Pattern Recognition and Machine Learning

Junya Yamaguchi

2018 年 2 月 28 日

4.2 確率的生成モデル

$$\begin{aligned} p(\mathcal{C}_1 \,|\, \mathbf{x}) &= \frac{p(\mathbf{x} \,|\, \mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x} \,|\, \mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x} \,|\, \mathcal{C}_2)p(\mathcal{C}_2)} \\ &= \frac{1}{1 + \exp(-a)} \\ &= \sigma(a) \end{aligned} \tag{4.57}$$

ここで a を

$$a = \ln \frac{p(\mathbf{x} \,|\, \mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x} \,|\, \mathcal{C}_2)p(\mathcal{C}_2)} \tag{4.58}$$

と定義した.

■シグモイド関数　$\sigma(\cdot)$ はシグモイド関数 *(logistic sigmoid function)* と呼ばれる関数.

$$\sigma(z) = \frac{1}{1 + \exp(-z)} \tag{4.59}$$

- 入力 z を区間 $[0, 1]$ に押し込んで出力するので，なんらかの計算を行った最後で確率を出力したい時によく出てくる．(e.g. ロジスティック回帰，ニューラルネットワーク)
- 対称性がある

$$\sigma(-z) = 1 - \sigma(z) \tag{4.60}$$

- 逆関数は ロジット関数 *(logit function)* として知られている（統計学の世界ではリンク関数，i.e. 活性化関数の逆関数, を考えることが多い）.

$$z = \mathrm{logit}(\sigma) = \ln\left(\frac{\sigma}{1 - \sigma}\right) \tag{4.61}$$

ロジスティック関数を他クラスに拡張したものは ソフトマックス関数 *(softmax function)* として知られている．

$$\mathbf{z} \in \mathbb{R}^M,\ \mathbf{s}(\mathbf{z}) = \begin{pmatrix} s_1(\mathbf{z}) \\ \vdots \\ s_M(\mathbf{z}) \end{pmatrix}$$

$$s_i(\mathbf{z}) = \frac{\exp(\mathbf{z}_i)}{\sum_j \exp(\mathbf{z}_j)}$$

$$p(\mathcal{C}_k \,|\, \mathbf{x}) = \frac{p(\mathbf{x} \,|\, \mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x} \,|\, \mathcal{C}_j)p(\mathcal{C}_j)} = s_k(\ln p(\mathbf{x} \,|\, \mathcal{C}_k)p(\mathcal{C}_k)) \tag{4.62}$$

4.2.1 連続値入力

クラスの条件付き確率 $p(\mathbf{x} \,|\, \mathcal{C}_k)$ がガウス分布であると仮定して，事後確率がどうなるかを考える．すべてのクラスが同じ共分散行列を共有していると仮定する．

$$p(\mathbf{x} \,|\, \mathcal{C}_k) = (2\pi)^{-\frac{D}{2}} |\mathbf{\Sigma}|^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \mathbf{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) \right\} \tag{4.64}$$

2 クラス

$$p(\mathcal{C}_1 \,|\, \mathbf{x}) = \sigma(\mathbf{x}^\top \mathbf{x} + w_0) \tag{4.63}$$

$$\mathbf{w} = \mathbf{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \tag{4.64}$$

$$w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^\top \mathbf{\Sigma}^{-1} \boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^\top \mathbf{\Sigma}^{-1} \boldsymbol{\mu}_2 + \ln \frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)} \tag{4.65}$$

- シグモイド関数の中身が $\mathbf{x}$ の線形関数である（i.e. 境界面 $\sigma(\cdot) = 0.5 \iff \mathbf{x}^\top \mathbf{x} + w_0 = 0$ が 入力空間において線形）．
- 事前分布バイアスパラメータ w_0 にしか影響しない（i.e. 境界面を平行移動させるだけ）．

K クラス

$$\begin{aligned} p(\mathcal{C}_k \,|\, \mathbf{x}) &= \mathrm{softmax}_k(a_k) \\ &= \mathrm{softmax}_k(\mathbf{w}_k^\top \mathbf{x} + w_{k0}) \end{aligned} \tag{4.66}$$

$$\mathbf{w}_k = \mathbf{\Sigma}^{-1} \boldsymbol{\mu}_k \tag{4.67}$$

$$w_{k0} = -\frac{1}{2}\boldsymbol{\mu}_k^\top \mathbf{\Sigma}^{-1} \boldsymbol{\mu}_k + \ln p(\mathcal{C}_k) \tag{4.68}$$

- a_k が $\mathbf{x}$ の線形関数となる．
- 誤分類率をロス関数とした場合（正解クラスに依らず，分類に成功したとき/失敗した時に課せられるペナルティが全て同じ場合）の境界面は，やはり $\mathbf{x}$ の線形関数．

4.2 最尤解

クラスの条件付き確率 $p(\mathbf{x}\,|\,\mathcal{C}_k)$ にパラメトリックな関数を定めると，クラスの事前確率 $p(\mathcal{C}_k)$ とともに，パラメータの値を最尤法を用いて決めることができる．

2 クラス

$\mathbf{x}_n \in \mathbb{R}^M$, $t_n \in \{0,1\}$ な訓練集合 $\mathcal{D} \stackrel{\text{def}}{=} \{\mathbf{x}_n, t_n\}_{n=1}^N$ が与えられている．

- $t_n = 0 \implies \mathbf{x}_n$ はクラス $\mathcal{C}_1$
- $t_n = 1 \implies \mathbf{x}_n$ はクラス $\mathcal{C}_2$

ここで，クラスの事前分布を $p(\mathcal{C}_1) = \pi$, $p(\mathcal{C}_2) = 1 - \pi$ とすると，尤度関数は

$$\begin{aligned} p(\mathcal{D}\,|\,\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) &= \prod_n \{\pi p(\mathbf{x}\,|\,\mathcal{C}_1)\}^{t_n} \{(1-\pi) p(\mathbf{x}\,|\,\mathcal{C}_2)\}^{1-t_n} \\ &= \prod_n \{\pi \mathcal{N}(\mathbf{x}_n\,|\,\boldsymbol{\mu}_1, \boldsymbol{\Sigma})\}^{t_n} \{(1-\pi) \mathcal{N}(\mathbf{x}_n\,|\,\boldsymbol{\mu}_2, \boldsymbol{\Sigma})\}^{1-t_n} \end{aligned} \tag{4.71}$$

■π の最尤推定　対数尤度 $\ln L(\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma})$ で π が関係しているのは

$$= \sum_n \{t_n \ln \pi + (1 - t_n) \ln(1-\pi)\} \tag{4.72}$$

なので，$\frac{\partial \ln L}{\partial \pi} = 0$ を解くと

$$\pi = \frac{N_1}{N_1 + N_2} \tag{4.73}$$

が得られる．ただし，N_1, N_2 はそれぞれ $\mathcal{C}_1, \mathcal{C}_2$ に属するデータの数．

■$\boldsymbol{\mu}_1$ の最尤推定　$\boldsymbol{\mu}_1$ が関係しているのは

$$\sum_n t_n \ln \mathcal{N}(\mathbf{x}_n\,|\,\boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = -\frac{1}{2} \sum_n t_n (\mathbf{x}_n - \boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_1) + \text{const.} \tag{4.74}$$

なので，$\frac{\partial \ln L}{\partial \boldsymbol{\mu}_1} = 0$ を解くと

$$\boldsymbol{\mu}_1 = \frac{1}{N_1} \sum_n t_n \mathbf{x}_n \tag{4.75}$$

$\boldsymbol{\mu}_2$ についても同様に解ける．

$$\boldsymbol{\mu}_2 = \frac{1}{N_2} \sum_n (1 - t_n) \mathbf{x}_n \tag{4.76}$$

■共分散行列の最尤推定　$\boldsymbol{\Sigma}$ が関係する項は以下の通り．

$$\begin{aligned} &-\frac{1}{2} \sum_n t_n \ln|\boldsymbol{\Sigma}| - \frac{1}{2} \sum_n t_n (\mathbf{x}_n - \boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_1) \\ &-\frac{1}{2} \sum_n (1 - t_n) \ln|\boldsymbol{\Sigma}| - \frac{1}{2} \sum_n (1 - t_n) (\mathbf{x}_n - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_2) \\ &= -\frac{N}{2} \ln|\boldsymbol{\Sigma}| - \frac{N}{2} \operatorname{Tr}\{\boldsymbol{\Sigma}^{-1} \mathbf{S}\} \end{aligned} \tag{4.77}$$

ここで,

$$\mathbf{S} = \frac{N_1}{N}\mathbf{S}_1 + \frac{N_2}{N}\mathbf{S}_2 \tag{4.78}$$

$$\mathbf{S}_1 = \frac{1}{N_1}\sum_{n\in\mathcal{C}_1}(\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^\top \tag{4.79}$$

$$\mathbf{S}_2 = \frac{1}{N_2}\sum_{n\in\mathcal{C}_2}(\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^\top \tag{4.80}$$

■やってみよう　式 (4.77) の微分を 0 と置くことで共分散行列の最尤推定をする．ただし，以下の行列の微分に関する公式を使って良い．

$$\frac{\partial \operatorname{tr}(\mathbf{X^{-1}A})}{\partial \mathbf{X}} = -(\mathbf{X}^{-1})^\top \mathbf{A}^\top (\mathbf{X}^{-1})^\top$$

$$\frac{\partial \ln |a\mathbf{X}|}{\partial \mathbf{X}} = (\mathbf{X}^{-1})^\top$$

4.2.3 離散特徴

入力が $\mathbf{x} = (x_1, \ldots, x_D)$, $x_i \in \{0, 1\}$ であるときを考える．こいつには 2^D 通りの実現値が考えられるので，$p(\mathbf{x})$ を表現する確率分布にはパラメータが $2^D - 1$ 個必要．

	$\mathbf{x}$	$p(\mathbf{x})$
1	$(0, 0, \ldots, 0)$	μ_1
	$\vdots$	$\vdots$
2^D	$(1, 1, \ldots, 1)$	$1 - \sum_{i=1}^{2^D-1} \mu_i$

ここでナイーブベイズ *(naive Bayes)*，つまり各要素は他の要素に影響しないという独立性を仮定すると，必要なパラメータがぐんと少なくなる．

$$p(\mathbf{x}) = p(x_1, x_2, \ldots, x_D) = p(x_1)p(x_2)\ldots p(x_D)$$

この仮定のもとで，クラスの条件付き確率分布は次のように与えられる．

$$p(\mathbf{x} \,|\, \mathcal{C}_k) = \prod_i \mu_{ki}^{x_i}(1 - \mu_{ki})^{1-x_i} \tag{4.81}$$

$$p(\mathcal{C}_k \,|\, \mathbf{x}) = \frac{\exp(a_k)}{\sum_j \exp(a_j)} \tag{4.62}$$

式 (4.62) に当てはめてみると，クラスの事後分布は

$$a_k = \sum_i^D \{x_i \ln \mu_{ki} + (1 - x_i) \ln(1 - \mu_{ki})\} + \ln p(\mathcal{C}_k) \tag{4.82}$$

で与えられる．やはりこれも x_i の線形関数である．

4.2.4 指数型分布族

ここまで，「クラスの条件付き確率 $p(\mathbf{x} \,|\, \mathcal{C}_k)$ を〜としてみる」⇒「すごい！ 事後分布 $p(\mathcal{C}_k \,|\, \mathbf{x})$ は $\mathbf{x}$ の線形関数にソフトマックス (シグモイド) かましたやつだ！」という茶番を繰り返してきたが，これはある条件を満たせば必ず成り立つ．

次の条件を満たせば，クラスの事後確率 $p(\mathcal{C}_k \,|\, \mathbf{x})$ が，ロジスティック関数またはソフトマックス関数の一般化線形モデル (e.g. $\sigma(A\mathbf{x} + b)$) で与えられる．

- クラスの条件付き確率 $p(\mathbf{x} \,|\, \mathcal{C}_k)$ が指数型分布族である．

$p(\mathbf{x}\,|\,\mathcal{C}_k)$ が指数型分布族なら，*natural (calibrated) parameter* $\boldsymbol{\lambda}_k$ と，適当な関数 $\boldsymbol{\phi}, h, A$ を用いて次のように表すことができる．

$$p(\mathbf{x}\,|\,\mathcal{C}_k) = p(\mathbf{x}\,|\,\boldsymbol{\lambda}_k) = h(\mathbf{x}) \exp\bigl\{\boldsymbol{\lambda}_k^\top \boldsymbol{\phi}(\mathbf{x}) + A(\boldsymbol{\lambda}_k)\bigr\} \tag{4.83}$$

ここで，$\boldsymbol{\phi}(\mathbf{x}) = \mathbf{x}$ となるような分布に注目してみる．尺度パラメータ s を導入すると，$p(\mathbf{x}\,|\,\mathcal{C}_k)$ をさらに以下の式で記述される指数分布族に絞り込むことができる．

$$p(\mathbf{x}\,|\,\mathcal{C}_k) = p(\mathbf{x}\,|\,\boldsymbol{\lambda}_k, s) = \frac{1}{s} h\left(\frac{1}{s}\mathbf{x}\right) \exp\left\{\frac{1}{s}\boldsymbol{\lambda}_k^\top \mathbf{x} + A(\boldsymbol{\lambda}_k)\right\} \tag{4.84}$$

■やってみよう　2 クラス分類の場合，クラスの事後分布は式 (4.57), (4.58) のように定義できることを既に示した．これに 式 (4.84) の結果を代入すると，$\mathbf{x}$ の線形な関数で a が与えられる（クラスの事後分布は一般化線形モデルである）ことがわかる．

$$p(\mathcal{C}_1\,|\,\mathbf{x}) = \sigma(a) \tag{4.57}$$

$$a = \ln \frac{p(\mathbf{x}\,|\,\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}\,|\,\mathcal{C}_2)p(\mathcal{C}_2)} \tag{4.58}$$

4.3 確率的識別モデル

特殊な生成モデル（指数分布族をクラスの条件付き確率に使用した生成モデル）を最尤推定つかってパラメータ求めた場合は，$p(\mathcal{C}_k \mid \mathbf{x})$ は必ず一般化線形モデルになるんだなぁ．みつお．

⇒はじめから事後確率 $p(\mathcal{C}_k \mid \mathbf{x})$ を一般化線形モデルでモデリングして，そのパラメータを推定すればいいのでは．生成モデルじゃなくなるけど，クラス分類するだけなら十分ちゃう？

$$p(\mathcal{C}_1 \mid \mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + w_0)$$
$$p(\mathcal{C}_k \mid \mathbf{x}) = \mathrm{softmax}_k(\mathbf{w}^\top \mathbf{x} + w_0)$$

事後確率 $p(\mathcal{C}_k \mid \mathbf{x})$ を一般化線形モデルの関数形式で仮定し，最尤法によるパラメータ推定を行う方法として，反復重み付け最小二乗 *(iterative reweighted squares; IRLS)* が知られている．

識別モデル
: 入力からクラスへの関数 $f : \mathcal{X} \to \mathcal{C}$ を何らかの形で表現する．

確率的識別モデル
: 入力 $\mathbf{x}$ が与えられた時，$\mathbf{x}$ が各クラスに属する確率（事後確率） $p(\mathcal{C}_k \mid \mathbf{x})$ を何らかの形で実現する．

生成モデル
: クラスの事前分布 $p(\mathcal{C}_k)$ とクラスの条件付き確率 $p(\mathbf{x} \mid \mathcal{C}_k)$ を何らかの形で実現する．$p(\mathcal{C}_k \mid \mathbf{x})$ はベイズの定理を使えば勝手に出てくるので，そいつで予測できる．生成モデルがすごいのは，

$$p(\mathbf{x}) = \int_{\mathcal{C}} p(\mathbf{x} \mid \mathcal{C}_k) p(\mathcal{C}_k) \, \mathrm{d}\mathcal{C}_k$$

を使えば $\mathbf{x}$ を人工的にサンプリングできる点．

4.3.1 固定基底関数

「入力 $\mathbf{x}$ の線形関数でほんとに分類できんの？」という煽りへの対策として，入力に基底関数をかました $\boldsymbol{\phi} = \boldsymbol{\phi}(\mathbf{x})$ に一般化線形モデルを適応することを考えます．

$$p(\mathcal{C}_1 \mid \mathbf{x}) = \sigma(\mathbf{w}^\top \boldsymbol{\phi})$$
$$p(\mathcal{C}_k \mid \mathbf{x}) = \mathrm{softmax}_k(\mathbf{w}^\top \boldsymbol{\phi})$$

- $\mathbf{x}$ を入力ベクトル，$\boldsymbol{\phi}$ を特徴ベクトルと呼ぶ．
- $\mathbf{x}$ が存在する空間を入力空間 *(input space)*，ϕ が存在する空間を特徴空間 *(feature space)* と呼ぶ．
- 入力空間から特徴空間 $\mathbb{R}^M$ への写像 $\boldsymbol{\phi}(\mathbf{x})$ は，M 個の基底関数 *(basis function)* と呼ばれる写像 $\phi_i(\mathbf{x})$ で構成される．

$$\boldsymbol{\phi}(\mathbf{x}) = \begin{pmatrix} \phi_1(\mathbf{x}) \\ \vdots \\ \phi_M(\mathbf{x}) \end{pmatrix} \in \mathbb{R}^M$$

- 一般に基底関数は非線形変換を行うが，バイアスに相当するパラメータを作るために $\phi_1(\mathbf{x}) = 1$ とすることが多い．
- 入力空間の任意の 2 点を受け取り，特徴空間での内積を計算する関数をカーネル関数 *(kernel function)* と呼ぶ．

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle$$

- 基底関数を固定することの問題点：次元の呪い *(the curse of dimentionality)*
  - 2 値の特徴を 10 個抽出するような基底関数を構成 $\rightarrow \boldsymbol{\phi}$ は 2^{10} 通りの可能性
  - この特徴空間をいい感じに埋めるには，少なくとも 1024 個のデータがほしい（例えば，データが 100 個しかないと特徴空間はスッカスカ）
  - つまり**特徴数を固定すると，それの指数個分のデータが必要となる**

4.3.2 ロジスティック回帰

2 クラス分類問題における一般化線形モデルを考える．

$$p(\mathcal{C}_1 \,|\, \boldsymbol{\phi}) = y(\boldsymbol{\phi}) = \sigma(\mathbf{w}^\top \boldsymbol{\phi}) \tag{4.87}$$

- 「任意の $\mathbf{x}$ がクラス $\mathcal{C}_k$ に属する確率」をパラメトリックに（形決め打ち方式で）モデリングしている．
- なぜこの形を考えるかというと，**次の条件を満たす**の生成モデルは，必ず上記のような一般化線形モデルが得られるから．
  1. パラメトリックなモデルを事前分布 $p(\mathcal{C}_k)$ に使用
  2. 指数分布族をクラスの条件付き確率 $p(\mathbf{x} \,|\, \mathcal{C}_k)$ に使用
  3. 最尤推定によって $p(\mathcal{C}_k)$ と $p(\mathbf{x} \,|\, \mathcal{C}_k)$ のパラメータを推定
- ロジスティック回帰とよばれるモデリング方法だが，**回帰というよりは分類のためのモデル**である．
  - 「クラスに属する確率」を回帰している
- 得られる結果（事後分布）は一緒だが，生成モデルの場合と比べると**必要なパラメータの数が少ない**．
  - 生成モデル：ガウス分布の平均に対して $2 \times M$ 個，共有共分散行列に対して $M(M+1)/2$ 個の計 $(M^2 + 5M)/2$ 個のパラメータ
  - ロジスティック回帰：重みパラメータ M 個

■最尤推定　データ集合 $\{\boldsymbol{\phi}_n = \boldsymbol{\phi}(\mathbf{x}_n), t_n\}_{n=1}^N, t_n \in \{0,1\}$ に対する尤度関数は

$$p(\mathbf{t} \mid \mathbf{w}) = \prod_n y_n^{t_n} (1 - y_n)^{1-t_n} \tag{4.89}$$

で与えられる．ここで，$\mathbf{t} = (t_1, \ldots, t_n)^\top$ であり，$y_n = \sigma(\mathbf{w}^\top \boldsymbol{\phi}_n) = p(\mathcal{C}_1 \mid \boldsymbol{\phi}_n)$.

誤差関数として，負の対数尤度を設定する．

$$E(\mathbf{w}) = -\ln p(\mathbf{t} \mid \mathbf{w}) = -\sum_n \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\} \tag{4.90}$$

- 実はこの $E(\mathbf{w})$ は，一般に交差エントロピー誤差関数 *(cross-entropy error function)* と呼ばれる誤差関数

■やってみよう　$E(\mathbf{w})$ の $\mathbf{w}$ に関する微分を計算する．

$$\frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1 - \sigma(z)) \tag{4.88}$$

の結果を用いると，

$$\frac{\partial\, y_n}{\partial\, \mathbf{w}} = \frac{\partial\, \sigma(\mathbf{w}^\top \boldsymbol{\phi}_n)}{\partial\, \mathbf{w}} =$$

$$\frac{\partial \ln y_n}{\partial\, \mathbf{w}} = \frac{1}{y_n} \frac{\partial\, y_n}{\partial\, \mathbf{w}} =$$

$$\frac{\partial \ln(1 - y_n)}{\partial\, \mathbf{w}} = \frac{-1}{(1 - y_n)} \frac{\partial\, y_n}{\partial\, \mathbf{w}} =$$

なので，結局

$$\frac{\partial\, E(\mathbf{w})}{\partial\, \mathbf{w}} =$$

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} \underbrace{(y_n - t_n)}_{\text{誤差}} \boldsymbol{\phi}_n \tag{4.91}$$

- データ n の勾配の寄与は，予測の誤差 $y_n - t_n$ と特徴ベクトル $\boldsymbol{\phi}_n$ の積で与えられる．
- 特徴変換を行う線形回帰モデルにおける，二乗和誤差関数の勾配と同じ．
- オンライン学習へ簡単に応用できる．
  - $\nabla E_n = (y_n - t_n)\boldsymbol{\phi}_n$ を使う $\Rightarrow$ 確率的勾配降下法 *(stochastic gradient descent; SGD)*
  - $\nabla E_{\mathbf{b}} = \sum_{n \in \mathbf{b}} (y_n - t_n)\boldsymbol{\phi}_n$ を使う $\Rightarrow$ ミニバッチ学習 *(mini-batch gradient descent)*
- **線形分離が可能な場合，過学習が起きる**ことが知られている．
  - $\mathbf{w}$ を大きくすればするほど y_n を t_n に近づけることができてしまう
  - $\|\mathbf{w}\| \to \infty$ のとき，新しい入力 $\boldsymbol{\phi}^\star$ に対する事後確率が極端な値しか取らない．

$$p(\mathcal{C}_1 \,|\, \boldsymbol{\phi}^\star) = \sigma(\mathbf{w}^\top \boldsymbol{\phi}^\star) = \begin{cases} \sigma(\infty) & = 1 \\ \sigma(-\infty) & = 0 \end{cases}$$

- 過学習は正則化項を加えたコスト関数を用いると回避できる．

$$C(\mathbf{w}) = E(\mathbf{w}) + \frac{1}{2}\lambda \mathbf{w}^\top \mathbf{w}$$

- これは事前分布 $p(\mathbf{w})$ に $\mathcal{N}(\mathbf{w} \,|\, \mathbf{0}, \lambda^{-1}\mathbf{I})$ を用いて MAP 推定したのと同じ．

$$p(\mathbf{w} \,|\, \mathcal{D}) \propto p(\mathcal{D} \,|\, \mathbf{w}) p(\mathbf{w})$$

  - 事後分布の最大化は $\ln p(\mathcal{D} \,|\, \mathbf{w}) p(\mathbf{w})$ を最大化することと同じ．

$$\begin{aligned} \ln p(\mathcal{D} \,|\, \mathbf{w}) p(\mathbf{w}) &= \ln p(\mathcal{D} \,|\, \mathbf{w}) + \ln p(\mathbf{w}) \\ &= -E(\mathbf{w}) - \frac{M}{2}\ln(2\pi) - \frac{1}{2}\ln\left|\lambda^{-1}\mathbf{I}\right| - \frac{\lambda}{2}\mathbf{w}^\top \mathbf{w} \end{aligned}$$

4.3.3 反復重み付け最小二乗

誤差関数の二次微分を利用することでより効率よく勾配降下する方法がニュートン–ラフソン法 *(Newton–Raphson method)* である．

Algorithm 8.1: Newton's method for minimizing a strictly convex function

1 Initialize $\boldsymbol{\theta}_0$;
2 **for** $k = 1, 2, \ldots$ *until convergence* **do**
3 Evaluate $\mathbf{g}_k = \nabla f(\boldsymbol{\theta}_k)$;
4 Evaluate $\mathbf{H}_k = \nabla^2 f(\boldsymbol{\theta}_k)$;
5 Solve $\mathbf{H}_k \mathbf{d}_k = -\mathbf{g}_k$ for $\mathbf{d}_k$;
6 Use line search to find stepsize η_k along $\mathbf{d}_k$;
7 $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \eta_k \mathbf{d}_k$;

皆さんご存知のテイラー展開によると，関数 $f(\mathbf{x})$ の $\hat{\mathbf{x}}$ 周りでの 2 次近似は

$$f(\mathbf{x}) \simeq f(\hat{\mathbf{x}}) + \hat{\mathbf{g}}^\top (\mathbf{x} - \hat{\mathbf{x}}) + \frac{1}{2}(\mathbf{x} - \hat{\mathbf{x}})^\top \hat{\mathbf{H}} (\mathbf{x} - \hat{\mathbf{x}})$$

$$\hat{\mathbf{g}} = \nabla f \big|_{\mathbf{x}=\hat{\mathbf{x}}}$$
$$\hat{\mathbf{H}} = \nabla^2 f \big|_{\mathbf{x}=\hat{\mathbf{x}}}$$

である．さらに簡便な表記に直すと

$$f(\mathbf{x}) \simeq \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{b}^\top \mathbf{x} + \text{const.}$$

$$\mathbf{A} = \frac{1}{2}\hat{\mathbf{H}}$$
$$\mathbf{b} = \hat{\mathbf{g}} - \hat{\mathbf{H}}\hat{\mathbf{x}}$$

となる．最小となる $\mathbf{x}_{min}$ の満たすべき停留条件 $f'(\mathbf{x}) = 0$ より

$$\begin{aligned} 2\mathbf{A}\mathbf{x}_{min} + \mathbf{b} &= \mathbf{0} \\ \mathbf{x}_{min} &= -\frac{1}{2}\mathbf{A}^{-1}\mathbf{b} \\ &= -\hat{\mathbf{H}}^{-1}(\hat{\mathbf{g}} - \hat{\mathbf{H}}\hat{\mathbf{x}}) \\ &= \hat{\mathbf{x}} - \hat{\mathbf{H}}^{-1}\hat{\mathbf{g}} \end{aligned}$$

今まで頻繁に目にしてきた最急降下法が

$$\mathbf{x}_{\text{next}} = \hat{\mathbf{x}} - \hat{\mathbf{g}}$$

で与えられる 1 次微分だけを使った反復最適化だったのに対し，ニュートン–ラフソン法では 2 次微分まで用いることでより効率的に反復最適化することができる．

線形回帰モデル

線形回帰モデルに対して，ニュートン–ラフソン法を用いてみる．

$$\mathbf{w}^{(\text{new})} = \mathbf{w}^{(\text{old})} - \mathbf{H}^{-1}\nabla E(\mathbf{w}) \tag{4.92}$$

線形回帰モデル $y_n = \mathbf{w}^\top \boldsymbol{\phi}_n$ の二乗和誤差関数

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\mathbf{w}^\top \boldsymbol{\phi}_n - t_n)^2$$

における勾配とヘッセ行列はそれぞれ

$$\begin{aligned}
\nabla E(\mathbf{w}) &= \sum_{n=1}^{N}(\mathbf{w}^\top \boldsymbol{\phi}_n - t_n)\boldsymbol{\phi}_n \\
&= \Big(\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_N\Big) \begin{pmatrix} \boldsymbol{\phi}_1^\top \mathbf{w} - t_1 \\ \vdots \\ \boldsymbol{\phi}_N^\top \mathbf{w} - t_N \end{pmatrix} \\
&= \boldsymbol{\Phi}^\top(\boldsymbol{\Phi}\mathbf{w} - \mathbf{t}) \\
&= \boldsymbol{\Phi}^\top\boldsymbol{\Phi}\mathbf{w} - \boldsymbol{\Phi}^\top\mathbf{t} \qquad (4.93) \\
\mathbf{H} &= \sum \boldsymbol{\phi}_n\boldsymbol{\phi}_n^\top \\
&= \Big(\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_N\Big) \begin{pmatrix} \boldsymbol{\phi}_1^\top \\ \vdots \\ \boldsymbol{\phi}_N^\top \end{pmatrix} \\
&= \boldsymbol{\Phi}^\top\boldsymbol{\Phi} \qquad (4.94)
\end{aligned}$$

である．

■やってみよう　$\nabla E(\mathbf{w})$ と $\mathbf{H}$ は，$E(\mathbf{w})$ を行列の形に直してから微分すると簡単に求まる．

■やってみよう　このとき，ニュートン–ラフソン法による $\mathbf{w}$ の更新は次の式で与えられる．