第 2 章

統計学習 (Statistical Learning)

2.1 統計学習って何？

何故そもそも統計なんて勉強するのか？ Figure 2.1 を見てほしい。

X 軸はそれぞれ TV、radio、新聞の広告予算であり、Y 軸はある商品の売上金額である。企業にとって売上を伸ばすことが重要課題であるが、それらは直接コントロールできない。その代わりに、広告を打ち出すことで間接的に売上をコントロールすることができる。どれくらい広告を打ち出せば、達成すべき売上に到達できるのか、そのモデルを与えるのが統計であり、企業の意思決定に必要なものである。

ここで、言葉の整理をしておく。

- 広告量：入力変数、もしくは説明変数（$X_1, X_2, \cdots$）
- 売上：目的変数、もしくは被説明変数（Y と表す）

より一般的に、連続的な値をとる目的変数 Y と p 個の異なる説明変数 $(X_1, \cdots, X_p)$ に、ある種の関係性があるとすると、

$$Y = f(X) + \varepsilon \tag{2.1}$$

と表せる。ここで、f は固定された、しかし未知の関数であり、ε は誤差項である。

別の例として、年収と教育年月との関係性を持つデータを紹介する。FIgure 2.2 には、x 軸に教育年数、y 軸に年収をとり、３０人のデータを示している（ただし、実際のデータではなくシミュレーションデータ)。シミュレーションデータなので、どのような関数 f を用いているか既知である（Figure 2.2 右の青い線）が、一般的に f の形は不明である。基本的には、データの分布を見て決定するのが１つの方法としてある。

より一般的には、関数 f は多数の説明変数を入力値として持つ。Figure 2.3 に年収を教育年数と、勤続件数の２つの説明変数を用いて予測するモデル結果を示している。

基本的に、統計学習は、f を算出するためのアプローチのことだと思ってくれればよい。

2.1.1　なぜ f を算出するのか？

目的は大別すると 2 つある：予測と推定。以下、それぞれについて説明する。

予測

よくある状況として、説明変数 X は簡単に得られるが、被説明変数 Y は簡単に得られない場合が多い。そんな時、f がわかっていると、

$$\hat{Y} = \hat{f}(X) \tag{2.2}$$

で被説明変数 Y を求めることができる。（平均すると、誤差項は 0 という分布を仮定したので、上の式には誤差項はない。）また、ここで $\hat{Y}$ や $\hat{f}$ は、我々が分析した推定値である。このような場合、$\hat{Y}$ さえわかれば良いという考えが往々にしてあるので、$\hat{f}$ はブラックボックスでも構わない。

さて、予測が（実測値と比較して）どれくらいの精度でできるのか？その精度は 2 つの量により定義される：reducible error と irreducible error。一般に、関数 f の形や、パラメータはわからないので、データから f を特定したとしても、実際の f と異なる可能性が十分ある。その差異が reducible error と呼ばれる。reducible（小さくすることができる）と名前がつけられているのは、もし関数 f を実際の f と同じ関数形に算出することができれば、関数 f と $\hat{f}$ の差はゼロにすることができるからである:

$$|\hat{f} - f| \sim 0$$

しかし、実際の値は関数 f だけでなく、誤差項 ε も含んでいるので、

$$|\hat{Y} - Y| \sim \varepsilon \quad (\text{when} \hat{f} = f)$$

と分析者がどうしても誤差項を小さくすることはできない。これが irreducible error（どんなに頑張っても小さく出来ない誤差項）である。

以上の議論を定量的に扱ってみる。 実測値と予測値の差の 2 乗の平均は、

$$\begin{aligned}
&E[(Y - \hat{Y})^2] \\
&= E[(f(X) + \varepsilon - \hat{f}(X))^2] \\
&= E[a + \varepsilon]^2 \qquad (a = f(X) - \hat{f}(X)) \\
&= E[a^2 + 2a\varepsilon + \varepsilon^2] \\
&= E[a^2] + \mathrm{Var}(\varepsilon) \qquad (E[x + y] = E[x] + E[y], E[\varepsilon^2] = \mathrm{Var}(\varepsilon)) \\
&= E[(f(X) - \hat{f}(X))^2] + \mathrm{Var}(\varepsilon) \\
&= [f(X) - \hat{f}(X)]^2 + \mathrm{Var}(\varepsilon).
\end{aligned} \tag{2.3}$$

となり、前の項が reducible error、後ろの項が irreducible error である。

この本では、（後ろの項はどうしようもないので）reducible error を最小にするような f を算出することを目的とする。ただし、気をつけないといけないのは、予測誤差の下限

値（一番良い誤差）は irreducible で定義されており、これはどうしようもないし、さらには、どれくらい irreducible error の大きさを持つのかも一般的にはわからない。

推定 (Inference)

推定の目的は、被説明変数に対する、各々の説明変数の影響度合いを調べることである。(例：TV 広告出稿を１０%増加させると、売上が２%上昇する)。このとき、予測の時で許されたブラックボックス f は許されないことが多い。

2.1.2 f の算出方法

以降の章で、f について色々な手法を紹介するが、共通する部分もあるので、ここで取り上げる。以降、データで n 観測点があるとする。また、説明変数は p 個あるとして、トレーニングデータセットは、$\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$、ここで $x_i = (x_{i1}, x_{i2}, \cdots, x_{ip})^T$ である。

パラメトリックな手法

パラメトリックな手法は以下の２ステップで構成される。

1. f の関数形を決める。例えば、X に対して線形だと仮定すると、

$$f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p. \tag{2.4}$$

   f の形を決めてしまえば、問題はものすごく単純になる。すなわち、もともと p 個の引数をとる $f(X)$ を求めることから、$p+1$ 個の係数 $\beta_0, \beta_1, \cdots, \beta_p$ を算出することに置き換わる。
2. 関数形を決めた後、トレーニングデータに合わせるようパラメータを決定する。上の例だと、$\beta_0, \beta_1, \cdots, \beta_p$ を求めることに等しい。よく使われる手法に、(ordinary) least squares がある（３章でやる)。それ以外にもたくさん手法があるが、それは６章で紹介する。

ここで紹介した方法がパラメトリックな手法と言われるもので、f を算出する問題を、幾つかのパラメータを算出する問題に置き換えていることに由来する。

パラメトリックな手法は問題を簡単にする分、別の問題が生じる。

- もし、本当の関数形（わかる分けないが）と異なる関数形を決定してしまうと、reducible error が大きくなる。
- パラメータ数をとてつもなく大きくして、関数が柔軟な形をとるようにもできるが、今度は、トレーニングデータに合わせすぎてしまい、テストデータと予測値が

全然合わないことがおきてしまう（過学習、オーバートレーニング)。

Figure 2.4 に、年収を目的変数に、教育と勤続年数を説明変数とした予測値と、実測値を示す。関数形は以下の式で与えられる。

$$(income) \sim \beta_0 + \beta_1 \times (education) + \beta_2 \times (seniority)$$

ノンパラメトリックな手法

ノンパラメトリックな手法は、関数の形を最初に決めない。そうすることで、真の関数形 f と全く異なるという危険性は回避できる。しかし、f を算出することを幾つかのパラメータを求める、という問題の単純化を行っていないので、データ数 n は十分大きいことが求められる。

Figure 2.5 にノンパラメトリック手法の 1 つの thin-plate spline で求めたモデルを示す。Figure 2.3 が真の関数形であるが、それとほぼ同じことがわかる。スプライン関数を使用するとき、関数形がどれくらいスムーズなのか、分析者は入力する必要がある。Figure 2.6 にスムーズ性をほぼなくした結果を示す。トレーニングデータと予測値の誤差が 0 となっていることがわかり、オーバートレーニングしていることが読み取れる。

どういう風にスムーズ性を入れればよいのかは 5 章で、スプライン関数については 7 章で述べる。

2.1.3　予測精度と解釈可能性のトレードオフ

線形回帰モデルの場合、関数の形を事前に決めてしまうため、柔軟性に欠ける。一方、ノンパラメトリックな手法のところで示した、スプライン関数による回帰は、柔軟に f を決定できるため、柔軟性がある。

しかし、柔軟性に欠ける線形回帰モデルは、意味の解釈は容易である。反対に、スプライン関数において説明変数と被説明変数の関係性を言及しようとすると、難しい。

Figure 2.7 に柔軟性を x 軸に、解釈可能性を y 軸にとり、様々なモデルをその中に示した。線形回帰 (Lease Squares) よりも解釈可能性が高いものに、Lasso 回帰モデルがあるが、これについては 6 章で詳細に述べる。Lasso 回帰は、単純に説明すると、Least Squares で見積もった回帰係数 $\beta_0, \beta_1, \cdots, \beta_p$ のうち、いくつかは 0 となり、パラメータの個数が小さくなるというものである。すなわち、

$$f(X) = \beta'_0 + \beta'_1 X_1 + \cdots \beta'_m X_m \quad (where \ \ m \leq p).$$

Generalized addtive models (GAMs) は 7 章で説明するが、線形モデルに、ある非線形関係を許すものである。bagging, boosting, support vector machines は 8 、 9 章で説明するが柔軟性は高いが、解釈は難しい。

2.1.4 教師つき vs. 教師なし学習

- 教師つき学習：線形回帰、ロジスティック回帰、GAM、boosting、SVM
- 教師なし学習：クラスター分析 or クラスタリングなど（詳細は１０章）

Figure 2.8 はクラスター分析の結果を示している。説明変数は２つしかないので、散布図を描くのは容易だが、説明変数が p 個あると、組み合わせは $p(p-1)/2$ 個となり、散布図を描くのも困難になる。この辺りも１０章で述べる。

また、教師つきか、教師なしか問題がはっきりしないこともある。例えば、n 個のデータのうち m 個のみが説明変数と被説明変数両方の値を持っていて、$n-m$ 個が説明変数のみ値を持っているという場合がある。こういう問題は semi-supervised learning 問題と呼ばれる。望むべきは、m 個のデータと、$n-m$ 個のデータを全て学習させることができる方法があればいいが、この話題は本書のスコープ外である。

2.1.5 回帰 vs. 分類問題

ひとまず、以下のように認識してもらえば問題ない。

- 回帰：被説明変数が連続値をとる
- 分類：被説明変数が離散値をとる

2.2 モデル精度の評価 (Assessing Model Accuracy)

この節では、あるデータが与えられた際にどの統計モデルを選択するか、という事について概念的に述べる。（具体的な例は後の章でやります。たぶん。）

2.2.1 フィッティング精度の測定

統計モデル f が与えられたとき、どれくらいデータにフィットされているのかを議論したい。回帰の場合、よく平均２乗誤差 (mean squared error, MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2 \tag{2.5}$$

が用いられる。ここで $\hat{f}(x_i)$ は i 番目の観測量における予測値である。もちろん、MSE が小さいほどデータをよく再現できているということになる。

（これから述べることは超重要なので、記憶しておいてください。）式 (**??**) はモデル f に対してフィッティングさせた、トレーニングデータに対して計算しているので、正確には training MSE と呼ばれる。しかし、一般的には、分析者はトレーニングデータの精度

はあまり気にしないことが多く、むしろ興味があるのは、未知のデータが入ってきたとき、どれくらい精度よく予測できるのだろうか、ということである（未知のテストデータ、あるいは誤解がないときは単にテストデータと呼ばれることもある)。例をあげると、

- 天気予報を過去数十年のデータから、天気予報が９０%あたるモデル式を構築したとしても、果たして明日の天気が９０%で予測できるのか？
- ある会社の株価を、有効求人倍率や、鉱工業生産数などで８０%の精度で予測したとしても、果たして明日の株価が８０%で予測できるのか？

数式で言ってみると以下のようになる。トレーニングのデータセットが n 個 $\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$ あって、それを用いて $\hat{f}$ を算出したとする。手元には、$\hat{f}(x_1), \cdots, \hat{f}(x_n)$ があって、上記の議論から、これは小さいと仮定する（すなわち、よくデータにあうように f を推定できた、という意味)。しかし、今興味があるのは、新たに (x_0, y_0) というデータ（複数のデータ含む）が手に入ったとき、どれくらい $\hat{f}$ で予測できるか、ということである。すなわち、以下の量を計算したい:

$$\mathrm{Ave}(y_0 - \hat{f}(x_0))^2. \tag{2.6}$$

この量 (test MSE と呼ぶ) を出来る限り小さくしたい。

トレーニングデータセットと、テストデータセットは共に与えられる場合があるが、トレーニングデータセットしか得られない場合の方が多い。そういう場合、どういう風にモデルを選択すればいいのか。単純に考えると、training MSE を小さくするモデルを採択したくなる（誤差が一番小さくするようなモデルを選ぶのは当然！)。しかし、この方法は大概オーバーフィッティングしてしまいがちである（すなわち training MSE は最小だけれど、test MSE は大きくなってしまう)。

Figure 2.9 に単純な例を示す。左側のグラフには、シミュレーションによって発生させたデータ点 (○) と、真のモデル f が黒線で描かれている。データに対してフィッティングさせた結果が他の３線で、オレンジは線形回帰を、水色と緑色はスプライン関数でフィッティングさせたものである（緑色の方が smoothness が小さい)。右の図には、トレーニングデータとテストデータの MSE の大きさを、柔軟性を x 軸にとって示している。線形回帰のもの（オレンジ）が柔軟性がもっとも低く、逆に、柔軟性が最も高い (smoothness が小さい) スプライン関数が右側にきている。赤色の線がテストデータ、灰色の線がトレーニングデータの値を示しており、テストデータの MSE はトレーニングデータの MSE よりも大きくなっている。また、柔軟性を大きくした、緑色は、トレーニングデータでの MSE は小さいが、テストデータの MSE は極小ではないことが見て取れる。

ここで重要なのは、（何度も言うが）training MSE が極小をとるときに、test MSE は

極小とならない、ということである。一般的に、training MSE は柔軟性が増すにつれて小さくなるが、test MSE は U 字のようになる。

Figure 2.10 に真の関数形が直線の場合を、Figure 2.11 に真の関数形が波をうっているような、ギザギザな場合をそれぞれ示している。Figure 2.9 と比べて、test MSE の極小点が左にいくか、それとも右にいっているか、という違いだけ。

最後に、テストデータが得られない場合に、トレーニングデータからテストデータを生成する手法があり (cross validation)、これについては 5 章で述べる。

2.2.2 バイアス-バリアンスのトレードオフ (The Bias-Variance Trade-Off)

test MSE に見られた U-shape は、統計学習において 2 つの競合する性質によるものである。数学的に導出するのは、この本のスコープ外であるが、説明だけはしておく。

test MSE の 2 乗の期待値は、3 つの量で定義される：関数 $\hat{f}(x_0)$ のバリアンス (variance)、2 乗のバイアス (squared bias)、誤差項のバリアンス。すなわち、

$$E(y_0 - \hat{f}(x_0))^2 = \mathrm{Var}(\hat{f}(x_0)) + [\mathrm{Bias}(\hat{f}(x_0))]^2 + \mathrm{Var}(\varepsilon). \tag{2.7}$$

$E(y_0 - \hat{f}(x_0))^2$ は、test MSE の期待値を意味する。すなわち、もし何度もトレーニングデータセットと、テストデータセットが得られる状況での、test MSE の平均ということである。この式から、最小の test MSE を見つけるときには、low variance かつ low bias の関数 $\hat{f}$ を算出すればよい、ということである（test MSE の下限値は irreducible error (2.3) で定義されており、これ以上の精度の改善は見込めない)。

さて、ここでバリアンス、バイアスの意味を以下にそれぞれ述べる。

- バリアンス (variance) とは、もし（今使用しているものと）異なるトレーニングデータを関数 $\hat{f}$ の算出に使用した場合、どれくらい変わるのか、という量。(真のモデルが既知ならば、誤差項の影響でデータが変わった、という意味。全く別のデータというわけではない。念のため。）以前の例で、スムーズ性が小さいスプライン関数があったが、もしそういう場合に別のトレーニングデータが提供されれば、関数の形は大きく異なることが想像できるだろう。すなわち、(柔軟性が大きい統計モデルは）大きなバリアンスをもつということである。また、線形回帰モデルの場合、トレーニングデータが変更されても、パラメータが少し影響を受け変更されるかもしれないが、大きく予測値が変わる、ということもないであろう（low variance)。
- バイアス (bias) とは、単純には算出されたモデル式 $\hat{f}$ と、実測値あるいは真の関数形との差である（真の関数形がわかっているシミュレーションの場合は真の関数形 f、そうでない場合はデータとの適合度みたいなもので認識しておいてくださ

い。ここは、ちょっと細かい定義とか言い出すと、とたんに難しくなるのでフワッとぼやかしておきます)。例えば、真のモデルが波をうったスプライン関数で、それに線形回帰をしても、全然合わないことが想像できるだろう（high bias)。

以上から、low-variance をとれば high-bias に、low-bias をとれば high-variance になることが理解できるだろう。

Figure 2.12 に Figure 2.9-2.11 での MSE に、Bias と Variance を重ねて描いたものを示す。見てわかるように、柔軟性を大きくする（すなわち bias を小さくする）と、variance が大きくなる。これを bias-variance トレードオフという。

2.2.3 **分類における精度評価** (The Classification Setting)

今までの議論は、回帰、すなわち目的変数が連続値をとる統計モデルに関して述べてきたが、分類の場合にも bias-variance トレードオフのような事が同じように当てはまる。今、トレーニングデータ $\{(x_1, y_1), \cdots, (x_n, y_n)\}$ があり、y_i は離散的な値をとるとする。この時、モデル $\hat{f}$ の精度評価をする際、一般的なアプローチは training error rate を以下のように定義し、それを使用する：

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i). \tag{2.8}$$

ここで、$\hat{y}_i$ は i 番目の観測値から $\hat{f}$ を使った予測値である。$I(y_i \neq \hat{y}_i))$ は *indicator variable*(インジケータ変数と日本語ではいう？) であり、もし予測値の分類と実測値の分類があっていれば 0 を、間違っていれば 1 を返すものである。すなわち、上の式は、n 個のうち分類を間違っていた数を n で割った量であり、誤差率と呼ぶにふさわしい。

ただ、回帰のときと同じように、正確には training error rate である。実際は training error rate よりも test error rate

$$\mathrm{Ave}(I(y_0 \neq \hat{y}_0)) \tag{2.9}$$

を最小にするようなモデルが欲しい。

ベイズ分類器 (The Bayes Classifier)

数学的な証明は省くが、test error を最小にするような分類方法は、各クラスの確率を算出して、確率が最大となるクラスを当てるというものである。すなわち、分類方法が J 個あって、説明変数 x_0 が与えられた時、j 番目のクラスになる確率

$$\Pr(Y = j | X = x_0) \tag{2.10}$$

を算出し、確率最大となるクラスを予測値とする。 (**??**) は条件付き確率と呼ばれる（説明変数 x_0 が与えられたもとでの j 番目のクラスになる確率)。この単純な分類器をベイ

ズ分類器と呼ぶ。よく例としてあがるのは、$J = 2$、すなわち、信号かバックグラウンドかの場合で、この時、確率が 0.5 以上になる方を予測値として採用する。

Figure 2.13 に $J = 2$ の場合のシミュレーションデータを用いたベイズ分類器での分類を示す。紫色の破線で示されているところが境界条件、すなわち、条件付き確率が 0.5 になる場合である。シュミレーションデータであるので、どういう確率分布からデータが生成されているかがこの場合は既知であり、条件付き確率が計算できる。

ベイズ分類器による誤差率をベイズ誤差率と呼ぶ。ベイズ分類器は条件付き確率が最大となる分類 j を採択するので、誤差率は $1 - \max_j Pr(Y = j|X = x_0)$ となる。一般に、多数のテストデータが得られる場合、誤差率は平均するのがほとんどなので、

$$1 - E(\max_j \Pr(Y = j|X)) \tag{2.11}$$

となる。

K-nearest Neighbors (K-**最近傍**?)

現実問題は、条件付き確率を計算できないので、ベイズ分類器は使用不可である。そのため、何らかの確率を計算し、それを基準にクラス分けする必要がある。そういう方法の１つに、K-nearest neighbors (KNN) 分類器がある。これは、テストデータの説明変数 x_0 の周辺にある K 個のトレーニングデータを同定し、そのトレーニングデータがどの分類になっているか、多数決をとるものである。確率としては、

$$\Pr(Y = j|X = x_0) = \frac{1}{K} \sum_{i \epsilon N_0} I(y_i = j) \tag{2.12}$$

を求めて、確率最大となるクラスをテストデータに当てはめる。ここで、N_0 はテストデータ x_0 の周辺にあるトレーニングデータ K 個の集合である。

Figure 2.15 に $K = 10$ とした KNN 分類結果を示す。破線がベイズ分類器による理論値（この場合シュミレーションデータなので、条件付き確率は算出できるため)、実線が KNN であり、大体よく予測できているようである。

容易に想像できるように、もしテストデータの最近傍の個数 K を 1 個などとしてしまうと、テストデータが変わる度に大きく KNN の予測が変化する (high variance)。逆に、K を大きくしてしまうと、今度はデータ全体でならされてしまい、予測精度が落ちる (high bias)。Figure 2.16 にその様子が示されている。

分類の場合の柔軟性は、KNN の場合 $1/K$ である。K の下限値はもちろん 1 であり、上限値は全トレーニングデータ数 N である。すなわち、$1/K$ は $1/N \leq K \leq 1.0$ を満たす。回帰の場合と同様に、test error と training error を y 軸に、柔軟性を x 軸にとったものを示す。回帰と同様に、test error が U-shape なのが見て取れる。

2.3　Lab: Introduction to R

今のところ割愛する。