

IT 資格 道 場

E資格 学習用テキスト

シラバス2026の全範囲 —— 読んで理解する1冊目

無料サンプル(第1章まで)

JDLA E資格 試験出題範囲(シラバス2026・第1.0版・E2026#2～) 準拠
2026年10月版

ご利用にあたって

- このテキストは「IT資格道場」(www.shikaku-doujou.com)の購入者限定テキストのうち、学習用テキストの第1章までを収録した**無料サンプル**です。どなたでもご覧いただけます。
- 無料サンプルも著作物です。転載・改変しての再配布・二次販売はできません。
- AI モデルの学習データとしてのご利用(学習目的の入力・データセット化を含む)はできません。
- 製品版(学習用テキスト・頻出度別ノート・暗記用ノートの3点)は、ご購入者を識別できる透かし入りのPDFとして提供されます。
- 本PDFはスマートフォン・タブレットでの閲覧に合わせたA5版面・文字サイズで組版しています。

目次

3点セットの中の位置づけ

この教材の使い方

試験の基本情報

出題範囲の全体像: 結果発表の4科目と章立て

第1章 数学的基礎 [1.2.2/1.3.1]

確率モデルとパラメータ推定の見取り図 [1.2.2]

ベイズ則と条件付き確率の計算 [1.2.2]

ナイーブベイズ分類器(条件付き独立の仮定) [1.2.2]

最尤推定と対数尤度 [1.2.2]

MAP推定とベイズ推定 [1.2.2]

自己情報量とエントロピーの計算 [1.3.1]

結合エントロピー・条件付きエントロピー・相互情報量の関係 [1.3.1]

クロスエントロピー(分類の損失との対応) [1.3.1]

KLダイバージェンスとJSダイバージェンス(非対称性・値の範囲) [1.3.1]

サンプルはここまでです

3点セットの中の位置づけ

このシリーズは3つのテキストで構成されています。

- **学習用:** これだけで問題が解けるようになる本編
- **頻出度別:** テキストを読んだら次はこれ。頻出順に絞った問題と解説で「解ける」に橋渡しします。試験直前の最終チェックにも
- **暗記用:** 覚えるべき要点だけを一問一答に凝縮。空き時間の反復と用語の再確認用

使い方: ① 学習用を読む → ② 頻出度別で解き方を掴む → ③ 問題演習で腕試し → ④ 頻出度別に戻って再確認(試験直前もここ)。暗記用は空き時間や用語の再確認に。

この教材の使い方

このテキストは、一般社団法人日本ディープラーニング協会(JDLA)が実施する **E資格(JDLA Deep Learning for ENGINEER)** の合格を目的に作られています。

E資格が問うのは、深層学習の仕組みを式と数値で追えること、そしてコードを読んで結果を選べることです。名前を知っているだけでは足りません。層の入出力の形、損失と勾配の式、既定値を、手で計算して答える問題が中心になります。

出題範囲は、公式の試験出題範囲(シラバス)で小項目の単位に決められています。**出題の再現に頼るのではなく、シラバスの項目を軸に据えて体系で押さえるのが、もっとも確実な道になるでしょう。**範囲の外とされた項目は、この教材にも書いていません。

試験の基本情報

項目	内容
試験名	E資格(JDLA Deep Learning for ENGINEER)
主催	一般社団法人日本ディープラーニング協会(JDLA)
受験資格	JDLA 認定プログラムを、試験日の過去2年以内に修了していること
受験形式	各地の指定試験会場での受験(申込はピアソン VUE の E資格受験サイト)
試験時間	120分

項目	内容
出題数	100問程度 (直近の2回の実績は104問と105問)
出題形式	知識問題・多肢選択式。コードを含む問題は Python かフレームワークで出ます
出題範囲	シラバス2026(第1.0版) 。E2026#2(2026年8月)から適用されています
受験料	一般 33,000円 ・学生 22,000円 ・会員 27,500円 (税込)

受験資格・受験料・試験時間・出題数・シラバスは改定されることがあります。
お申し込みの前に、JDLA 公式サイトで最新の情報をご確認ください。

試験の前に知っておきたい点が2つあります。

- **フレームワーク:** コードの問題は、試験開始時に PyTorch か TensorFlow のどちらかを選んで受ける。この教材は両方の名前を併記
- **合格点と内訳:** 合格点も、分野ごとの出題数の内訳も、公式には非公表。
この教材の章の配分は、受験者の体験記から見積もった目安

出題範囲の全体像：結果発表の4科目と章立て

分野	対応する章	一言でいうと
応用数学	第1章	損失関数がなぜその形をしているのかを、確率と情報理論で説明する
機械学習	第2章	データの分け方、過剰適合の見分け方、性能指標の計算

分野	対応する章	一言でいうと
深層学習	第3章・第4章	層・損失・最適化・正則化の仕組みと、画像・言語・生成・強化学習などの応用
開発・運用環境	第5章	モデルの軽量化、分散学習、計算装置、コンテナ

SAMPLE

第1章 数学的基礎 [1.2.2/1.3.1]

E資格の数学は、深層学習の損失関数がなぜその形をしているのかを説明するために置かれています。範囲は2つです。1つは確率モデルのパラメータ推定で、ベイズ則・ナীবベイズ・最尤推定・MAP推定・ベイズ推定が入ります。もう1つは情報理論で、自己情報量・エントロピー・相互情報量・クロスエントロピー・KLダイバージェンス・JSダイバージェンスが入ります。この2つは別々の話に見えますが、1本の線でつながっています（出題の多くは、その線の上の1点を問います）。

データの尤度を最大にする最尤推定は、経験分布とモデル分布のKLダイバージェンスを最小にすることと同じです。KLダイバージェンスはクロスエントロピーからデータ側のエントロピーを引いたものなので、クロスエントロピーを最小にしても同じ最適点になります。分類の損失として使うクロスエントロピー誤差は、このつながりの最後の形です。回帰の平均二乗誤差も、ガウス分布を仮定した最尤推定として同じ線の上に置けます。MAP推定は、この線に事前分布の項を1つ足したもので、足しても線そのものは切れません。

この章の式は、記号の意味を読み取れば計算できます。言葉で問われる問題は、どの量とどの量が同じ最適点を持つか、どちらの向きのKLを使うか、値の範囲はいくつか、といった判断です。計算の問題は、数表から確率を作り、定義に代入する形で出ます。数値例はどれも手で追える大きさと、筆算で再現できます。

確率モデルとパラメータ推定の見取り図 [1.2.2]

確率モデルは、データがどう生まれるかを、パラメータ θ をもつ確率の式で書いたもの。観測したデータを x 、知りたいパラメータを θ と書くと、パラメータ

推定の問いは「 x を見た後で θ をどう決めるか」になります。コインの表が出る確率を θ として、10回投げた結果から θ を決める作業が、最も小さな例です。

用語は3つです。 $p(x|\theta)$ は、 θ を与えたときにデータ x が出る確率で、 θ の関数として見るときは尤度と呼びます。 $p(\theta)$ は、データを見る前の θ についての知識で、これを事前分布と呼びます（英語では prior）。 $p(\theta|x)$ は、データを見た後の θ についての知識で、事後分布と呼びます。尤度は「 θ が正しいとしたらこのデータはどれだけ出やすいか」を表すもので、 θ についての確率分布ではありません。 θ について足し合わせても1にならない点が、事後分布との違いです。

3つの推定法は、この3つの量をどこまで使うかで分かります。最尤推定は尤度だけを使い、 θ の1点を返します。MAP推定は尤度と事前分布の積が最大になる1点を返し、ベイズ推定は点を決めず、事後分布そのものを返します。この順に使う情報が増え、事前分布を使うのはMAP推定とベイズ推定です（最尤推定は使いません）。

ベイズ則と条件付き確率の計算 [1.2.2]

ベイズ則は、条件付き確率の向きを入れ替える式です。

$$P(x|y) = P(x) P(y|x) / P(y)$$

$$P(y) = \sum_x P(y|x) P(x)$$

$P(y|x)$ と $P(x)$ が分かっている、 $P(x|y)$ が知りたいときに使います。分母の $P(y)$ は、 x の取り得る値すべてについて $P(y|x)P(x)$ を足した値で、「 y が起きる確率の総量」にあたります。分子は「 x であって y でもある」場合の確率なので、事後確率は、全体に占める割合として読めます。

クラス y と特徴 $x_1, \dots, x_n$ の形に書き直すと、次の式になります (分類で使うのはこの形)。

$$P(y|x_1, \dots, x_n) = P(y) P(x_1, \dots, x_n|y) / P(x_1, \dots, x_n)$$

$P(y)$ が事前確率、 $P(x_1, \dots, x_n|y)$ が尤度、左辺が事後確率です。分母は、尤度と事前確率の積を、全クラスについて足した値になります。[c_1-1][c_1-3]

計算の手順

計算問題は、いつも同じ4つの手順で解けます。

1. 事前確率 $P(x)$ を表から拾う。
2. 尤度 $P(y|x)$ を、 x の取り得る値ごとに拾う。
3. 分母 $P(y)$ を、事前確率と尤度の積の総和で作る。
4. 分子を分母で割る。

例を1つ解きます。ある病気の有病率は1%です。検査は、病気の人を90%の確率で陽性にし、病気でない人も5%の確率で陽性にします。陽性が出たとき、本当に病気である確率を求める問題でした。分母は $P(\text{陽性}) = 0.9 \times 0.01 + 0.05 \times 0.99 = 0.009 + 0.0495 = 0.0585$ で、答えは $P(\text{病気}|\text{陽性}) = 0.009 / 0.0585 \approx 0.154$ 。感度が90%でも、有病率が低いと、陽性者の大半は病気でない人になります。分母のうち、病気でない人からの陽性0.0495が、病気の人からの陽性0.009を大きく上回るためです。

検査を独立に2回受けて、どちらも陽性だったとします。1回目の事後確率 0.154 を、2回目の事前確率に使います。分母は $0.9 \times 0.1538 + 0.05 \times 0.8462 \approx 0.1808$ で、答えは $P(\text{病気}|2\text{回陽性}) = 0.9 \times 0.1538 / 0.1808 \approx 0.766$ 。事後確率が次の更新の事前確率になる関係は、ベイズ推定でデータを足すたびに

分布を更新する操作のもとになります。検査どうしが独立なら、最後の事後確率は尤度の積で決まるので、更新の順序を入れ替えても値は変わりません。

原因が3つ以上ある場合も、手順は変わりません。3台の機械 A・B・C が、製品の50%・30%・20%を作り、不良品の率がそれぞれ2%・3%・5%だとします。不良品を1個選んだとき、各機械で作られた確率を求めます。分子は $0.5 \times 0.02 = 0.010$ ・ $0.3 \times 0.03 = 0.009$ ・ $0.2 \times 0.05 = 0.010$ 、分母はその和の 0.029 です。事後確率は A が 0.345 、B が 0.310 、C が 0.345 になります。作る数の少ない C が A と並ぶのは、不良率の高さが、作る数の少なさを打ち消すからで、確率の大小は、不良率だけでも生産量だけでも決まりません。3つの事後確率の和は、必ず1になります。

間違えやすいのは、 $P(\text{陽性} | \text{病気})$ と $P(\text{病気} | \text{陽性})$ を同じ値として扱う誤りです。前者は検査の性能で、後者は陽性者のうちの病気の割合を指します。2つを結ぶのがベイズ則です。事前確率が入るぶん、値は大きく違うことがあり、同じ値にはなりません。事前確率が0の原因は分子 $P(x)P(y|x)$ が0なので、どんな観測の後でも事後確率は0から動きません。

ナイーブベイズ分類器(条件付き独立の仮定) [1.2.2]

ナイーブベイズは、ベイズ則を分類に使い、特徴の間に強い仮定を置いて計算を単純にした分類器です。特徴が多いと、 $P(x_1, \dots, x_n | y)$ を表にするには、組み合わせの数だけの値が要ります。そこで、クラス y が決まれば特徴同士は互いに独立だと仮定します。

$$P(x_i | y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i | y)$$

この仮定を条件付き独立と呼びます。単なる独立とは違い、 y を与えたうえでの独立で、 y を無視した独立ではありません。 y を与えなければ、特徴同士に

相関があってもかまいません。この仮定を置くと、事後確率は次のように分解されます。

$$P(y|x_1, \dots, x_n) \propto P(y) \prod_i P(x_i|y)$$

分母 $P(x_1, \dots, x_n)$ は、入力が決まれば全クラスで同じ定数です。クラスを比べるだけなら分母は要らず、分子の大きいクラスを選べば足ります。決定則は次の形です。

$$\hat{y} = \operatorname{argmax}_y P(y) \prod_i P(x_i|y)$$

scikit-learn の説明では、 $P(y)$ と $P(x_i|y)$ をどちらもMAP推定で求め、 $P(y)$ は訓練集合での各クラスの相対頻度になるので、特徴の値には依存しません。[1-3] 積は小さな値のかけ算になり、桁が足りなくなるので、実装では対数をとって和にします (argmax は対数をとっても変わりません)。

尤度の置き方とパラメータ

特徴の型で $P(x_i|y)$ の置き方が変わります。特徴が連続値のときは、ガウス分布を置きます。

$$P(x_i|y) = 1/\sqrt{2\pi\sigma_y^2} \cdot \exp(- (x_i - \mu_y)^2 / (2\sigma_y^2))$$

μ_y と σ_y は、クラス y に属する訓練データから、最尤推定で求めます。scikit-learn の GaussianNB の既定値は `priors=None, var_smoothing=1e-09` です。`priors=None` の場合、事前確率は訓練集合の割合から決まり、自分で与える必要はありません。`var_smoothing` は、全特徴の分散のうち最大のものに

掛ける割合で、その値を各分散に足して、分散が0になって割り算が壊れるのを防ぎます(計算の安定のため)。^[^c1-4]

特徴が単語の出現回数のときは、多項ナイーブベイズを使います。クラス y の単語 i の確率を、数え上げから次のように推定します。

$$\theta_{yi} = (N_{yi} + \alpha) / (N_y + \alpha n)$$

N_{yi} はクラス y の文書で単語 i が出た回数、 N_y はクラス y の文書の全単語数、 n は語彙の大きさです。 α は平滑化の強さで、 $\alpha=1$ がラプラス平滑化、 $\alpha<1$ がリッドストーン平滑化です。 $\alpha=1$ は、語彙の全単語があと1回ずつ出たと数えるのと同じです(分母には語彙数 n の分を足します)。MultinomialNBの既定値は `alpha=1.0, force_alpha=True, fit_prior=True, class_prior=None` で、既定のままでラプラス平滑化($\alpha=1$)が働き、0の確率は出ません。^{[^c1-3][^c1-5]}

平滑化が必要な理由を計算で見る

小さな例で確かめます。語彙は「安い・無料・会議」の3語です。訓練データは、スパムの文書群で全8語(安い3・無料4・会議1)、通常のメールの文書群で全6語(安い0・無料1・会議5)でした。事前確率はスパム0.4、通常0.6です。

$\alpha=0$ (平滑化なし)では、通常メールで「安い」の確率が0になります。「無料 安い」というメールを分類すると、通常側の積は $0.6 \times (1/6) \times 0 = 0$ で、スパムの事後確率が1になります。1つの単語が訓練データに1度も出なかっただけで、他の単語の証拠がすべて無視される形です。

$\alpha=1$ に直すと、スパム側は安い $4/11$ ・無料 $5/11$ ・会議 $2/11$ 、通常側は安い $1/9$ ・無料 $2/9$ ・会議 $6/9$ になり、どの確率も0になりません。「無料 安い」の分子は、スパムが $0.4 \times (5/11) \times (4/11) \approx 0.0661$ 、通常が

$0.6 \times (2/9) \times (1/9) \approx 0.0148$ で、スパムの事後確率は $0.0661 / (0.0661 + 0.0148) \approx 0.817$ です。「無料 会議」では、スパムが約 0.0331 、通常が約 0.0889 となり、スパムの事後確率は約 0.271 に下がります。分母の α_n の n は語彙数なので、足し忘れると、確率の総和が1を超え、確率として使えません。逆に α を大きくするほど、各単語の θ は $1/n$ に近づき、出現回数の差が薄れます。尤度の差が消えるので、事後確率は事前確率に近づき、単語の証拠では差が付きません。

実装では、確率の積ではなく、対数の和でクラスを比べます。上の「無料 安い」では、スパムが $\ln 0.4 + \ln(5/11) + \ln(4/11) \approx -2.716$ 、通常が $\ln 0.6 + \ln(2/9) + \ln(1/9) \approx -4.212$ で、スパムの方が大きい値です。「無料 会議」では、スパムが約 -3.409 、通常が約 -2.420 で、通常の方が大きくなります。対数は単調増加なので、順位は確率の積で比べたときと同じになります（値は負で、0に近いほど大きい）。比べたいのは順位だけなので、分母 $P(x_1, \dots, x_n)$ を計算する必要もありません。

ナイーブベイズの性質は、次のとおりです。条件付き独立の仮定は現実にはほぼ成り立ちません。それでも、クラスの順位付けだけなら十分に働くことが多く、学習は数え上げだけで終わります。確率の値そのものは、極端な値に寄りやすい点が弱みです。

最尤推定と対数尤度 [1.2.2]

最尤推定は、観測したデータが最も出やすくなるパラメータを選ぶ方法です。データ $x^{(1)}, \dots, x^{(m)}$ が独立に同じ分布から得られたとすると、次のように書けます。

$$\theta_{ML} = \operatorname{argmax}_{\theta} p_{\text{model}}(X; \theta) = \operatorname{argmax}_{\theta} \prod_i p_{\text{model}}(x^{(i)}; \theta)$$

この積は、 m が大きいと0に近い値の積になり、計算機で桁があふれ（アンダーフローし）、正しく扱えません。そこで対数をとって和にします。

$$\theta_{ML} = \operatorname{argmax}_{\theta} \sum_i \log p_{\text{model}}(x^{(i)}; \theta)$$

対数は単調増加なので、最大を与える θ は変わりません。この和が対数尤度です。損失として最小化したいときは、符号を反転した負の対数尤度を使います。[^c1-2]

コインの例で確かめます。表を θ の確率で出すコインを10回投げて、表が7回、裏が3回でした。対数尤度は $7 \log \theta + 3 \log(1 - \theta)$ で、 θ で微分して0とおくと $7/\theta - 3/(1 - \theta) = 0$ となり、 $\theta = 0.7$ が出ます。表の割合そのものが最尤推定量です。10回では偏りが残るので、少ないデータでは、最尤推定が極端な値を返す場合があります。

最尤推定とダイバージェンス

データの経験分布を p_{data} 、モデルの分布を p_{model} とします。最尤推定は、 p_{data} と p_{model} のKLダイバージェンスを最小にすることと同じで、最適な θ は変わりません。

$$D_{KL}(p_{\text{data}} \parallel p_{\text{model}}) = E_{\{x \sim p_{\text{data}}\}} [\log p_{\text{data}}(x) - \log p_{\text{model}}(x)]$$

右辺の第1項は、データだけで決まり、 θ に依存しません。したがって θ を動かして最小にするのは、第2項の $-E[\log p_{\text{model}}(x)]$ だけです。これは負の対数尤度の平均と同じ量なので、最尤推定とKL最小化は、同じ最適点を返します。この対応は、クロスエントロピーの節でもう一度使います。[^c1-2]

ガウス分布を仮定すると平均二乗誤差になる

回帰では、入力 x のときの出力 y の分布を、平均がモデルの予測 $\hat{y}(x; w)$ 、分散が固定の値 σ^2 のガウス分布と仮定します。

$$p(y|x) = N(y; \hat{y}(x; w), \sigma^2)$$

m 個のデータの対数尤度は、次のようになります。

$$\sum_i \log p(y^{(i)} | x^{(i)}) = -m \log \sigma - (m/2) \log(2\pi) - (1/(2\sigma^2)) \sum_i (y^{(i)} - \hat{y}^{(i)})^2$$

w に依存するのは最後の項だけで、負の符号がついています。対数尤度を最大にすることは、二乗誤差の和を最小にすることと同じです。平均二乗誤差

$$MSE(y, \hat{y}) = (1/n) \sum_i (y_i - \hat{y}_i)^2$$

の最小化が返す w と、一致します。MSE の値と対数尤度の値は違っても、最適な w の位置は同じです。この関係が、MSE を最尤推定として正当化できる理由です。^{[1][6]} 分散を固定すると σ は定数になるので、MSE の最小化は σ の値に影響されません。分散まで推定対象にすると、別の式が出ます。

具体的な数値で確かめる

ガウス分布の平均と分散を最尤推定する例です。観測値が 2, 4, 6 の3つのとき、平均の最尤推定量は標本平均の 4 です。分散の最尤推定量は、偏差の二乗の和 $(-2)^2 + 0^2 + 2^2 = 8$ を観測数 $m=3$ で割った $8/3 \approx 2.667$ になります。観測数-1 で割る不偏分散は $8/2 = 4$ で、最尤推定量とは一致しません。最尤推定の分散は m で割るので、少し小さく出る点を覚えておきます。同じ観

測を複製して 2m 個にしても、和と個数が同じ倍率で増えるので、平均も分散の最尤推定量も変わりません。

MAP推定とベイズ推定 [1.2.2]

最尤推定は、データだけから θ を決めます。データが少ないと、極端な値に決まりやすくなります。MAP推定は、尤度に事前分布を掛けて、事後分布が最大になる θ を選びます。

$$\theta_{\text{MAP}} = \operatorname{argmax}_{\theta} p(\theta | x) = \operatorname{argmax}_{\theta} [\log p(x | \theta) + \log p(\theta)]$$

右辺の第1項が通常対数尤度で、第2項が事前分布の対数にあたります。ベイズ則の分母 $p(x)$ は θ に依存しないので、最大化の式には要りません。事前分布が平らで何も教えない(一様)ときは、第2項が定数になり、MAP推定は最尤推定と一致します。[^c1-2]

事前分布と正則化の対応

重み w にガウス事前分布をおきます。平均は0、分散は $1/\lambda$ の形です。

$$p(w) = \text{N}(w; 0, (1/\lambda) I)$$

このとき対数事前項は、定数を除いて $-(\lambda/2) w^T w$ に比例します。MAP推定の目的関数は、負の対数尤度に $(\lambda/2) w^T w$ を足した形で、重み減衰 (weight decay、L2正則化) と同じです。事前分布が狭い (λ が大きい) ほど、重みを0に引き寄せる力が強くなり、 $\lambda=0$ なら引き寄せる力は働きません。ガウス事前分布のMAP推定は、重み減衰を持つ最尤推定に対応します。[^c1-2]

数値で確かめます。観測値 2, 4, 6 から平均 w を推定します。観測のノイズは分散 $\sigma^2=1$ のガウス、事前分布は平均0・分散 τ^2 のガウスとします。事後

分布の最大化は、次の式の最小化に変わります。

$$\sum_i (x_i - w)^2 / (2\sigma^2) + w^2 / (2\tau^2) \rightarrow \text{最小}$$

微分して0とおくと、 $w_{\text{MAP}} = \sum x_i / (m + \sigma^2/\tau^2)$ が出ます(σ^2/τ^2 が縮みの強さを決めます)。 $\tau^2=1$ なら $12/(3+1) = 3$ 、 $\tau^2=0.5$ なら $12/(3+2) = 2.4$ です。最尤推定の 4 から、事前分布の中心0へ向かって縮みます。縮み方は、事前分布の分散が小さいほど大きくなり、データ数 m が増えるほど小さくなります。データが十分にあれば、MAP推定は最尤推定に近づきます。

ベイズ推定と点推定の違い

ベイズ推定は、データを見る前の θ についての知識を事前分布 $p(\theta)$ で表し、データを見た後の知識を事後分布 $p(\theta|x)$ として更新します ($p(\theta|x) \propto p(x|\theta)p(\theta)$)。事前分布には、通常は広い、つまりエントロピーの高い分布を選びます。何も知らない状態を表すためです。

MAP推定は、完全なベイズ推定の事後分布を使う代わりに、事後分布の最大点だけを使う点推定です。ベイズ推定は事後分布そのものを使うので、推定の不確かさまで扱えます。予測するときは、事後分布で重みを付けて平均をとり、 θ の1点には頼りません。そのぶん計算が重く、多くのモデルでは事後分布を使う計算が扱えない (intractable) ため、扱いやすい近似として1点の推定値を求めたい場面が多くあります。MAP推定は、そのための方法の1つです。^[^c1-2]

3つの推定法は、次の表で区別します。

推定法	使う量	返すもの	事前分布
最尤推定	尤度	θ の1点	使わない

推定法	使う量	返すもの	事前分布
MAP推定	尤度と事前分布	事後分布が最大の θ の1点	使う
ベイズ推定	尤度と事前分布	事後分布そのもの	使う

場面で手順を組み合わせる

学習データが少なく、特徴が多いモデルで、過学習が出ている場面を考えます。重みにガウス事前分布をおくMAP推定にすると、目的関数に重み減衰の項が入り、重みが小さく抑えられます。事前分布の分散を小さくすると、重み減衰の係数 λ は大きくなります。逆に、データを増やすと尤度の項が相対的に大きくなり、尤度の項に押されて、事前分布は決定打になりません。モデルの構造を変えたわけではなく、事前分布をどう置いたかが、正則化の形を決めています。

自己情報量とエントロピーの計算 [1.3.1]

情報理論は、確率の低い出来事ほど知らせる価値が大きい、という考えから始まります。必ず起こる出来事を知らされても、驚きはありません。

自己情報量は、1つの出来事 x の情報量です。

$$I(x) = -\log P(x)$$

確率が小さいほど大きく、確率が1なら0になります。 $\log$ の底で単位が変わります。 e を底にする自然対数なら単位は nat で、確率 $1/e$ の出来事を観測したときの情報量が1 nat です。底を2にすると単位は bit (shannon) で、nat と bit は定数倍の関係にあります ($1 \text{ nat} = 1/\ln 2 \text{ bit} \approx 1.443 \text{ bit}$)。ディープラーニングの教科書は、 $\log$ を自然対数としています。[^{c1-1}]

エントロピー (シャノンエントロピー) は、自己情報量の期待値です。

$$H(x) = E_{\{x \sim P\}}[I(x)] = -\sum_x P(x) \log P(x)$$

期待値は、出来事ごとの値を確率で重みを付けて平均したものです。エントロピーは、分布から出る出来事が、平均してどれだけ予測しにくいかを表します。ほぼ決まった結果しか出ない分布は低く、すべての結果が等しく出る一様な分布は高くなります。2値の変数なら、確率 $p=0.5$ のときに最大になります (値は1 bit)。^[^c1-1]

連続変数のときのシャノンエントロピーは、微分エントロピーと呼びます。和の中で $P(x)=0$ になる項は、 $0 \log 0 = 0$ と約束します。これは $\lim_{x \rightarrow 0} x \log x = 0$ が根拠で、確率0の結果が無視できることを意味します。^[^c1-1]

計算例

4つの結果が確率 $1/2, 1/4, 1/8, 1/8$ で出る分布を考えます。それぞれの自己情報量は、底2で 1, 2, 3, 3 bit です。エントロピーは $0.5 \times 1 + 0.25 \times 2 + 0.125 \times 3 + 0.125 \times 3 = 1.75$ bit。底を e にすると $1.75 \times \ln 2 \approx 1.213$ nat です。同じ4つの結果が一様に $1/4$ ずつなら、エントロピーは $\log_2 4 = 2$ bit で、これが4つの結果をもつ分布の上限になります。上の分布の1.75はそれより小さく、偏りがあるほど下がることが分かります。件数の表のままでは式に入れられません。先に合計で割って確率にします。

2値で $p=0.9$ のときは $-(0.9 \log_2 0.9 + 0.1 \log_2 0.1) \approx 0.469$ bit で、 $p=0.5$ の1 bit より小さくなります。 p と $1-p$ を入れ替えても式は同じなので、 $p=0.1$ でも 0.469 bit です。エントロピーは、取り得る結果の数が同じなら、一様なときに最大で、結果が1つに決まるときに0になります (負にはなりません)。結果

の数が3つなら、上限は $\log_2 3 \approx 1.585$ bit (自然対数なら $\ln 3 \approx 1.099$ nat) になります。

間違えやすいのは単位です。同じ分布のエントロピーは、底2で計算した値と、自然対数で計算した値が違います。選択肢に数値が並ぶ問題では、どちらの単位かを先に確かめます (単位の書かれていない数値は読めません)。

結合エントロピー・条件付きエントロピー・相互情報量の関係 [1.3.1]

2つの確率変数 X と Y をいっしょに見るときの量が、3つあります。結合エントロピーは、組 (X, Y) 全体のエントロピーです。

$$H(X, Y) = - \sum_x \sum_y P(x, y) \log P(x, y)$$

条件付きエントロピーは、 X を知ったあとに残る Y のエントロピーの平均です。

$$H(Y|X) = - \sum_x \sum_y P(x, y) \log P(y|x)$$

相互情報量は、片方を知ることによって、もう片方のエントロピーがどれだけ減るかを表します。

$$I(X; Y) = H(Y) - H(Y|X) = H(X) - H(X|Y)$$

$$H(X, Y) = H(X) + H(Y|X) = H(Y) + H(X|Y)$$

$$I(X; Y) = H(X) + H(Y) - H(X, Y)$$

最初の式は、組全体の不確かさが「Xの不確かさ」と「Xを知ったあとのYの不確かさ」の和だという意味で、連鎖律と呼ばれる関係です(chain rule)。2つ目の式は、別々のエントロピーの和から結合エントロピーを引いた分が、2つの変数の共有している情報だという意味です。XとYが独立なら $H(X,Y)=H(X)+H(Y)$ なので、 $I(X;Y)=0$ です。YがXから完全に決まるなら $H(Y|X)=0$ で、 $I(X;Y)=H(Y)$ になります。相互情報量は0以上で、条件を付けるとエントロピーは増えません。条件付きエントロピー $H(Y|X)$ は0以上 $H(Y)$ 以下なので、相互情報量の上限は $H(X)$ と $H(Y)$ の小さい方で、片方が他方から決まるときに達します。結合エントロピー $H(X,Y)$ は、 $H(X)$ と $H(Y)$ の大きい方以上、 $H(X)+H(Y)$ 以下に収まります。[^c1-8][^c1-15]

数表から計算する

XとYが次の同時分布をもつとします。 $P(0,0)=0.4$ 、 $P(0,1)=0.1$ 、 $P(1,0)=0.1$ 、 $P(1,1)=0.4$ です。周辺分布は、XもYも $(0.5, 0.5)$ で、 $H(X)=H(Y)=1$ bit になります。結合エントロピーは $-(2 \times 0.4 \log_2 0.4 + 2 \times 0.1 \log_2 0.1) \approx 1.722$ bit です。連鎖律から $H(Y|X) = 1.722 - 1 = 0.722$ bit、相互情報量は $I(X;Y) = 1 + 1 - 1.722 = 0.278$ bit となります(単位はどちらも bit)。Xを知ると、Yの不確かさは1 bit から0.722 bit に減り、減った分が0.278 bit です。

scikit-learn は、クラスタリングの評価で、同じ量を数え上げの形で定義しています。ラベルの割り当てUとVについて、次の式です。

$$MI(U,V) = \sum_i \sum_j (|U_i \cap V_j|/N) \cdot \log(|U_i \cap V_j| / (|U_i| |V_j|))$$

自然対数を使い、ラベルの付け替えに不変で、UとVを入れ替えても同じ値になります(対称)。([^c1-7] 同じ文書は、条件付きエントロピーを $H(C|K) = -\sum_c \sum_k (n_{\{c,k\}}/n) \log(n_{\{c,k\}}/n_k)$ と定義し、homogeneity $h = 1 - H(C|K)/H(C)$ 、completeness $c = 1 - H(K|C)/H(K)$ として使います。[^c1-8]

向きと対称性

相互情報量は対称です。 $H(Y|X)$ と $H(X|Y)$ は、一般に値が違います。それでも $H(Y) - H(Y|X)$ と $H(X) - H(X|Y)$ は、同じ値になります。条件付きエントロピーを問う問題では、どちらの変数を条件にしているかを確認めます。条件が逆だと、答えが変わります ($H(Y|X)$ と $H(X|Y)$ は別の量)。

独立になる例も見ておきます。4つの組がすべて確率 0.25 なら、周辺分布は $(0.5, 0.5)$ のままで、 $H(X, Y) = 2$ bit、 $H(X) + H(Y) = 2$ bit です。連鎖律から $H(Y|X) = 1$ bit で、 $I(X; Y) = 0$ です。X を知っても、Y の不確かさは減りません。逆に、Y が X のコピーなら $H(X, Y) = 1$ bit で、 $I(X; Y) = 1 + 1 - 1 = 1$ bit となり、片方を知ればもう片方の不確かさは0になります。同じ周辺分布でも、結合の形で相互情報量は0から上限まで動き、周辺分布からは決まりません。

クロスエントロピー (分類の損失との対応) [1.3.1]

クロスエントロピーは、真の分布 P で出る出来事を、別の分布 Q の自己情報量で測った期待値です。

$$H(P, Q) = -E_{\{x \sim P\}}[\log Q(x)] = H(P) + D_{\text{KL}}(P \parallel Q)$$

右辺の2つ目の式が要点です。クロスエントロピーは、 P 自身のエントロピーにKLダイバージェンスを足したものです。 Q について最小化するとき、 $H(P)$ は Q に関係しないので、クロスエントロピーの最小化とKLダイバージェンスの最小化は、同じ最適点を返します。[1]

数値で確かめます。 $P=(0.5, 0.5)$ 、 $Q=(0.9, 0.1)$ とすると、クロスエントロピーは $-(0.5 \ln 0.9 + 0.5 \ln 0.1) \approx 1.204$ nat です。 $H(P) = \ln 2 \approx 0.693$ nat、 $D_{\text{KL}}(P \parallel Q) \approx 0.511$ nat で、2つの和 $0.693 + 0.511 = 1.204$ と一致します。

分類の損失になる理由

分類では、正解が1つのクラスに決まっているデータの P は、正解クラスだけが確率1の、one-hot な分布です。このとき $H(P)=0$ なので、クロスエントロピーは $-\log Q(\text{正解クラス})$ に縮みます ($H(P)$ の項が消えるため)。これが、正解クラスに割り当てた確率の負の対数をとる、分類の損失の形です。正解の確率が高いほど小さく、低いほど大きくなります。データ全体では、各データの値を平均します。

多クラス分類では、モデルの出力する生の値 (logits) をソフトマックスで確率に直します。PyTorch の `CrossEntropyLoss` は、この処理を損失の内側に含んでいるため、自分でソフトマックスを掛ける必要はありません。n 番目のデータの損失は、次の式です。

$$l_n = -w_{\{y_n\}} \cdot \log(\exp(x_{\{n,y_n\}}) / \sum_c \exp(x_{\{n,c\}}))$$

入力正規化されていない logits で、確率ではありません。ソフトマックスを自分で掛けた確率を渡すと、二重にソフトマックスを掛けた損失になり、意図した値になりません。既定の引数は `weight=None, ignore_index=-100, reduction='mean', label_smoothing=0.0` です。ラベルを1つのクラスの番号ではなく、クラスごとの確率で与える形も使えます。[^c1-9]

数値例です。3クラスの logits が $(2.0, 1.0, 0.1)$ で、正解がクラス0のとき、ソフトマックスは $(0.659, 0.242, 0.099)$ になり、損失は $-\ln 0.659 \approx 0.417$ です。正解がクラス1だったとすると、損失は $-\ln 0.242 \approx 1.417$ に増えます。同じ出力でも、正解が出力の1位と一致するかどうかで、損失が大きく変わります。

2値分類

2値分類では、出力 x を確率とし、正解 y を 0 か 1 として、次の式を使います (バイナリクロスエントロピー)。

$$l_n = -w_n [y_n \cdot \log x_n + (1-y_n) \cdot \log(1-x_n)]$$

PyTorch の `BCELoss` はこの式で、入力が確率であることを前提にします。 $\log$ の出力は -100 以上に切り上げられ、損失が有限に保たれます。 $x=0$ のときの $\log$ が負の無限大になって、勾配が壊れるのを防ぐためです。^[^c1-10] 例えば $y=1$ で出力が 0.8 なら損失は $-\ln 0.8 \approx 0.223$ 、 $y=0$ で出力が 0.8 なら $-\ln(1-0.8) \approx 1.609$ です。

シラバス2026 では、クロスエントロピー誤差の項が、2値分類から多クラス分類の形へ移されました。^[^c1-14] 多クラス分類の損失はクロスエントロピー誤差、2値の場合はバイナリクロスエントロピーと呼び分けます。

場面で手順を組み合わせる

画像分類のモデルを、正解ラベルを使って学習する場面です。損失はクロスエントロピーで、出力はソフトマックスを通した確率として得られます。これは、データの経験分布 (one-hot) とモデルの出力分布のKLダイバージェンスを最小にしていること、さらに最尤推定そのものであることを意味します。one-hot の $H(P)$ は0なので、KLダイバージェンスとクロスエントロピーの値も一致し、差は生じません。ラベルをなだらかにした分布を使うと $H(P)$ が0でなくなり、損失の値はKLより大きくなりますが、最適な Q は同じです。値の比較と、最適点の比較は、分けて判断します。

KLダイバージェンスとJSダイバージェンス (非対称性・値の範囲)

[1.3.1]

KLダイバージェンスは、2つの分布 P と Q のずれを測る量です。

$$D_{\text{KL}}(P \parallel Q) = E_{\{x \sim P\}} [\log P(x) - \log Q(x)] = \sum_x P(x) \log(P(x)/Q(x))$$

0以上で、 P と Q が同じ分布のとき(連続ならほとんど至る所で等しいとき)だけ0になります。対称ではないので、 $D_{\text{KL}}(P \parallel Q) \neq D_{\text{KL}}(Q \parallel P)$ になることがあり、距離ではありません。 P が正で Q が 0 になる結果が 1 つでもあれば $D_{\text{KL}}(P \parallel Q)$ は無限大ですが、逆向きの $D_{\text{KL}}(Q \parallel P)$ はその項が $0 \log 0 = 0$ で消え、有限の値になることがあります。[¹c1-1]

数値で確かめます。 $P=(0.5, 0.5)$ 、 $Q=(0.9, 0.1)$ のとき、 $D_{\text{KL}}(P \parallel Q) = 0.5 \ln(0.5/0.9) + 0.5 \ln(0.5/0.1) \approx 0.511$ nat です。向きを入れ替えると、 $D_{\text{KL}}(Q \parallel P) = 0.9 \ln(0.9/0.5) + 0.1 \ln(0.1/0.5) \approx 0.368$ nat になります。値が違うので、向きを変えれば答えも変わります。

向きによる近似の違い

分布 p を、別の分布 q (例えばガウス分布1つ) で近似する場面を考えます(p は山が2つの混合分布とします)。 $D_{\text{KL}}(p \parallel q)$ を最小にする q は、 p が高確率の場所に高い確率を置こうとします。 p が複数の山をもつときは、 q が山をまたいで広がり、全部を覆います。 $D_{\text{KL}}(q \parallel p)$ を最小にする q は、 p が低確率の場所を避けるので、山が離れていれば、1つの山にだけ寄ります。前者は覆い、後者は選ぶ、と覚えます。期待値をとる分布が p か q かで、 q への要求が決まります(向きを取り違えると結論が逆になる点に注意)。^[1]c1-1]

JSダイバージェンス

JSダイバージェンスは、KLを対称にした量です。P と Q の平均の分布 $M = (P+Q)/2$ を作り、それぞれを M と比べます。

$$JS(P \parallel Q) = (1/2) D_{KL}(P \parallel M) + (1/2) D_{KL}(Q \parallel M)$$

対称で、 $JS(P \parallel Q) = JS(Q \parallel P)$ です。値は $0 \leq JS \leq \log 2$ に収まります。上限 $\log 2$ (自然対数なら約0.693、底2なら1 bit) に達するのは、P と Q の確率が正の場所が全く重ならないときだけで、少しでも重なれば上限には届きません。この場合、KLは無限大になりますが、JSは M を介するので有限に保たれます。P=(0.5, 0.5)、Q=(0.9, 0.1) では、M=(0.7, 0.3) となり、 $JS \approx 0.102 \text{ nat}$ です。P=(1, 0)、Q=(0, 1) なら、M=(0.5, 0.5) で、 $JS = \ln 2 \approx 0.693$ になります。ここで挙げた範囲は、係数 1/2 を付ける定義での値です。

流儀の違いに注意が要ります(係数 1/2 の有無)。TensorFlow Probability の `jensen_shannon` は、 $KL[p, m] + KL[q, m]$ と書き、ドキュメントの式には係数 1/2 が付いていません。^[^c1-12] 1/2 を付ける定義なら上限は $\log 2$ 、付けない定義なら上限は $2 \log 2$ です。選択肢の数値が上限を問うときは、どちらの定義かを問題文から読み取ります。

フレームワークの実装

PyTorch の `KLDivLoss` は、入力の渡し方に決まりがあります。input は確率ではなく、対数をとった値で、target は確率です(既定の `log_target=False`)。点ごとの損失は `target*(target.log() - input)` です。既定の `reduction='mean'` は、真のKLダイバージェンスを返しません。数学的なKLダイバージェンスと一致するのは、合計をバッチサイズで割る `'batchmean'` です。^[^c1-11]

```
import torch

p = torch.tensor([[0.5, 0.5]])          # target: 確率
q = torch.tensor([[0.9, 0.1]])
loss = torch.nn.KLDivLoss(reduction="batchmean")
print(loss(q.log(), p))                  # input は log q、target は p
```

この例は $D_{KL}(p \parallel q) \approx 0.511$ を返します。`reduction='mean'` なら、全要素(バッチ1×クラス2)の平均になり、約 0.255 に半減します。input と target を逆に渡すと、別の向きのKLになります。input に確率をそのまま渡すのは誤りで、log をとる前の値では、損失が意味を持ちません。

TensorFlow の Keras の `KLDivergence` は、`loss = y_true * log(y_true / y_pred)` で、`y_true` と `y_pred` が確率分布であることを前提にします。値は $[0, 1]$ の範囲に切り詰められます。^[^c1-13]

場面で手順を組み合わせる

生成モデルが、データ分布 p に似た分布 q を作る場面を考えます。学習の目的関数に $D_{KL}(p \parallel q)$ を使うと、 q は p の全体を覆うように広がります。 $D_{KL}(q \parallel p)$ を使うと、 q は p の山の1つだけを深く捉えます。 p と q の台が重ならない状況では、KLが無限大になり、勾配が意味を持ちません。JSなら有限の値を保ち、上限の $\log 2$ に張り付きます。KLの向きの選択と、JSの有限性は、別々の判断です。

サンプルはここまでです

続き(第2章～巻末)は、E資格 問題集のご購入で、頻出度別ノート・暗記用ノートと合わせて3点すべてダウンロードできます。

SAMPLE

E資格 学習用テキスト(無料サンプル)

版

2026年10月版(JDLA E資格 試験出題範囲(シラバス2026・第1.0版・E2026#2～) 準拠)

発行

IT資格道場(<https://www.shikaku-doujou.com>)

お問い合わせ

info@shikaku-doujou.com

E資格(JDLA Deep Learning for ENGINEER)およびG検定は、一般社団法人日本ディープラーニング協会(JDLA)の商標または登録商標です。記載されているその他の会社名・製品名・サービス名は、各社の商標または登録商標です。

本教材は公式の試験出題範囲(シラバス2026)の内容を基に作成したもので、日本ディープラーニング協会(JDLA)との提携・承認関係はありません。実際の試験問題の転載・復元は行っていません。

本文・図表・設問はすべてオリジナルの著作物です。無断での転載・複製・再配布・二次販売はできません。お気づきの点は info@shikaku-doujou.com までご連絡ください。