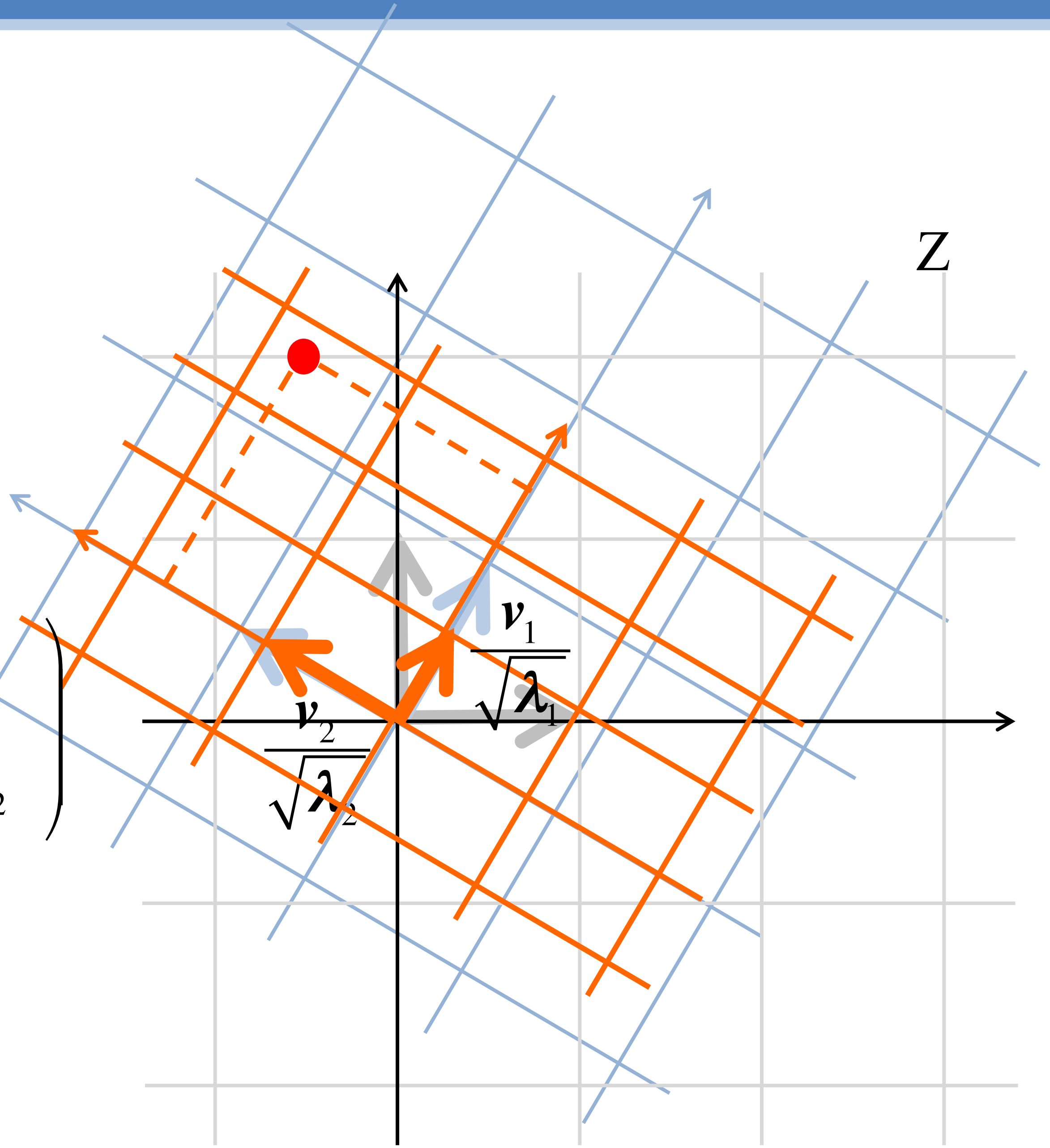
$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{v}_1^T \boldsymbol{x} \\ \boldsymbol{v}_2^T \boldsymbol{x} \end{pmatrix}$$



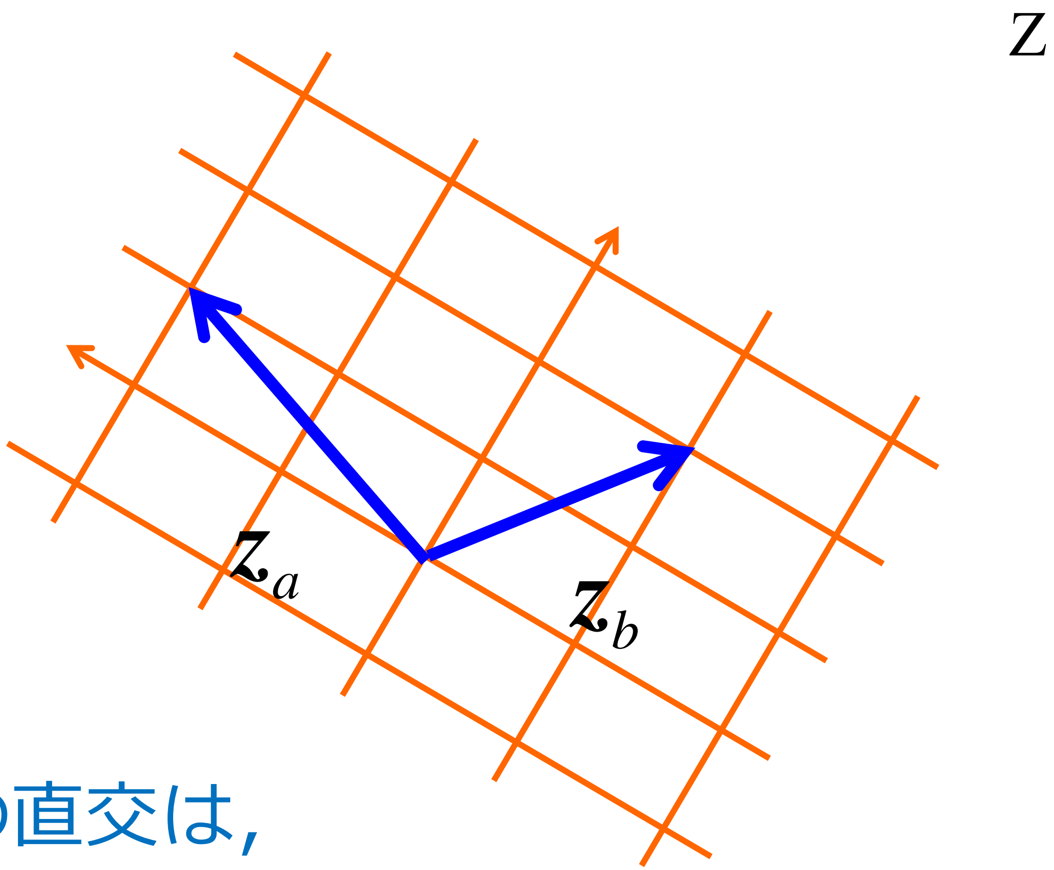
$$\boldsymbol{z} = \Lambda^{\frac{1}{2}} V^T \boldsymbol{x}$$

$$\boldsymbol{z} = \Lambda^{\frac{1}{2}} \boldsymbol{y}$$

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} \sqrt{\lambda_1}\, y_1 \\ \sqrt{\lambda_2}\, y_2 \end{pmatrix}$$



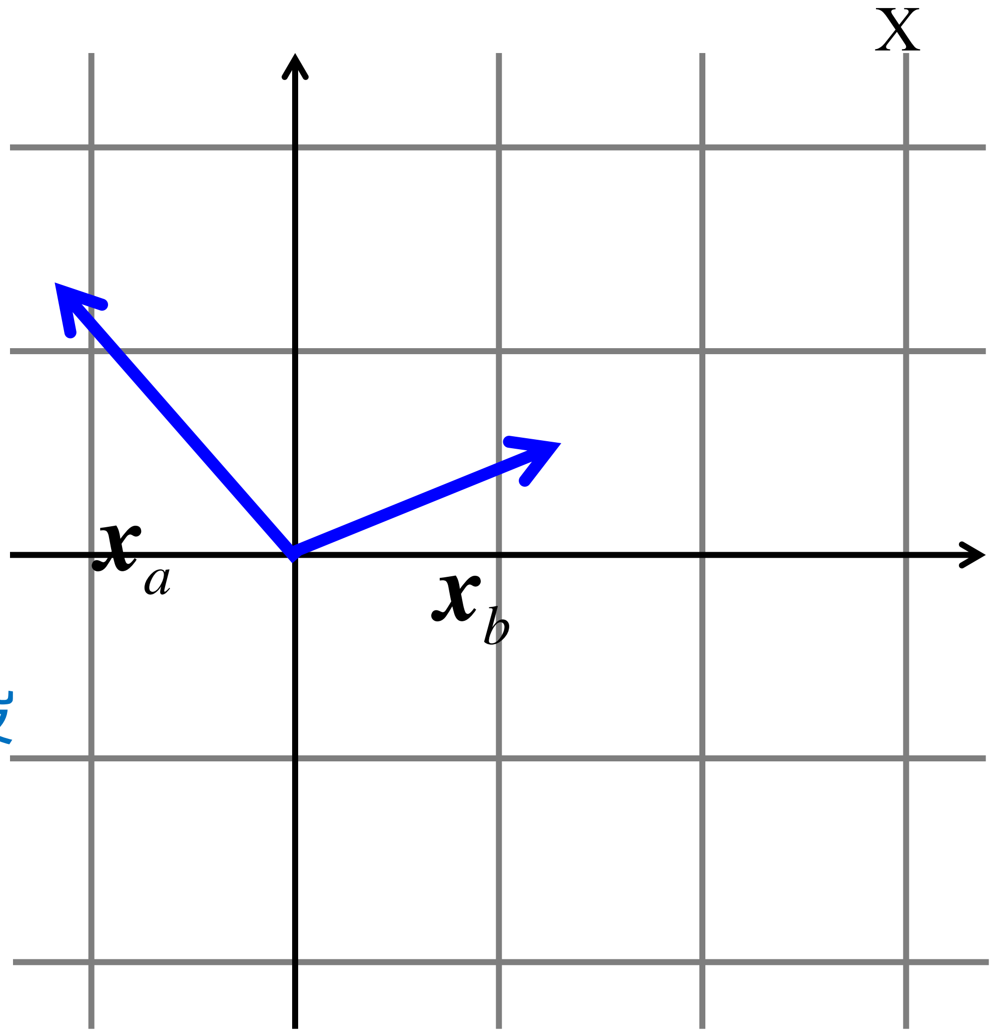
直交



座標変換後の直交は,

$$\mathbf{z}_a^T \cdot \mathbf{z}_b = 0$$

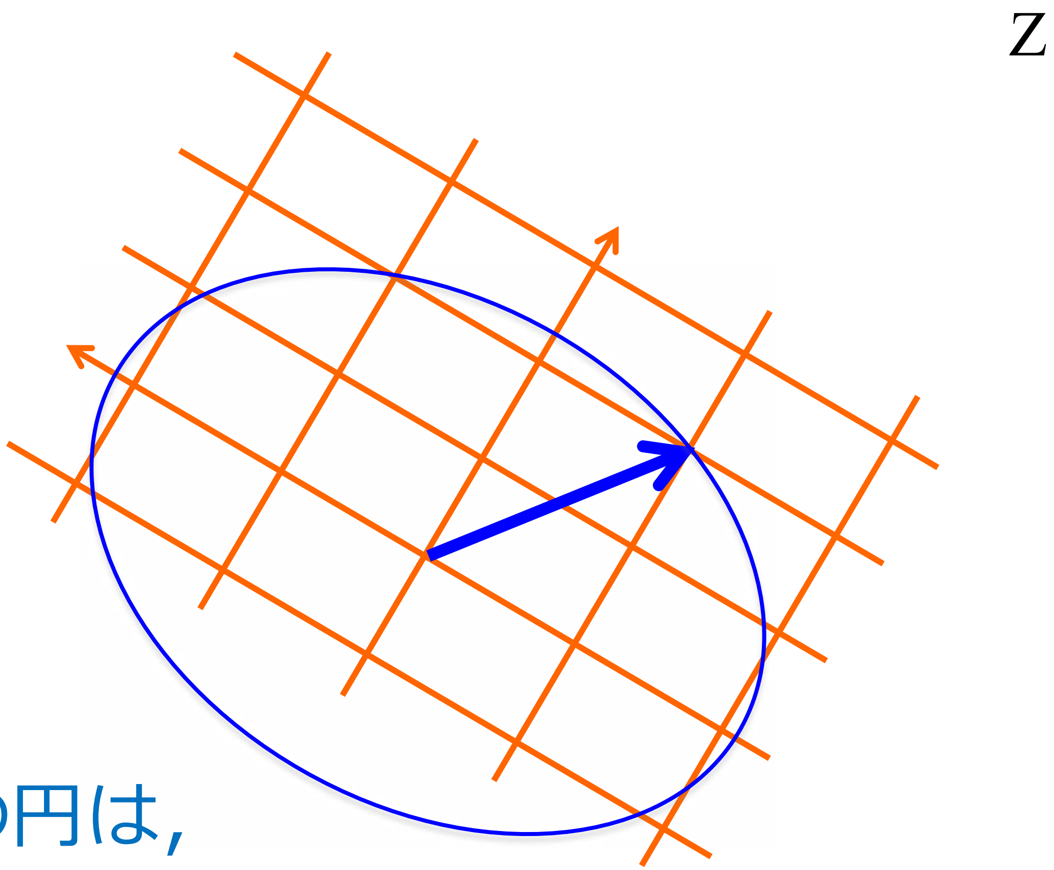
共役



元の座標では共役

$$\boldsymbol{x}_a^T A \boldsymbol{x}_b = 0$$

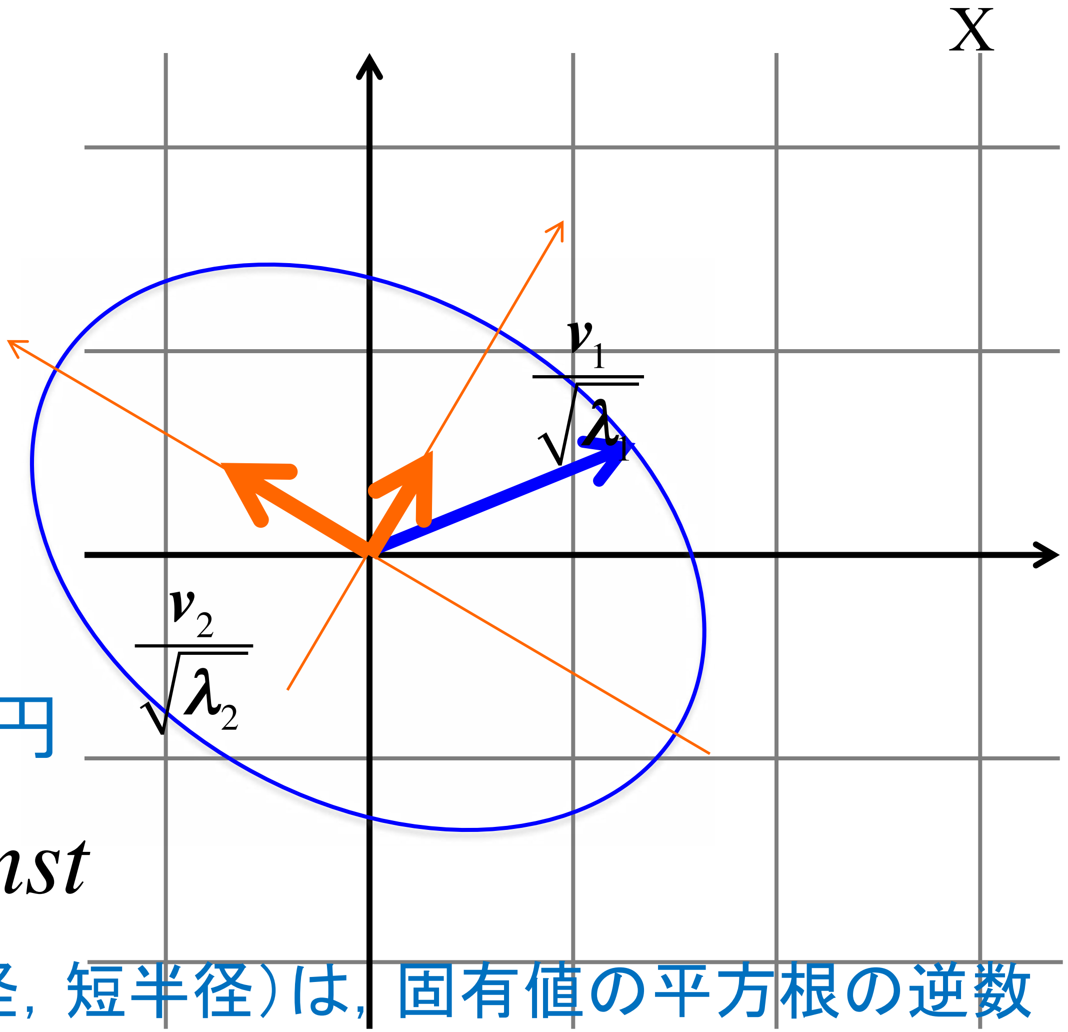
円



座標変換後の円は,

$$z^T \cdot z = const$$

楕円



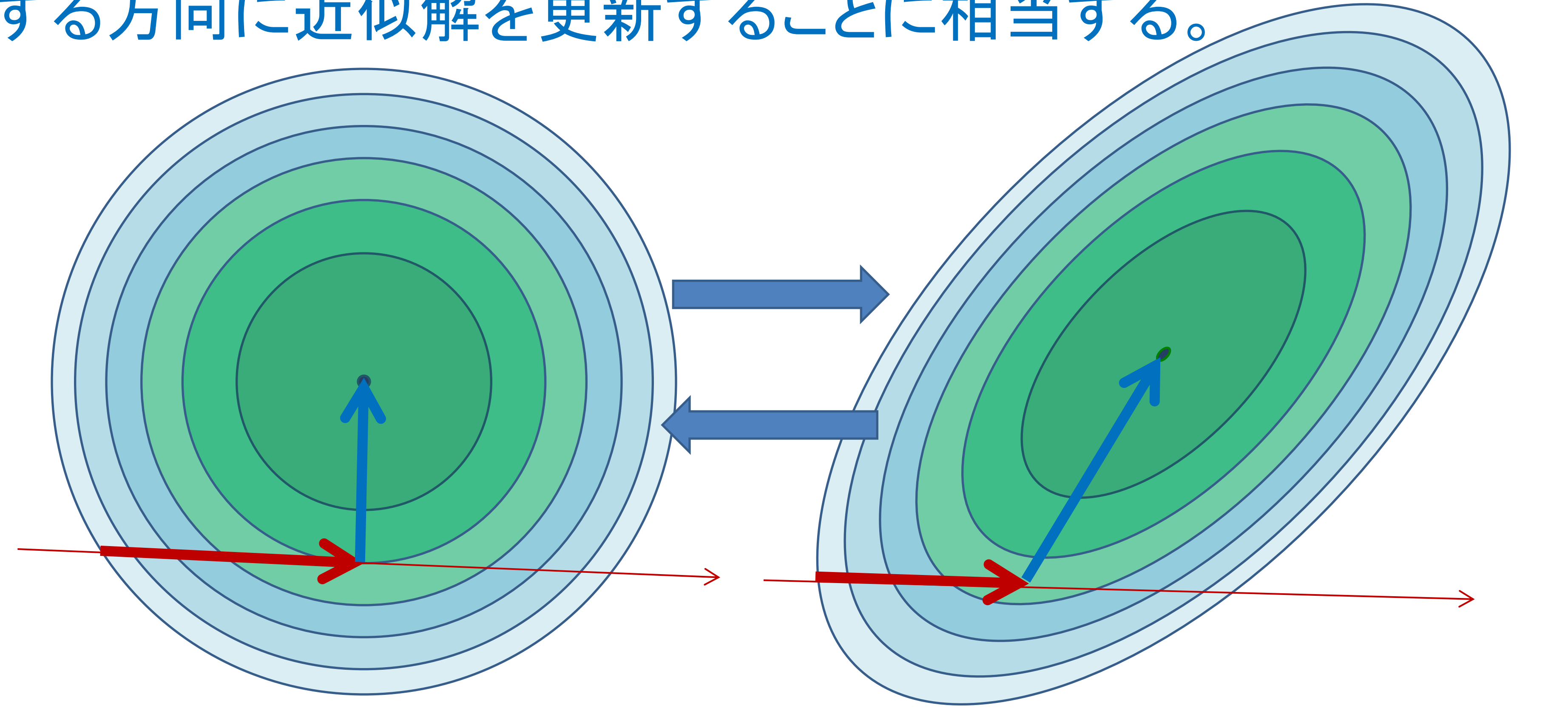
元の座標では楕円

$$x^T A x = const$$

半径(長半径，短半径)は，固有値の平方根の逆数

共役とは

共役の方向に近似解を更新することは，
楕円を円に変換したとき，
直交する方向に近似解を更新することに相当する。



共役なベクトルによる $\boldsymbol{Ax} = \boldsymbol{b}$ の解の表現

互いに共役なベクトル$\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_N$を用いて，

$$A\boldsymbol{x} = \boldsymbol{b} \qquad (1)$$

（Aは，N次正定値対象行列）

の解 $\boldsymbol{x}^*$を表現することを考える。

$$\boldsymbol{x}^* = \sum_i^N c_i \boldsymbol{p}_i \qquad \boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_N \qquad (2)$$

とおくと，$\boldsymbol{p}_i$は，$\boldsymbol{p}_j$ $(i \neq j)$ と共役であるから，

$$\boldsymbol{p}_i{}^T A\boldsymbol{x}^* = \boldsymbol{p}_i{}^T \sum_k^N c_k A\boldsymbol{p}_k = c_i \boldsymbol{p}_i{}^T A\boldsymbol{p}_i = \boldsymbol{p}_i{}^T \boldsymbol{b}$$

$$c_i = \frac{\boldsymbol{p}_i{}^T \boldsymbol{b}}{\boldsymbol{p}_i{}^T A\boldsymbol{p}_i} \qquad (3)$$

と c_i を定めることができる。

共役なベクトル$\{\boldsymbol{p}_i\}$を用意すれば，$A\boldsymbol{x}$=$\boldsymbol{b}$の解は，それらを基底として表現できる。

基底$\boldsymbol{p}_i$の係数は，$c_i = \frac{\boldsymbol{p}_i{}^T \boldsymbol{b}}{\boldsymbol{p}_i{}^T A\boldsymbol{p}_i}$ となる。

共役勾配法　（2次形式の場合）

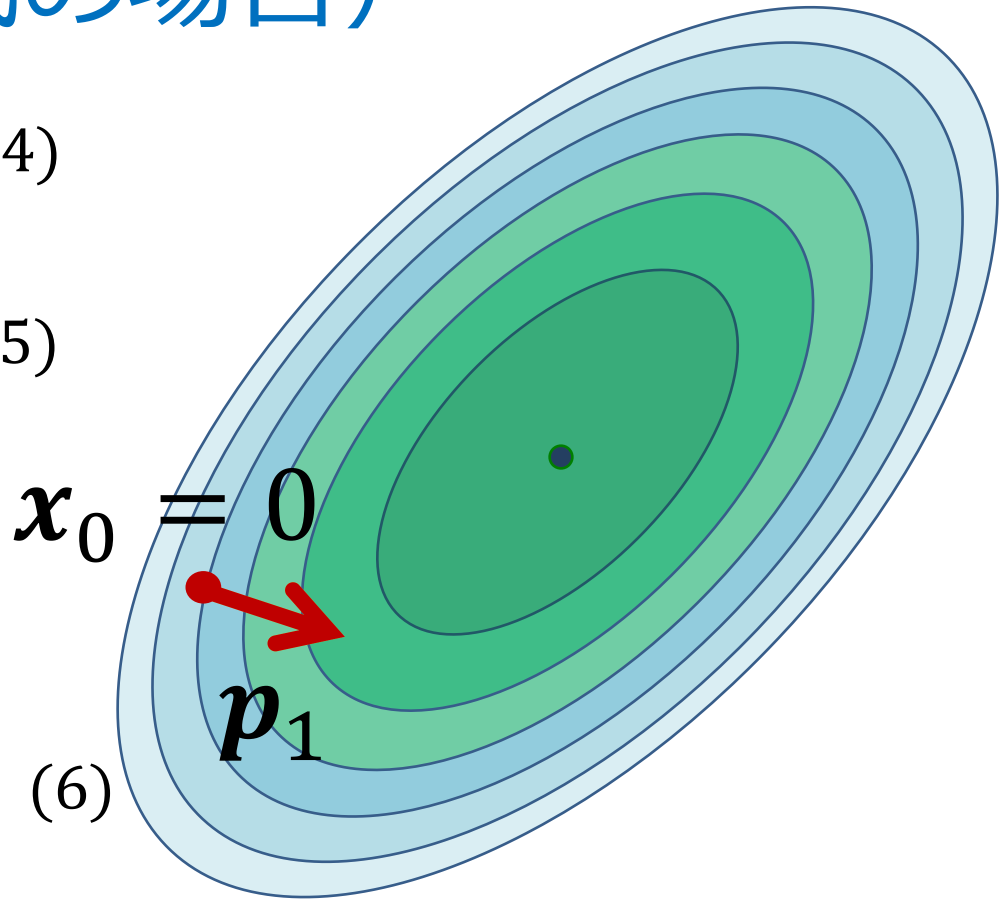
2次形式　$f(\boldsymbol{x}) = \dfrac{1}{2}\boldsymbol{x}^T A\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}$　(4)

に対し，　$\nabla f(\boldsymbol{x}) = A\boldsymbol{x} - \boldsymbol{b} = 0$　(5)

即ち，$A\boldsymbol{x} = \boldsymbol{b}$ の解を求める。

$\boldsymbol{x}_0 = 0$ として初期値を定め，

$$\boldsymbol{x}_k = \sum_{i=1}^{k} \boldsymbol{p}_k = \boldsymbol{x}_{k-1} + c_k\boldsymbol{p}_k \quad (6)$$



としながら,$\boldsymbol{p}$　,c　,$\boldsymbol{x}$　を漸化的に求める。

最初の基底 $\boldsymbol{p}_1$ に適当な値をあたえる。

共役勾配法

第k次の基底 p_k が定まったとする。

これを用いて， c_k, x_k を

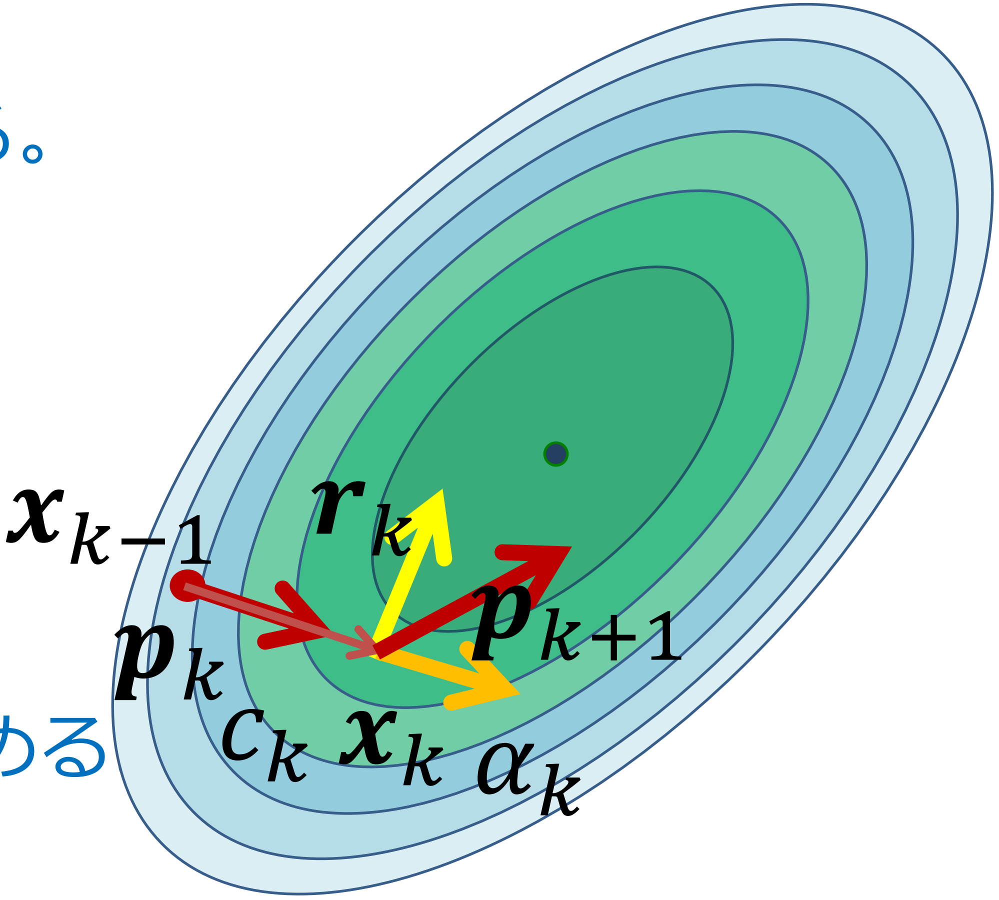
$$c_k = \frac{\boldsymbol{p}_k{}^T \boldsymbol{b}}{\boldsymbol{p}_k{}^T A \boldsymbol{p}_k} \qquad (3)$$

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + c_k \boldsymbol{p}_k \qquad (6)$$

と求め，

さらに，x_kでの負の勾配 r_k を求める

$$\boldsymbol{r}_k = -\nabla f(\boldsymbol{x}_k) = -A\boldsymbol{x}_k + \boldsymbol{b} \qquad (7)$$



第k+1次の基底 p_{k+1} を，r_k と p_k の合成，すなわち，

$$\boldsymbol{p}_{k+1} = \boldsymbol{r}_k + \alpha_k \boldsymbol{p}_k \qquad (8)$$

とおいて，これがp_kと共役になるように定める。

共役勾配法

α_kは，

$$\boldsymbol{p}_k{}^T A\boldsymbol{p}_{k+1} = \boldsymbol{p}_k{}^T A(\boldsymbol{r}_k + \alpha_k \boldsymbol{p}_k) = \boldsymbol{p}_k{}^T A\boldsymbol{r}_k + \alpha_k \boldsymbol{p}_k{}^T \boldsymbol{p}_k = 0$$

$$\alpha_k = -\frac{\boldsymbol{p}_k{}^T A\boldsymbol{r}_k}{\boldsymbol{p}_k{}^T A\boldsymbol{p}_k} \qquad (9)$$

第k+1次の基底は，

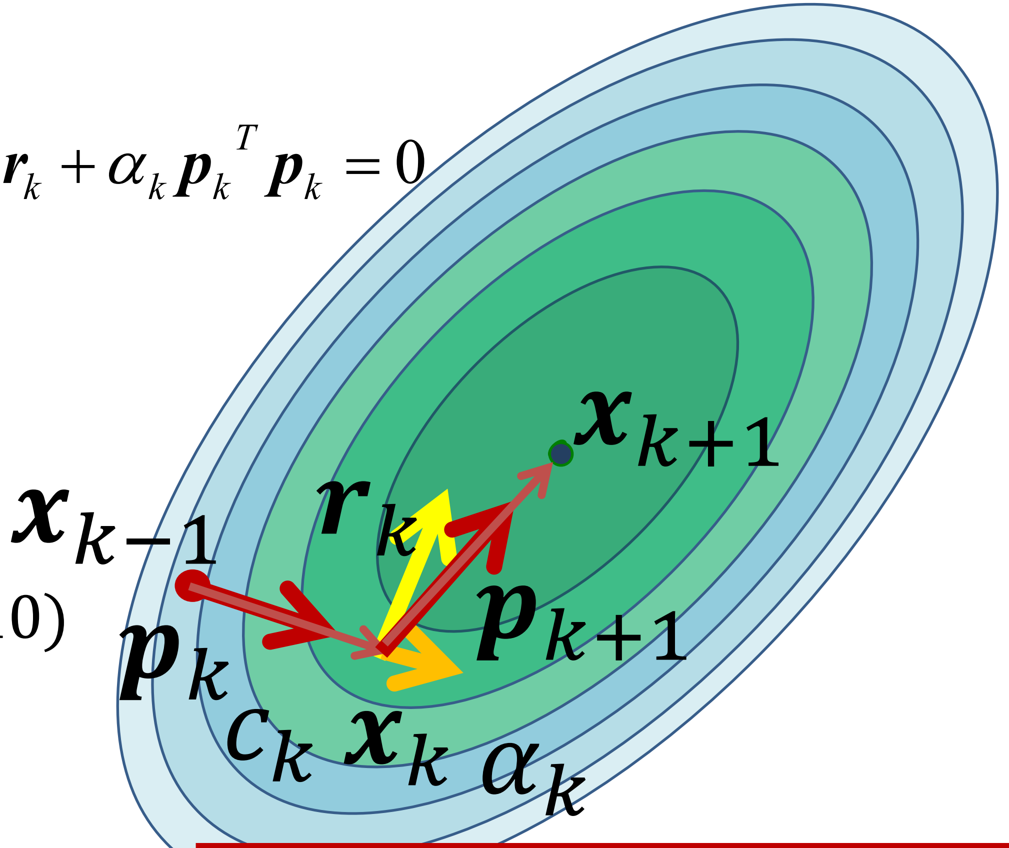
$$\boldsymbol{p}_{k+1} = \boldsymbol{r}_k + \alpha_k \boldsymbol{p}_k = \boldsymbol{r}_k - \frac{\boldsymbol{p}_k{}^T A\boldsymbol{r}_k}{\boldsymbol{p}_k{}^T A\boldsymbol{p}_k}\boldsymbol{p}_k \qquad (10)$$

となる。

さらに，これを用いて，x_{k+1} を，

$$c_{k+1} = \frac{\boldsymbol{p}_{k+1}{}^T \boldsymbol{b}}{\boldsymbol{p}_{k+1}{}^T A\boldsymbol{p}_{k+1}} \qquad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + c_{k+1}\boldsymbol{p}_{k+1} \qquad (11)$$

と求める。

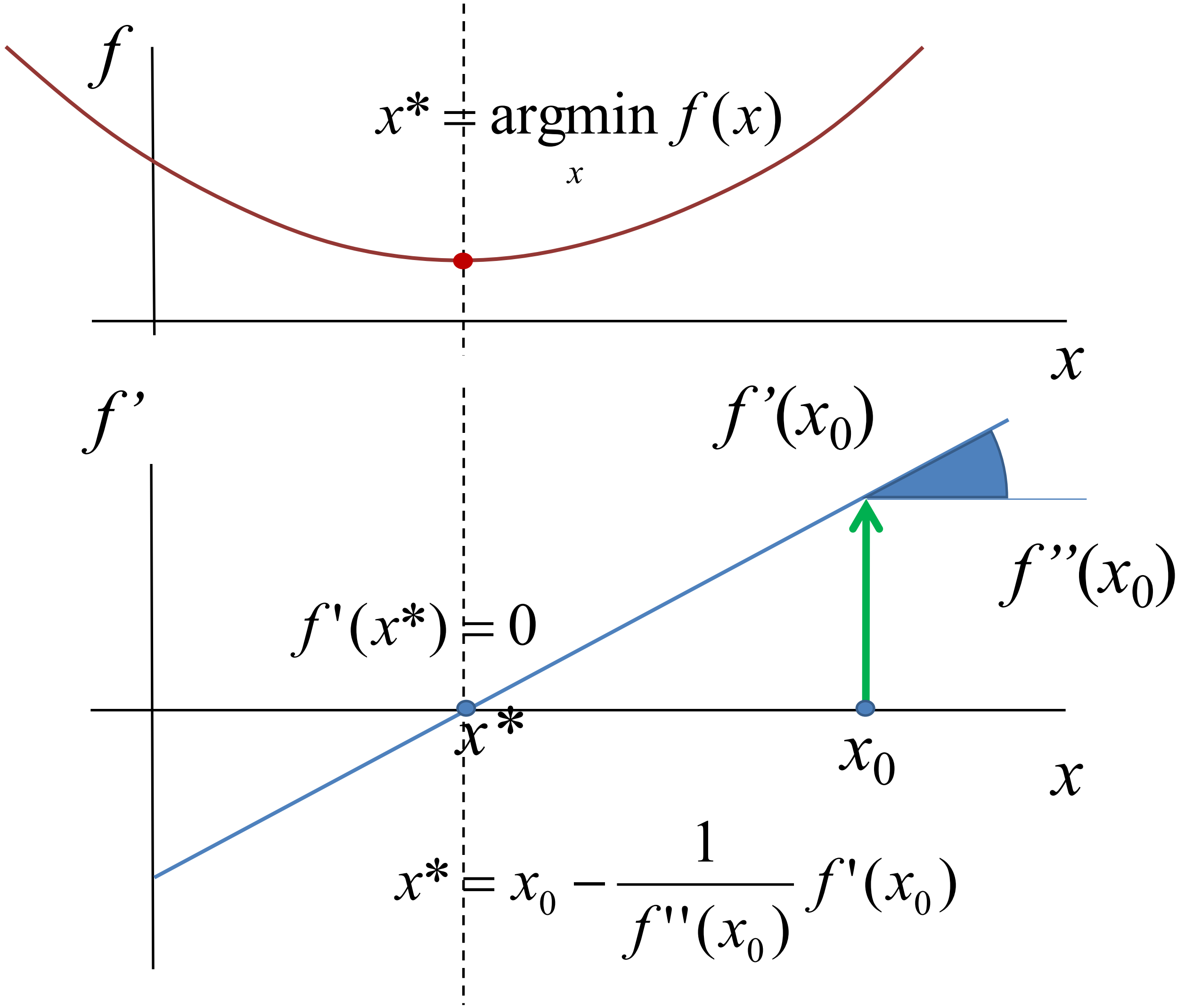


関数が2次形式であれば，
くりかえし回数は，
たかだか共役なベクトルの数
（＝xの次元数）となる。

§ ニュートン法

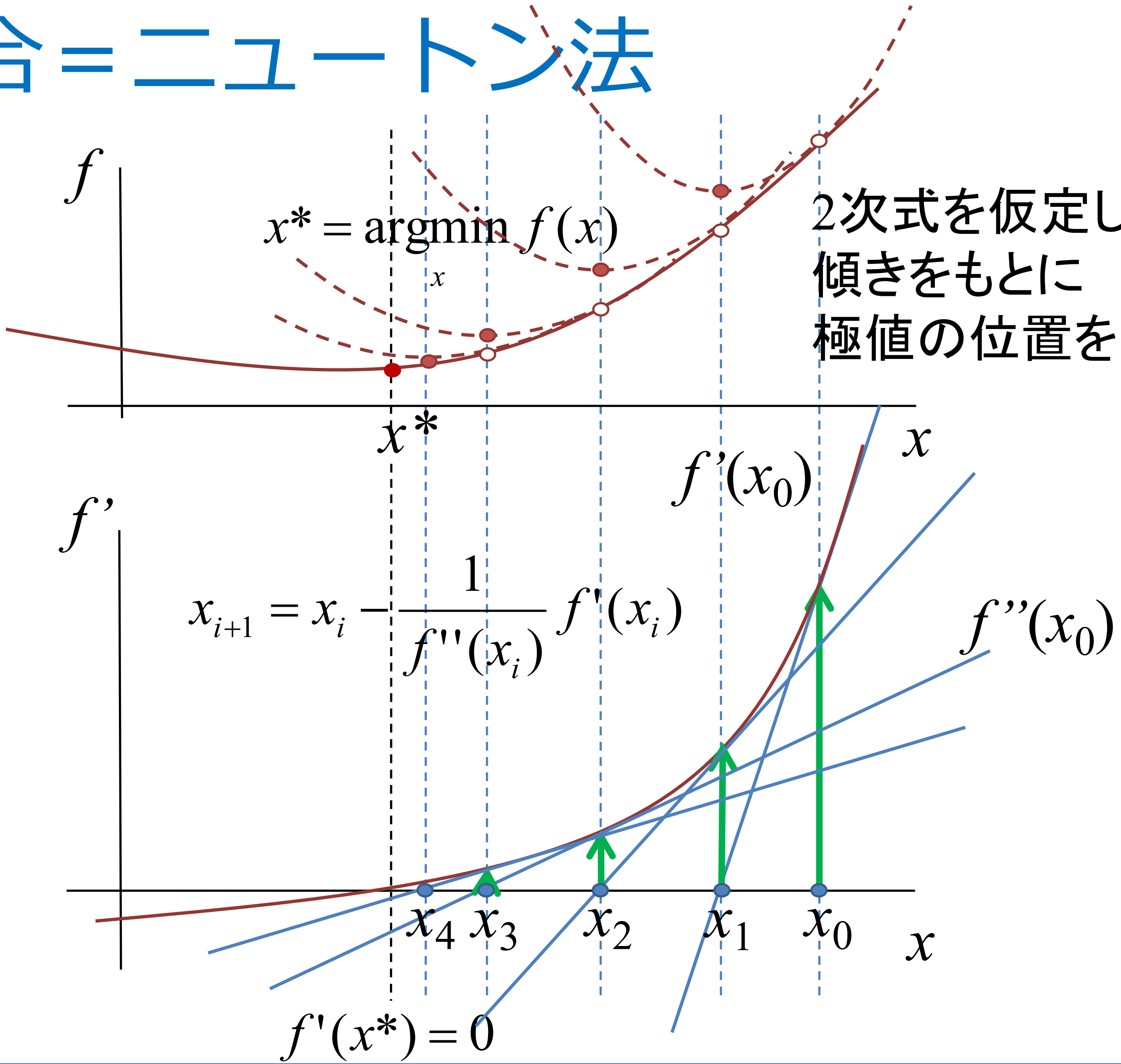
- 勾配だけでなく，ヘシアン（スカラにおける２次微係数に相当する量）も使って，近似解の変更の方向と量を決定する。

２次式の最小化問題



f
$x^* = \underset{x}{\operatorname{argmin}} f(x)$
x
f'
$f'(x_0)$
$f''(x_0)$
$f'(x^*) = 0$
x*
x_0
x
$x^* = x_0 - \frac{1}{f''(x_0)} f'(x_0)$

一般の場合＝ニュートン法



2次式を仮定して
傾きをもとに
極値の位置を推定

多変数のニュートン法

$$H(\boldsymbol{x}_0)=\begin{pmatrix}\dfrac{\partial^2 f(\boldsymbol{x}_0)}{\partial x_1\partial x_1} & \dfrac{\partial^2 f(\boldsymbol{x}_0)}{\partial x_1\partial x_2}\\ \dfrac{\partial^2 f(\boldsymbol{x}_0)}{\partial x_2\partial x_1} & \dfrac{\partial^2 f(\boldsymbol{x}_0)}{\partial x_2\partial x_2}\end{pmatrix}$$

:ヘッセ行列
（ヘシアン，ヘシアン行列）

$$\frac{\partial}{\partial \boldsymbol{x}} f(\boldsymbol{x}_0+\Delta\boldsymbol{x})=\nabla f(\boldsymbol{x}_0)+H(\boldsymbol{x}_0)\Delta\boldsymbol{x}=0$$

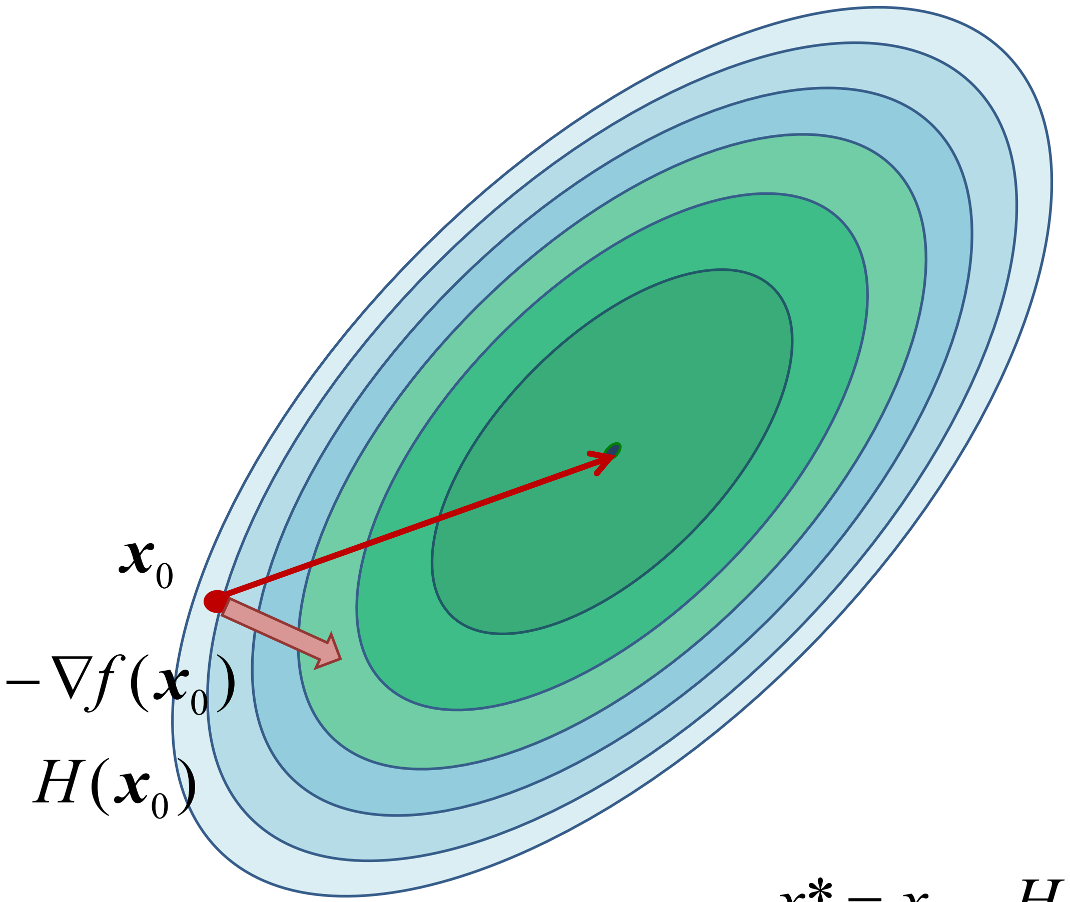
$$\Delta\boldsymbol{x}=-H(\boldsymbol{x}_0)^{-1}\nabla f(\boldsymbol{x}_0)$$

$$x^*=x_0-H(\boldsymbol{x}_0)^{-1}\nabla f(\boldsymbol{x}_0)$$ （2次系の場合は即最適解）

$$x_{i+1}=x_i-H(\boldsymbol{x}_i)^{-1}\nabla f(\boldsymbol{x}_i)$$

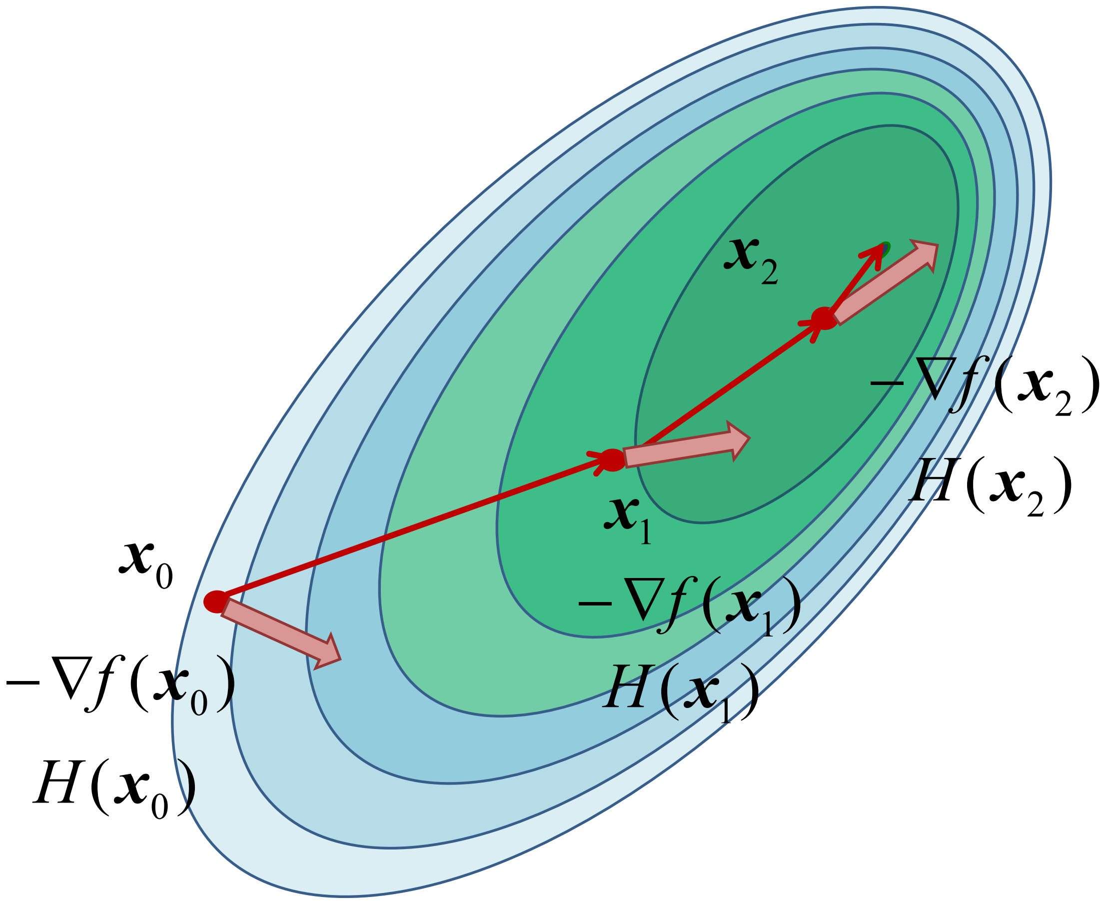
（一般の場合は漸化式とする）

ニュートン法：2次系の場合



$$x^* = x_0 - H(\boldsymbol{x}_0)^{-1}\nabla f(\boldsymbol{x}_0)$$

ニュートン法：一般の場合

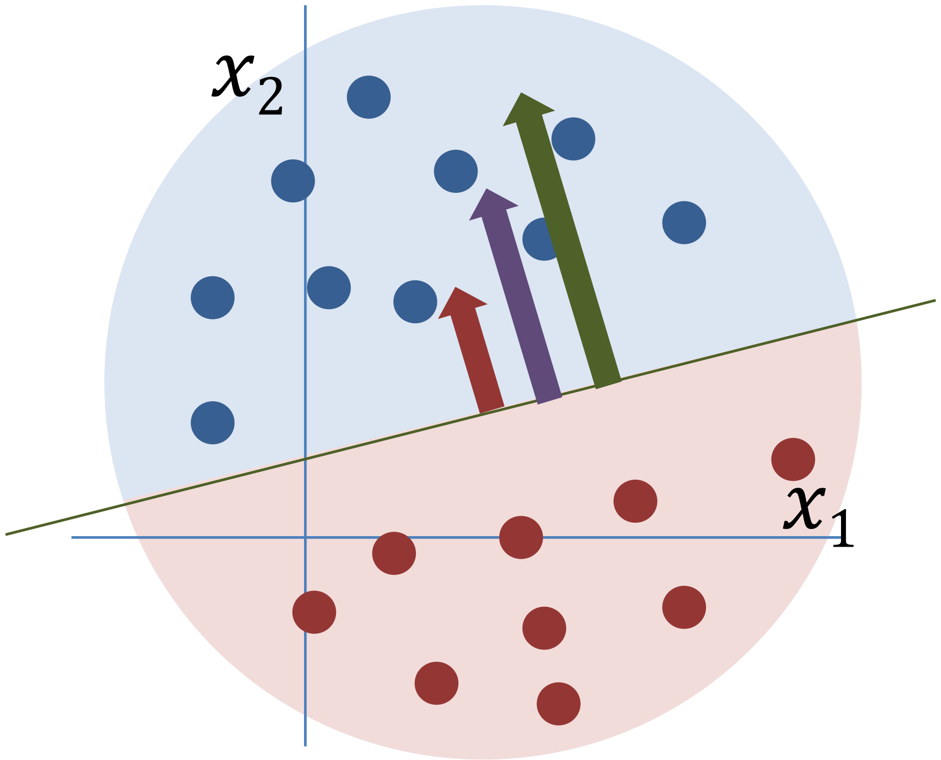


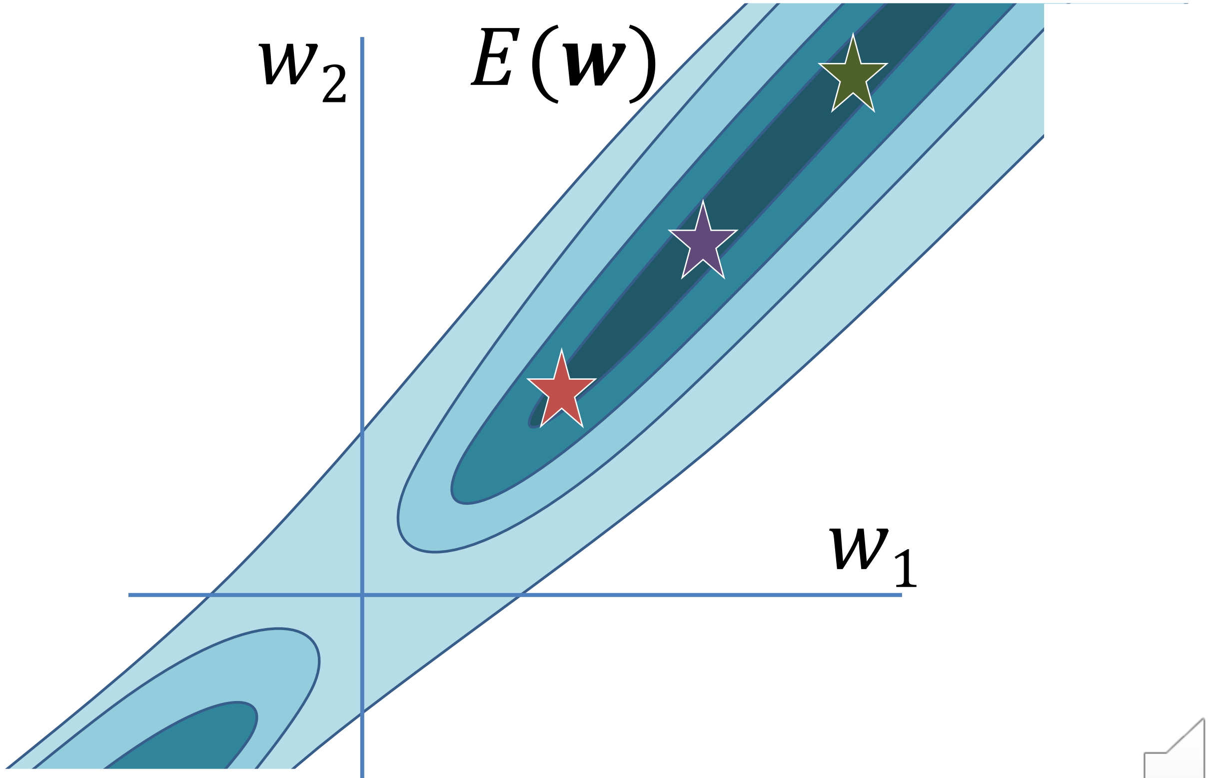
$$x_{i+1} = x_i - H(\boldsymbol{x}_i)^{-1}\nabla f(\boldsymbol{x}_i)$$

§ L2正則化

幾つかの問題で不定解になることがある。

例）ある方向の法線ベクトルでデータが既に分離できているなら，その大きさは一定以上大きければ目的関数値に影響を与えない。



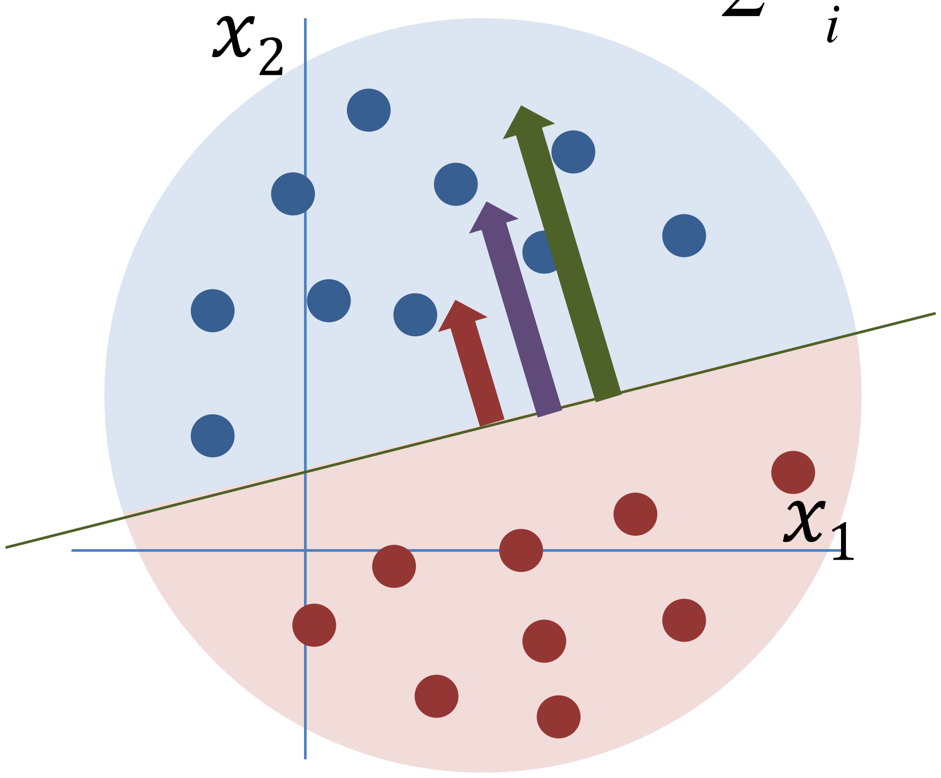


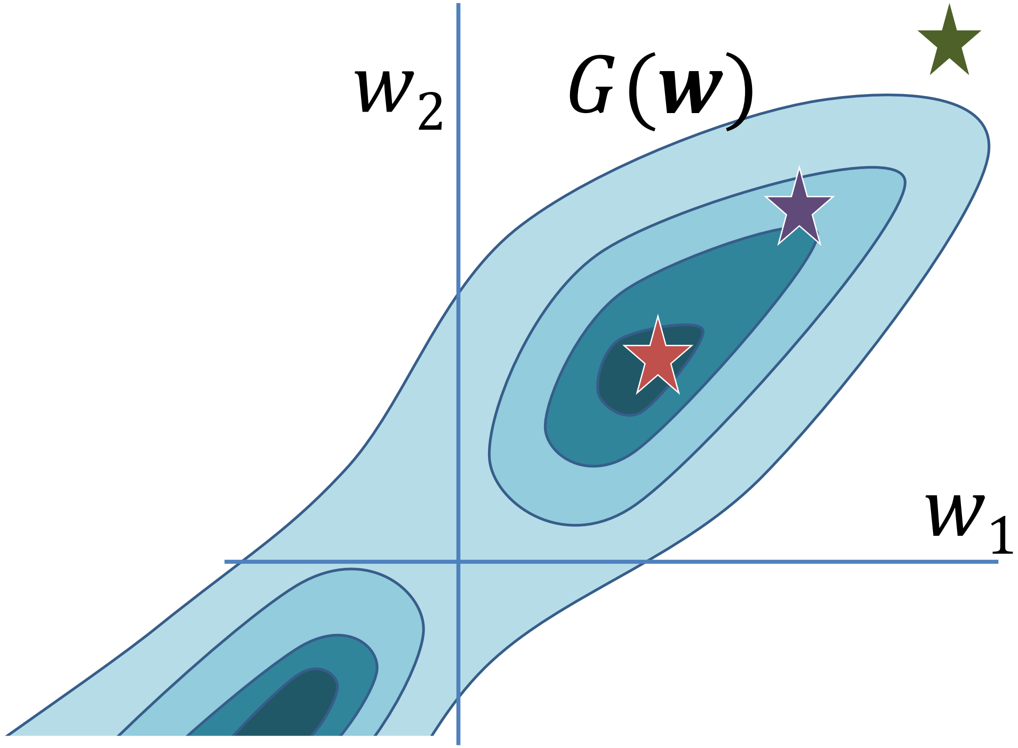
この場合は，法線ベクトルの大きさは小さい方が望ましいと考える。

L2正則化

L2 Norm

$$G = \frac{1}{2}\sum_i (o_i^0 - d_i)^2 + \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

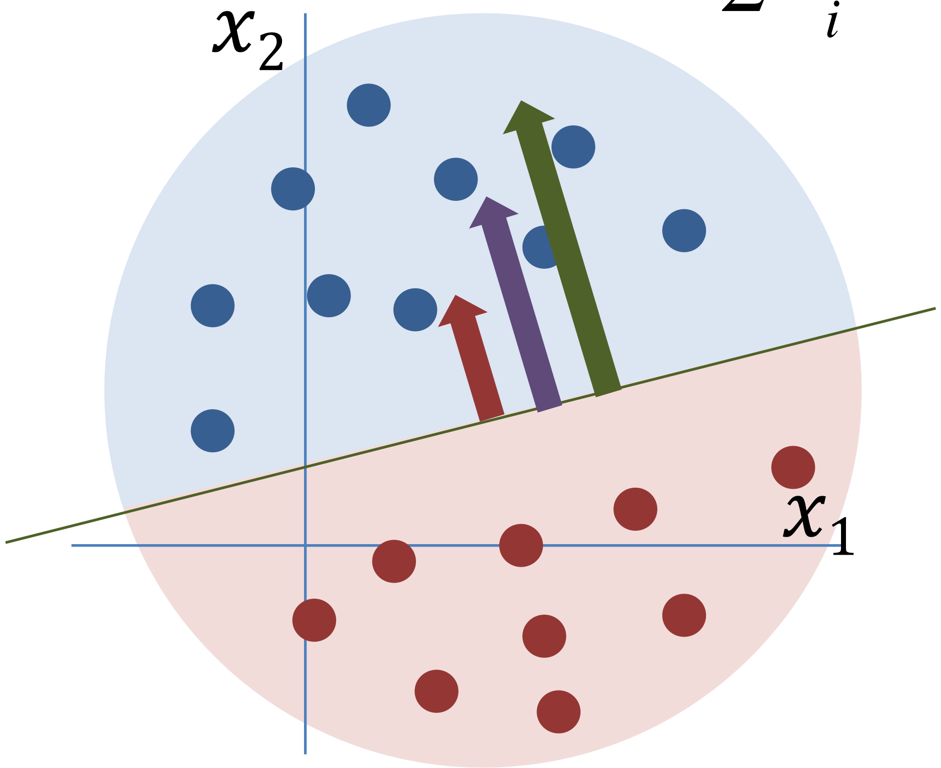


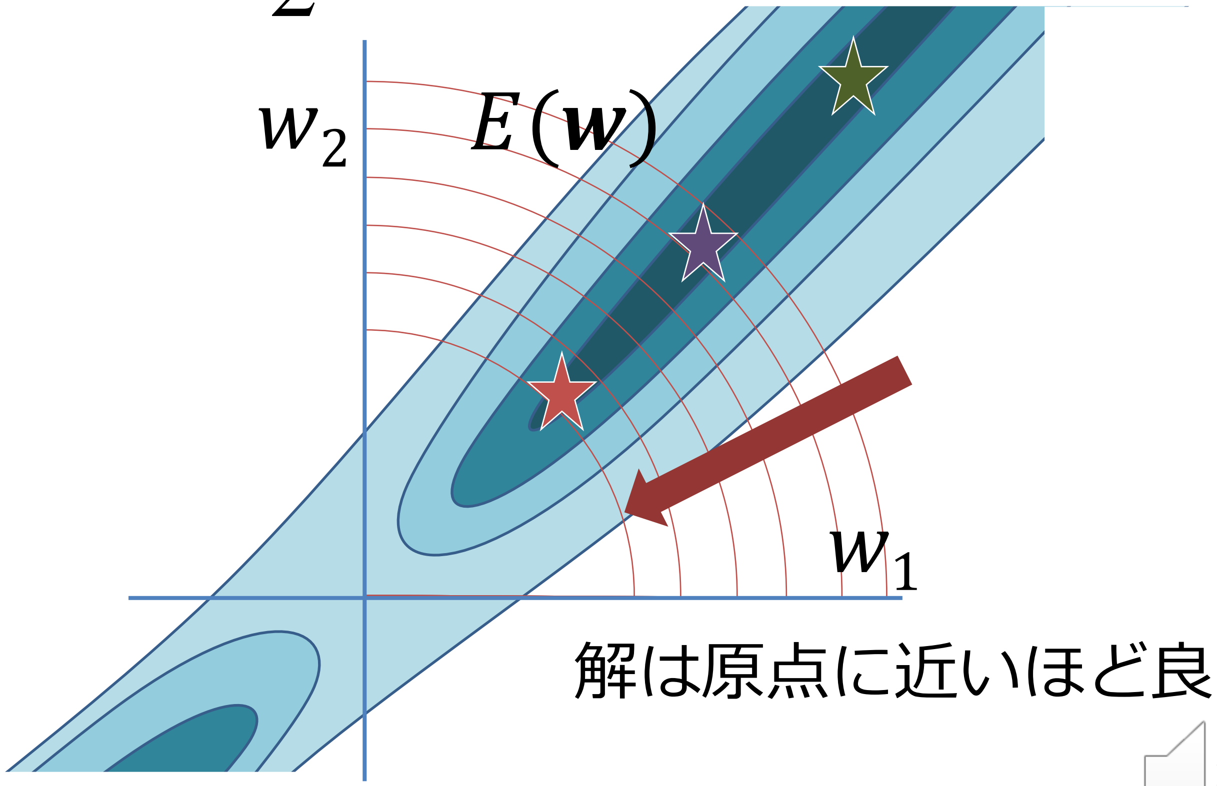


L2正則化

L2 Norm

$$G = \frac{1}{2}\sum_i (o_i^0 - d_i)^2 + \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

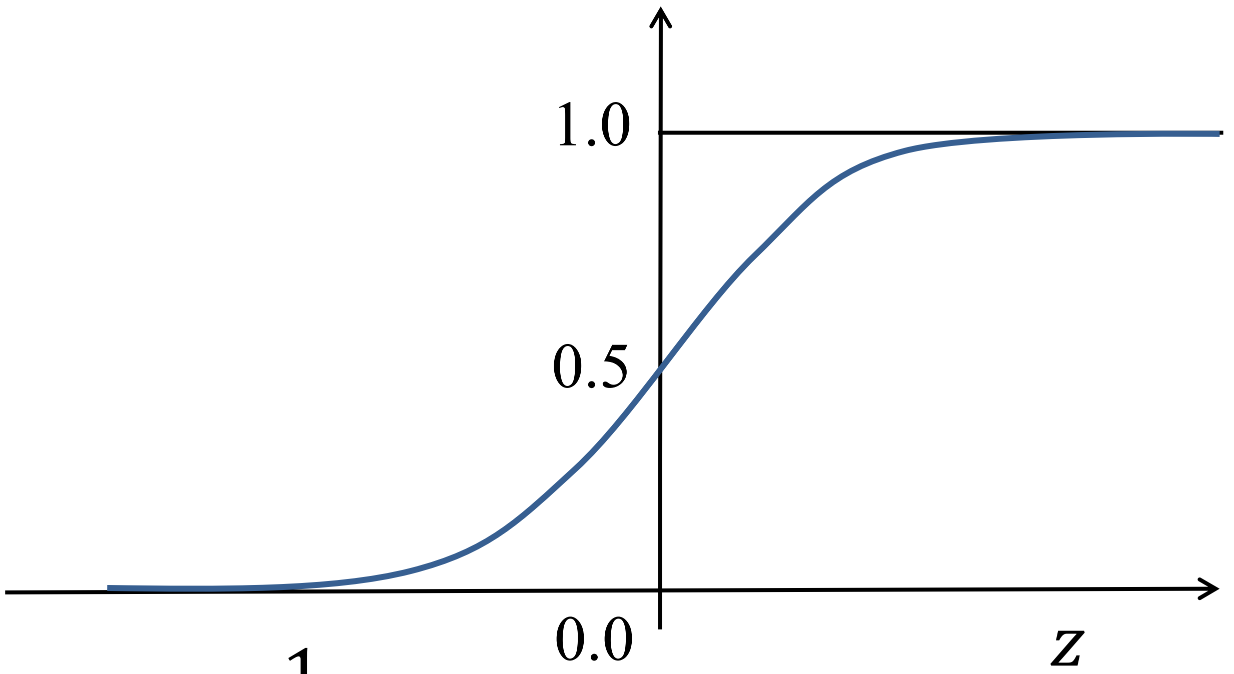




解は原点に近いほど良い

何故ノルムは小さい方が良いと考えるのか

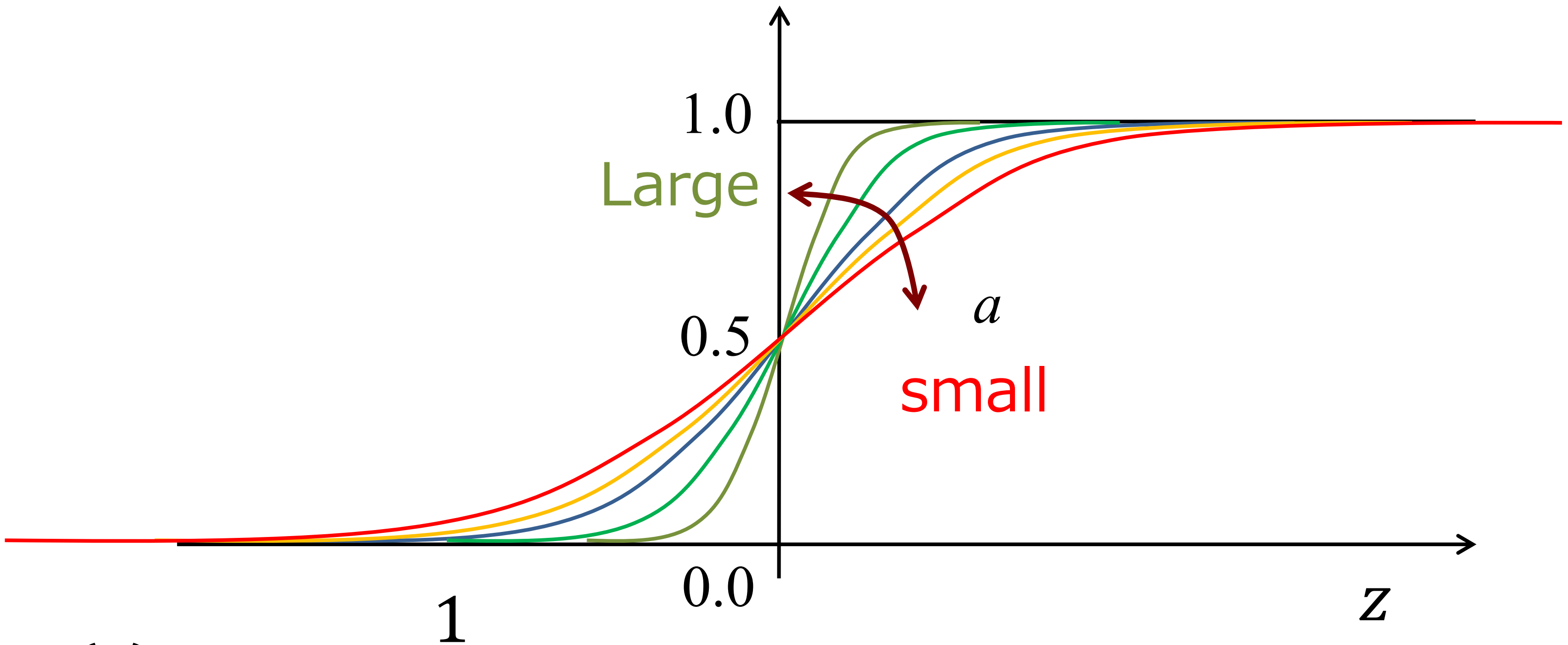
- シグモイド関数



$$f(z) = \frac{1}{1 + \exp(-z)}$$

何故ノルムは小さい方が良いと考えるのか

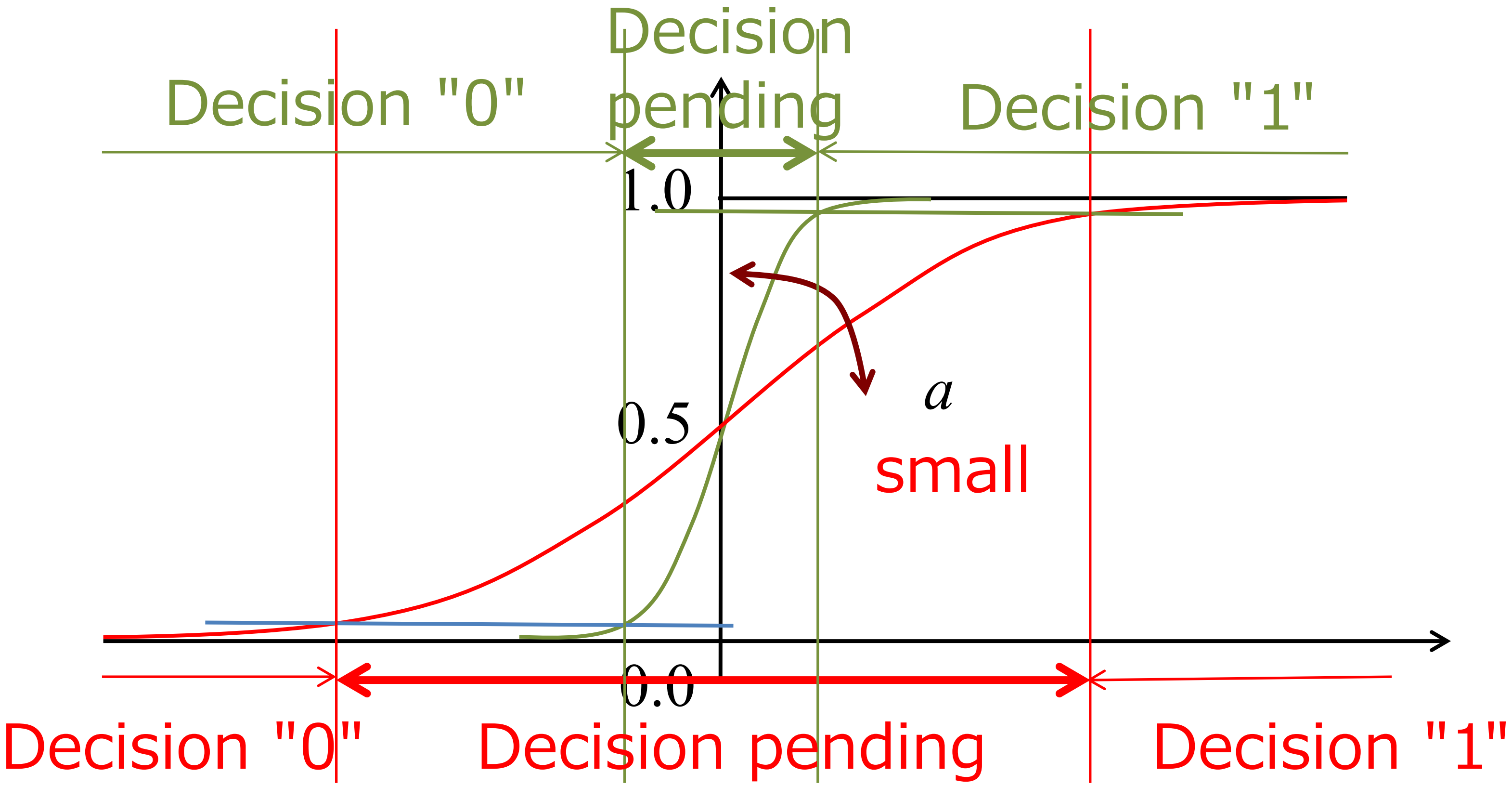
- パラメタ a はシグモイド関数の立ち上がり部分の傾斜の深さに関係する



$$f(z) = \frac{1}{1 + \exp(-az)}$$

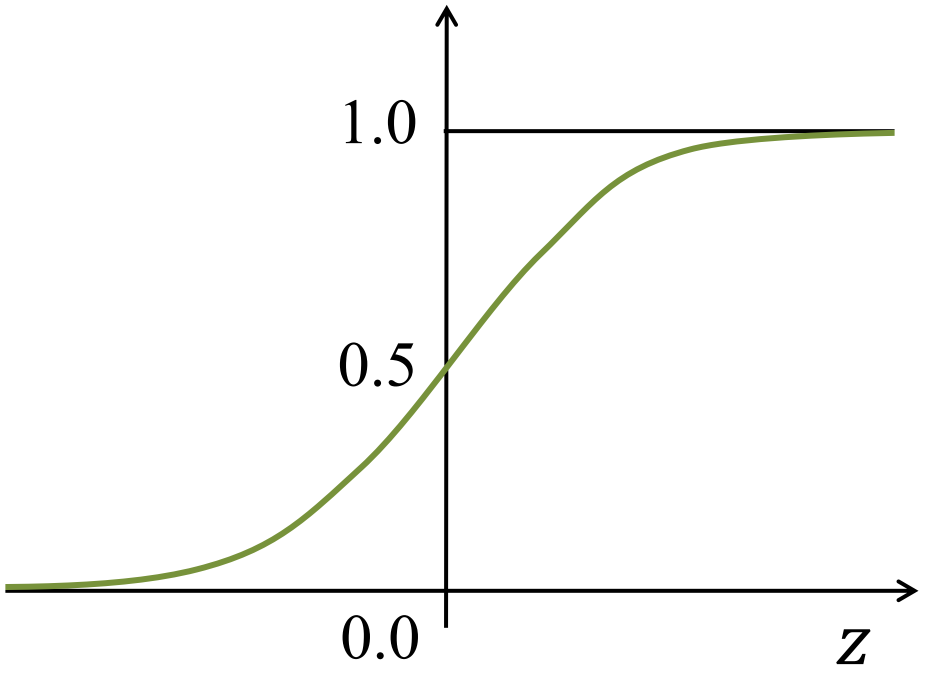
何故ノルムは小さい方が良いと考えるのか

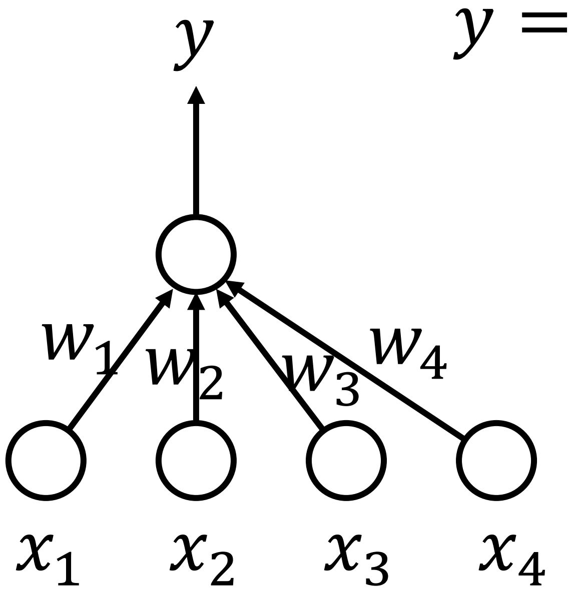
- "a" が小さければゆるい傾斜 = 広い範囲で決定が保留される
  ⇒ 慎重な識別器



何故ノルムは小さい方が良いと考えるのか

- 線形識別器の重みの大きさは，シグモイドの傾斜パラメタ a に関係している。





$$y = \frac{1}{1 + \exp(-az)}$$

$$y = \frac{1}{1 + \exp(-\boldsymbol{w}^T \boldsymbol{x})}$$

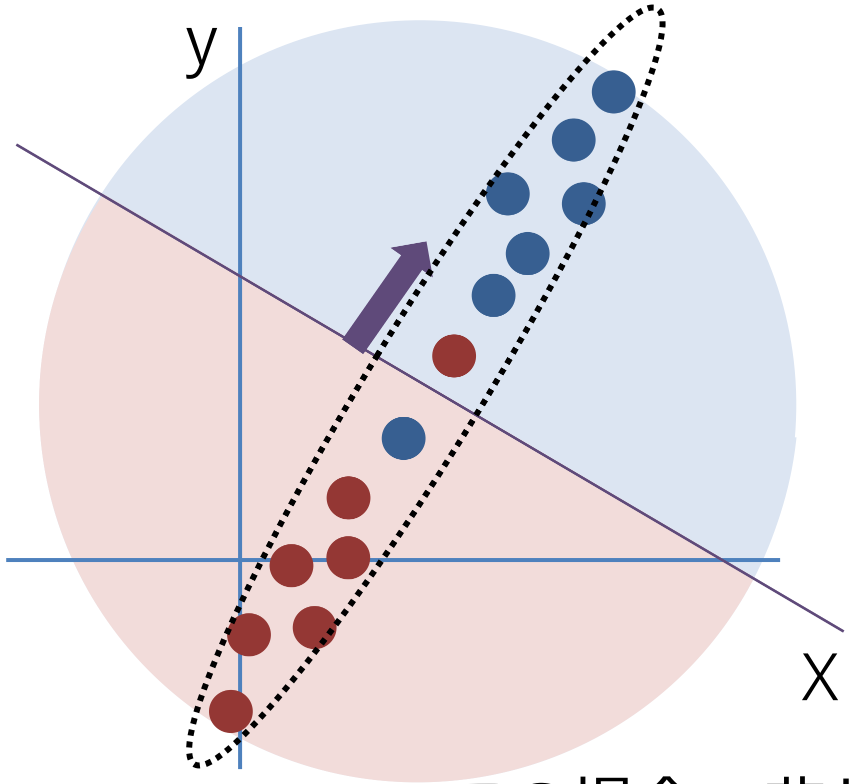
$$y = f(\boldsymbol{w}^T \boldsymbol{x})$$

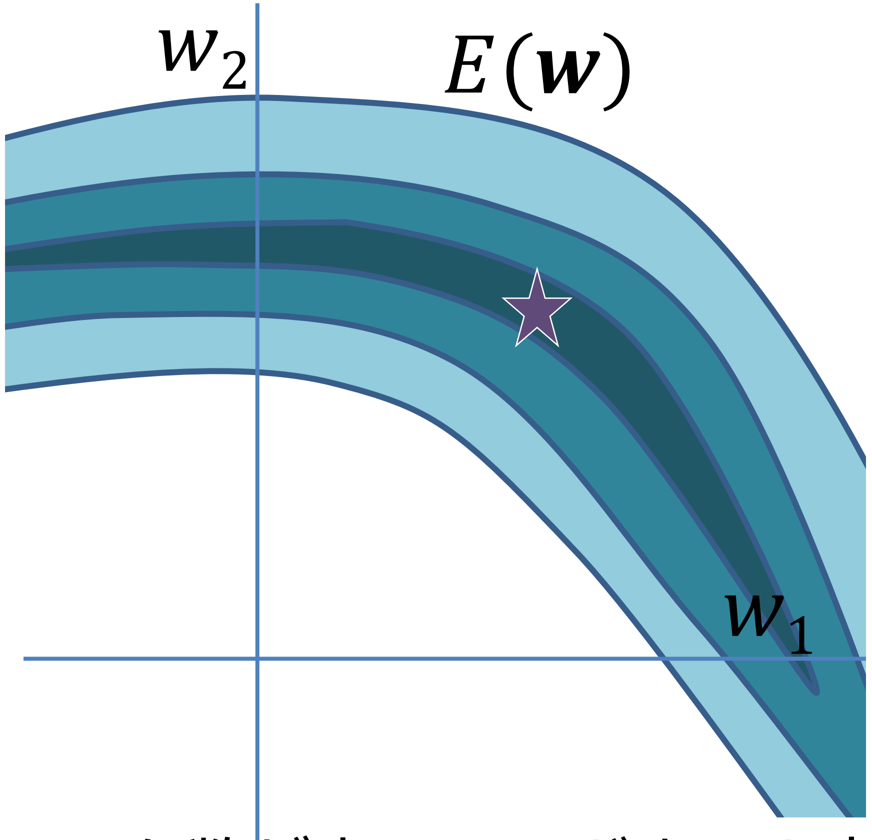
$$f(z) = \frac{1}{1 + \exp(-z)}$$

§ L1正則化

幾つかの問題で不定解になることがある。

例）互いに従属なパラメタがあるとき，不定解を生じる。



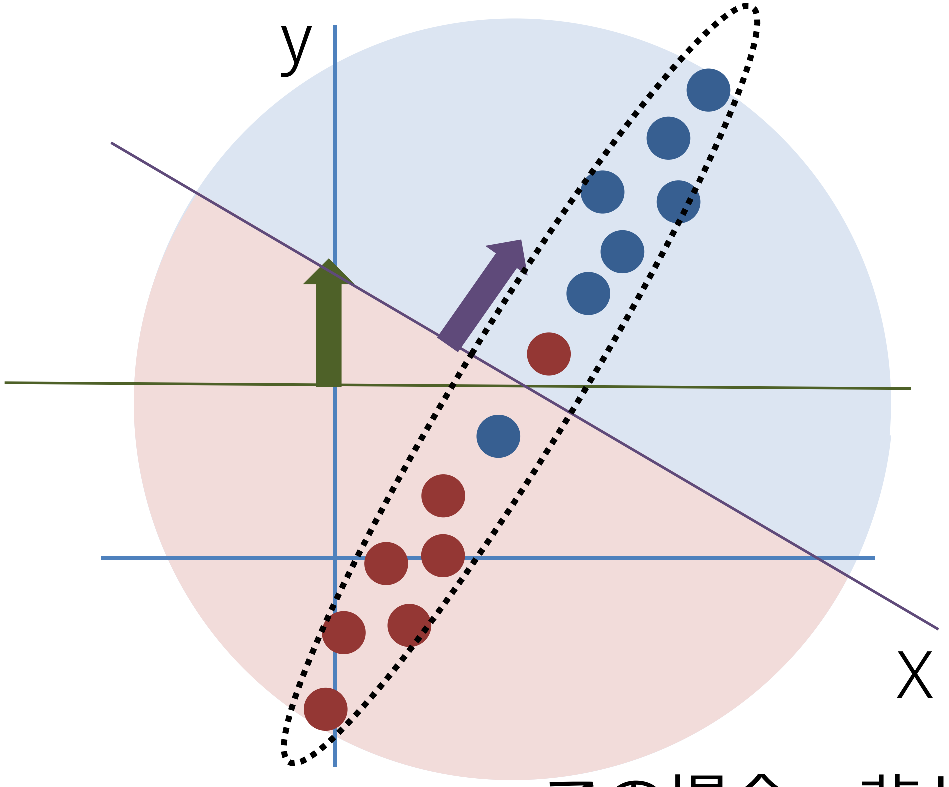


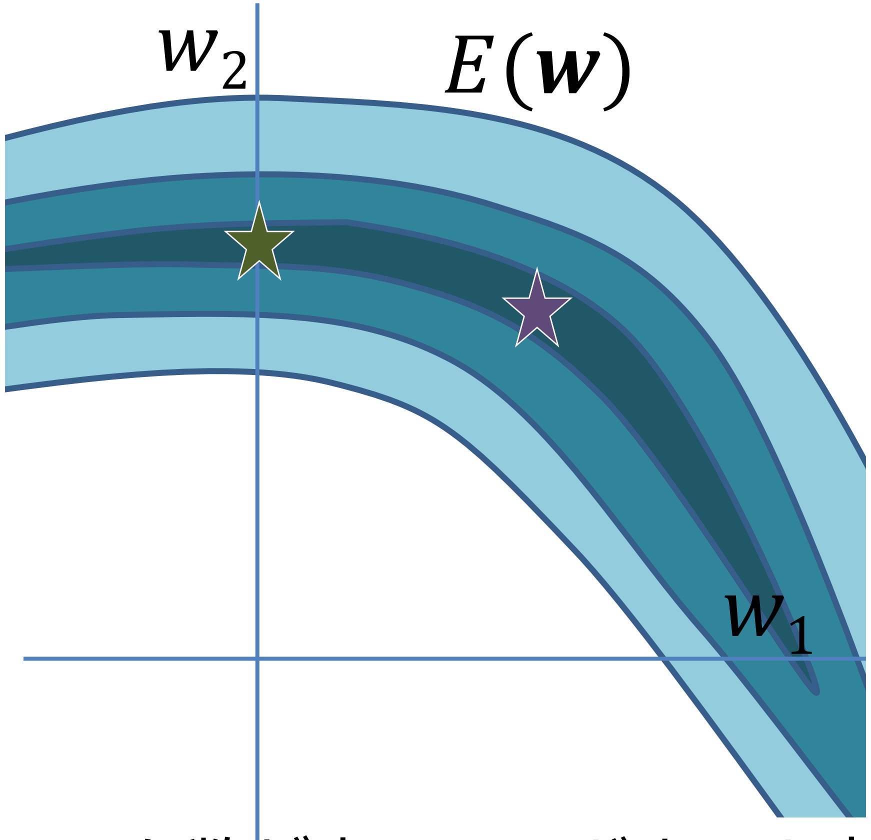
この場合，非ゼロのパラメタ数が少ないほどよいと考える。

L1正則化

幾つかの問題で不定解になることがある。

例）互いに従属なパラメタがあるとき，不定解を生じる。



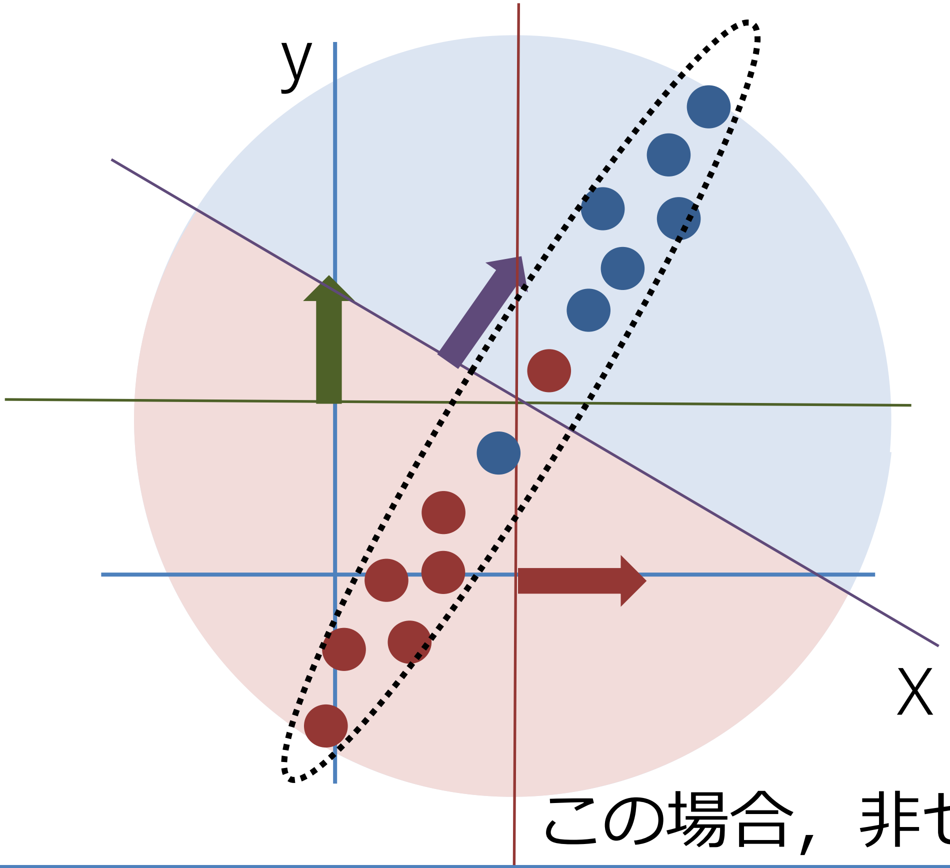


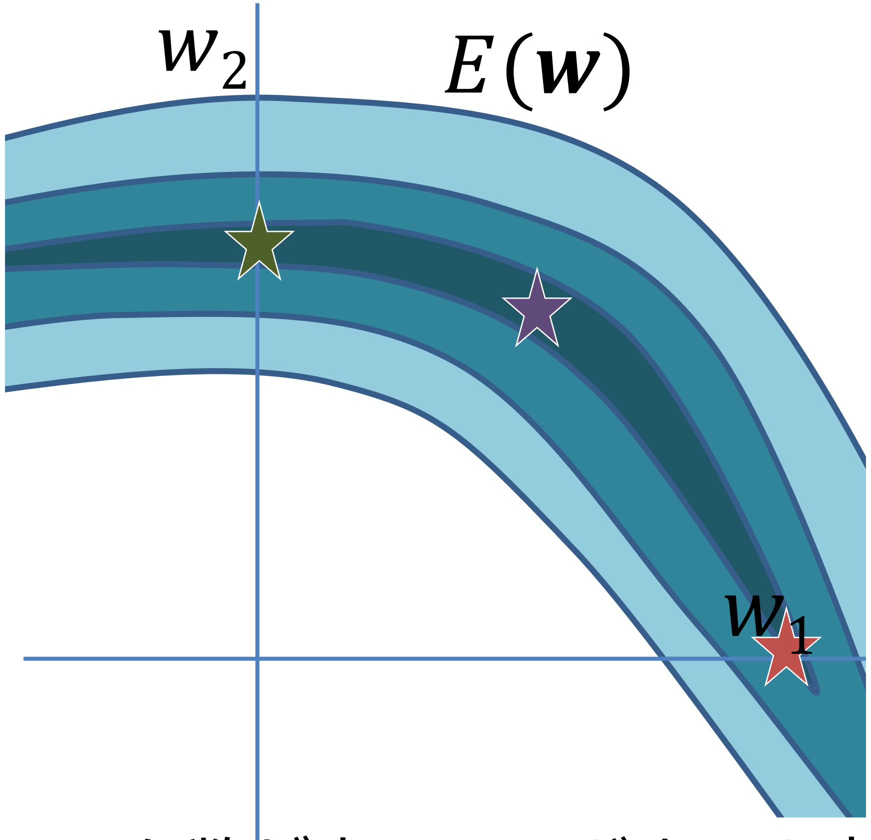
この場合，非ゼロのパラメタ数が少ないほどよいと考える。

L1正則化

幾つかの問題で不定解になることがある。

例）互いに従属なパラメタがあるとき，不定解を生じる。



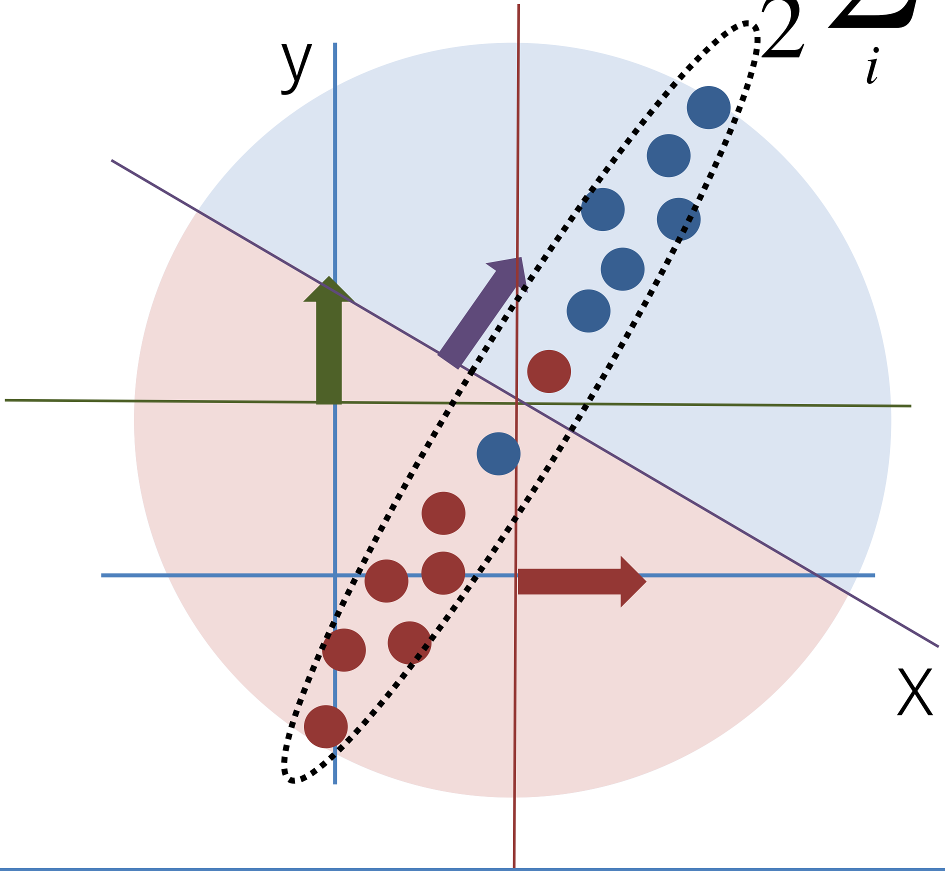


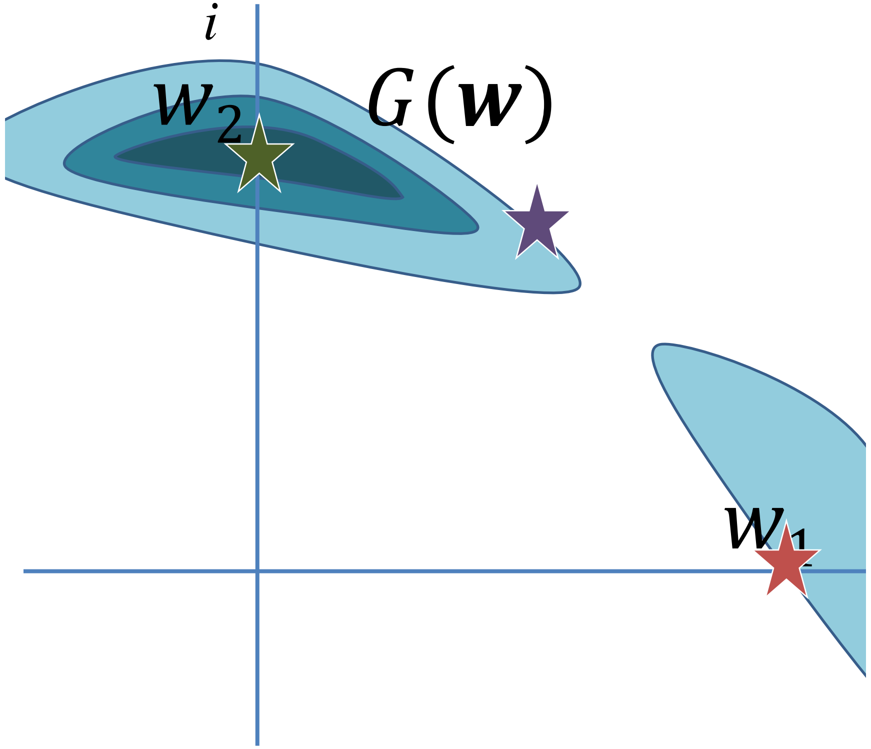
この場合，非ゼロのパラメタ数が少ないほどよいと考える。

L1正則化

L1 Norm

$$G = \frac{1}{2}\sum_i (o_i^0 - d_i)^2 + \sum_i |w_i|$$

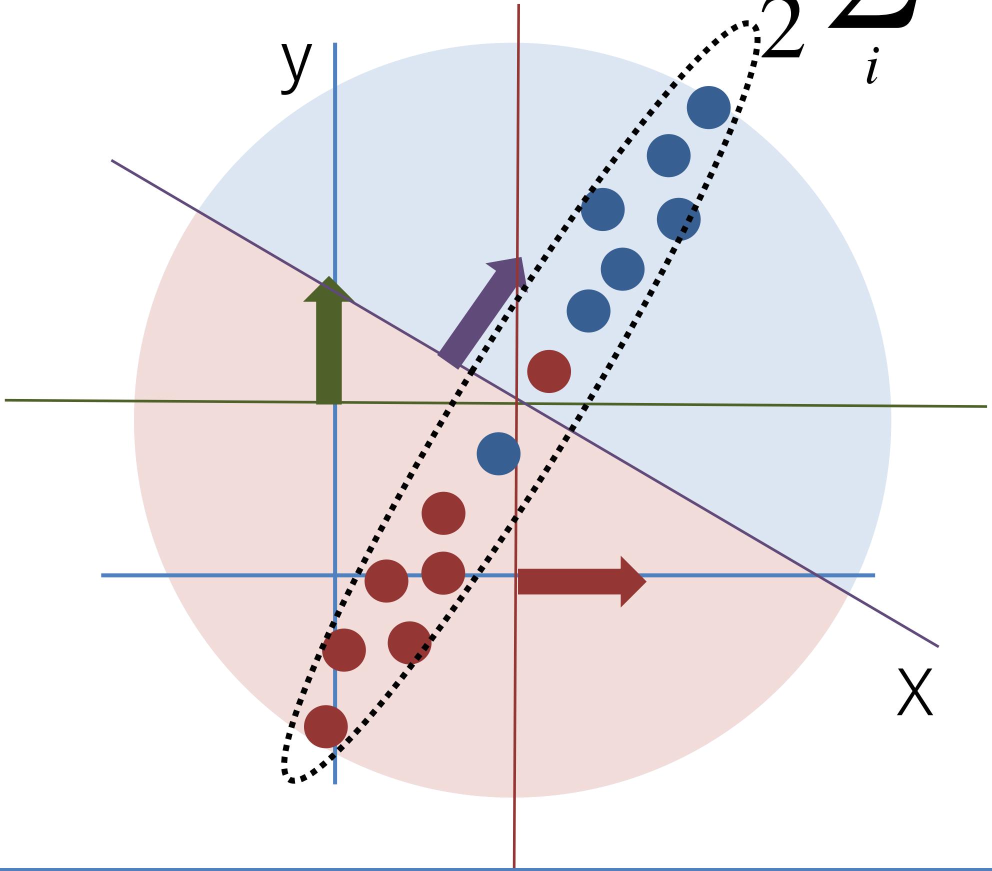


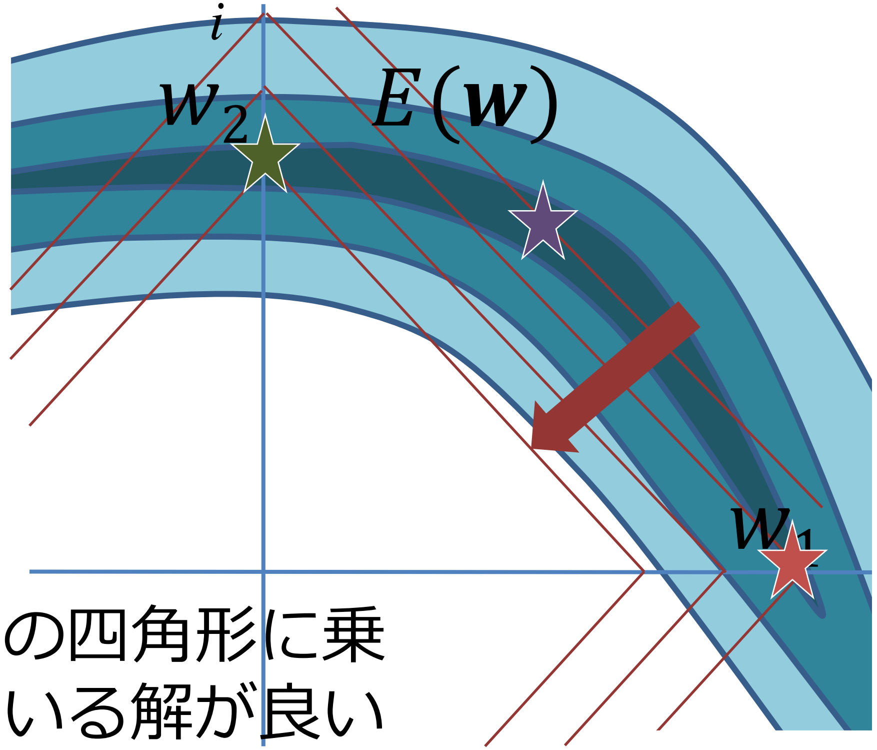


L1正則化

L1 Norm

$$G = \frac{1}{2}\sum_i (o_i^0 - d_i)^2 + \sum_i |w_i|$$

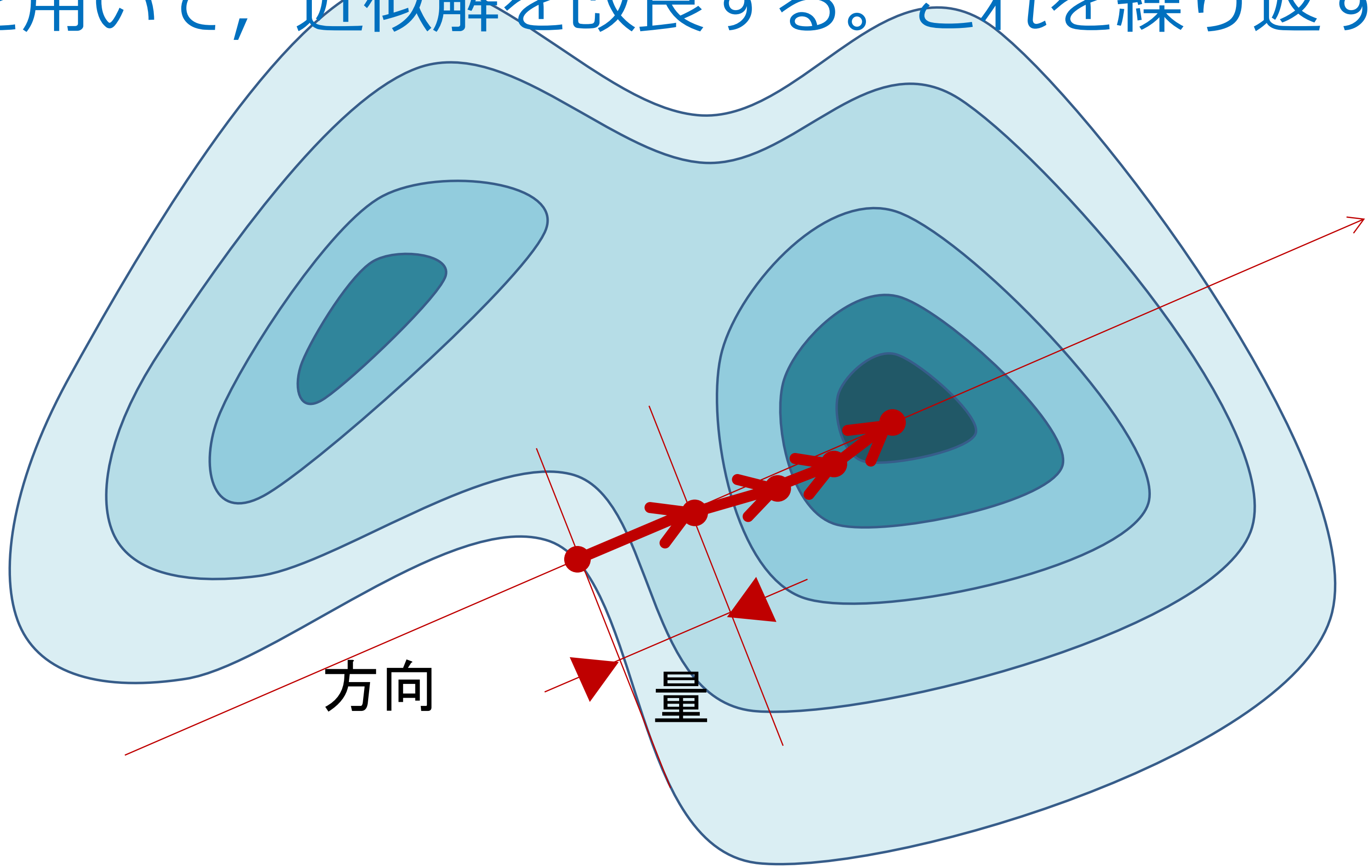




内側の四角形に乗っている解が良い

§ 勾配法のまとめ

勾配法：まずある近似値を定め，この近似解の近傍の関数の勾配（等）を用いて，近似解を改良する。これを繰り返す。

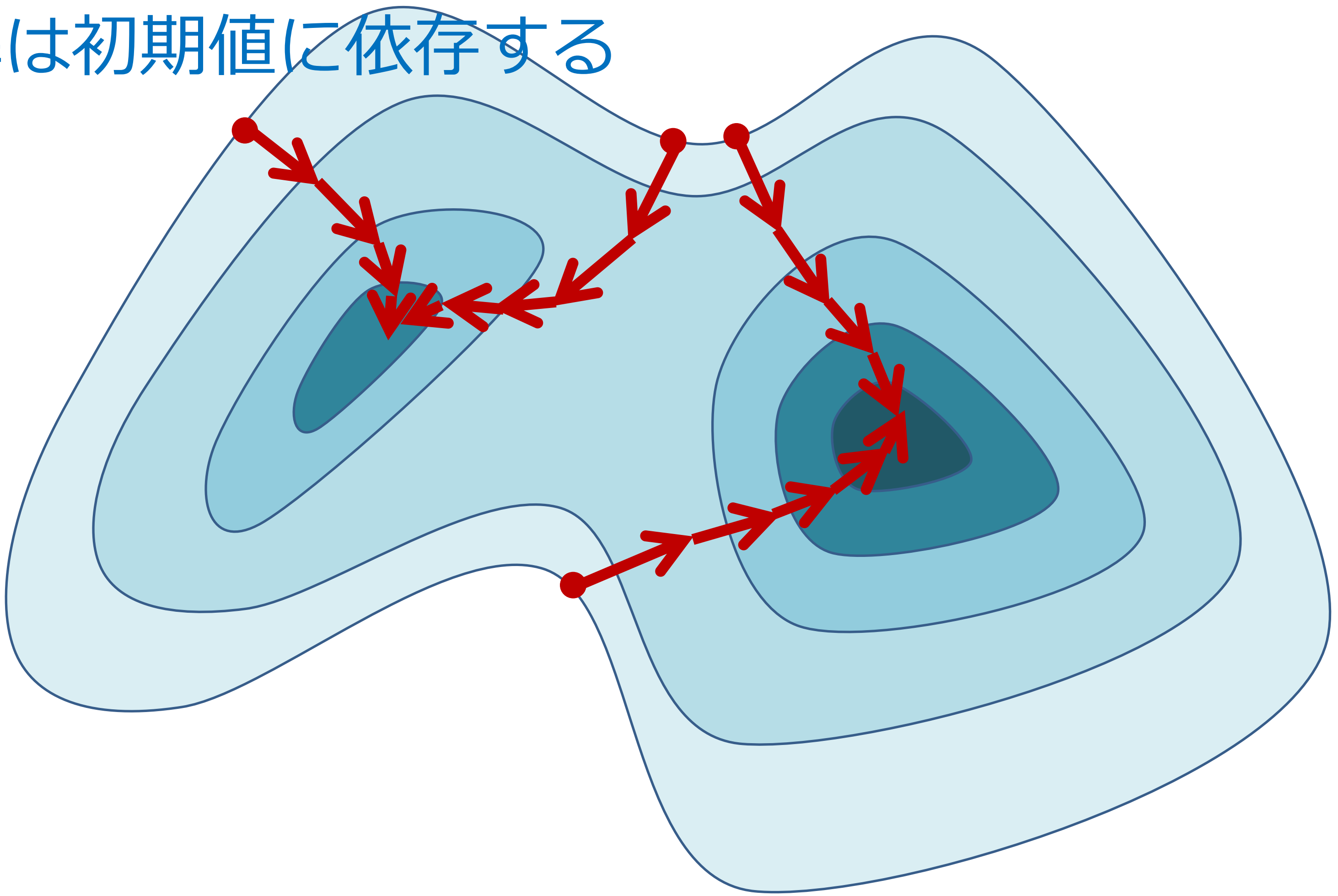


種々の勾配法の関係

	修正ベクトルの方向	量
最急降下法	傾き	予め定めた量など
（最適勾配法）	傾き	最適量（ラインサーチなど）
共役勾配法	共役性	最適量（共役性）
ニュートン法	傾き＋ヘシアン	最適量（２次系の仮定）

勾配法と初期値問題

勾配法においては，
得られる解は初期値に依存する



§ 演習問題

1. 関数

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T \begin{pmatrix} 6 & 1 \\ 1 & 2 \end{pmatrix} \boldsymbol{x} - \begin{pmatrix} 2 \\ 1 \end{pmatrix}^T \boldsymbol{x}$$

の　最小値を，

共役勾配法，ニュートン法のそれぞれで求めよ。

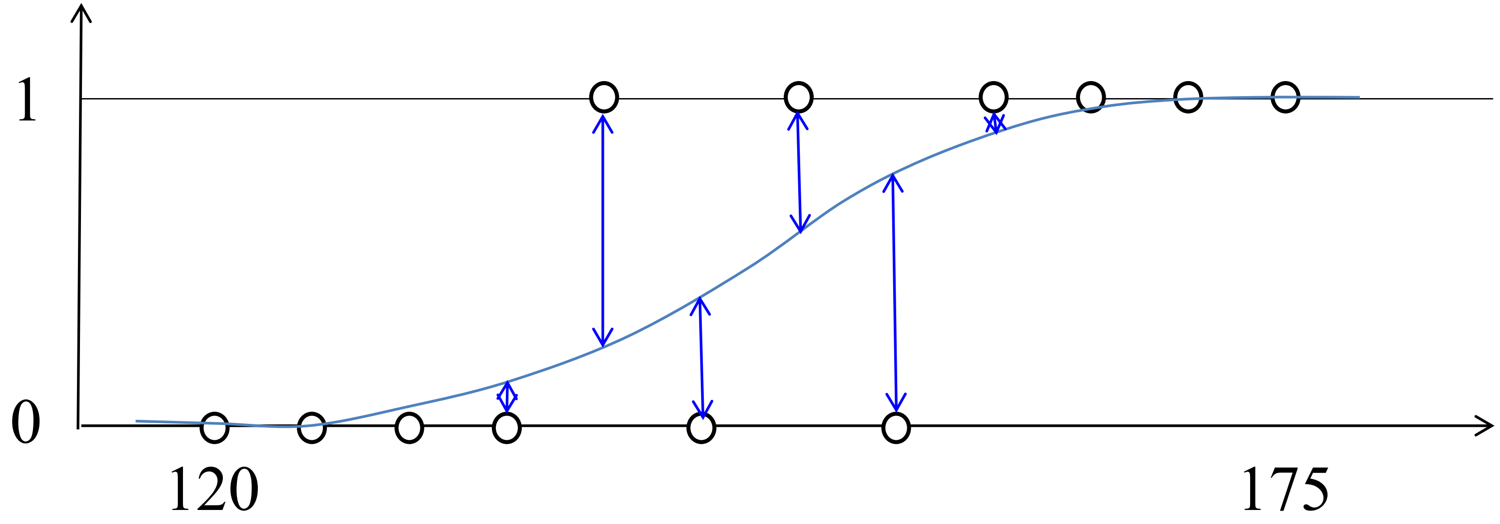
演習問題

2. 下記の学習データに対し，ロジスティック関数（シグモイド関数）を当てはめることを考える。

（1.20，0）（1.25，0）（1.30，0）（1.35，0）（1.40，1）（1.45，0）
（1.50，1）（1.55，0）（1.60，1）（1.65，1）（1.70，1）（1.75，1）

(1)二乗誤差最小化するパラメタを，勾配法を用いて求めよ。

(2)二乗誤差最小化するパラメタを，ニュートン法を用いて求めよ。



演習問題

3.　勾配法における，近似解の更新方法（更新ベクトルの定め方,学習率の定め方 ）には様々な方法がある。

　授業で触れなかったものについて，どのような方法があるか各自調査せよ。

　また，そのいくつかを実装し，先のロジスティック回帰の問題（演習問題2）に適用して，収束の様子を比べよ。