CSE 5523: Lecture Notes 5
Decision Theory

Contents

5.1 Decision procedures / policies

Given a model and training data, we have options for making classification/regression decisions a.

This lets us talk about utility (value / negative cost) without confusing it with epistemology (truth).

(E.g., we prefer false positive to false negative in a cancer screen, even if estimator is more wrong.)

Estimation decision actions a come from **estimator** functions δ based on variable values $x_1, \ldots, x_V$:

$$a = \delta(x_1, \ldots, x_V)$$

This estimator (or 'decision procedure') can then be defined to minimize expected loss $\mathsf{L}(y, a)$:

$$\delta_{p_1,p_2,\ldots}(x_1, \ldots, x_V) = \operatorname*{argmin}_a \mathsf{E}_{y \sim \mathsf{P}_{p_1,p_2,\ldots}(y \mid x_1,\ldots,x_V)} \mathsf{L}(y, a)$$

($\mathsf{E}_{y\sim\mathsf{P}(y)} f(y) = \sum_y \mathsf{P}(y) \cdot f(y)$ is the expected value of $f(y)$, weighted by $\mathsf{P}(y)$.)

We can define different **loss functions**, which measure the cost (−utility) of wrong decisions:

- **zero-one loss** — lose a point for each wrong answer:

$$\mathsf{L}_{0,1}(y, a) = [\![y \neq a]\!] = \begin{cases} 0 & \text{if } a = y \\ 1 & \text{if } a \neq y \end{cases},$$

- **zero-one loss with reject** — lose fewer points if we admit we don't know (a_{REJECT}):

$$\mathsf{L}_{0,\lambda_R,\lambda_S}(y, a) = \begin{cases} 0 & \text{if } a = y \\ \lambda_R & \text{if } a = a_{\text{REJECT}} \\ \lambda_S & \text{if } a \neq y \wedge a \neq a_{\text{REJECT}} \end{cases},$$

- **false-positive false-negative loss** — lose different points for false positive/negative:

$$\mathsf{L}_{\text{FP,FN}}(y, a) = \begin{cases} 0 & \text{if } a = y \\ \lambda_{\text{FP}} & \text{if } a{=}1 \wedge y{=}0 \\ \lambda_{\text{FN}} & \text{if } a{=}0 \wedge y{=}1 \end{cases},$$

- **linear/absolute loss** — lose the difference between the estimate and the training example:
  $\mathsf{L}_1(y, a) = |y - a|$,
- **quadratic loss** — lose the square of the difference between estimate and training example:
  $\mathsf{L}_2(y, a) = (y - a)^2$.
- **negative log loss** — lose the neg. log of the difference betw. estimate and training example:
  $\mathsf{L}_{\mathrm{NL}}(y, a) = -\ln(a)$ (used for probabilities; goal probability of training example is 1).

We can still define estimators to output a-posteriori most probable outcomes for y as a:

$$\begin{aligned}\delta_{p_1,p_2,\ldots}(x_1,\ldots,x_V) &= \operatorname*{argmin}_a \mathsf{E}_{y\sim\mathsf{P}_{p_1,p_2,\ldots}(y\,|\,x_1,\ldots,x_V)}\,\mathsf{L}(y,a)\\ &= \operatorname*{argmin}_a \sum_y \mathsf{P}_{p_1,p_2,\ldots}(y\,|\,x_1,\ldots,x_V)\cdot\mathsf{L}_{0,1}(y,a) \qquad \text{(for discrete } y\text{)}\\ &= \operatorname*{argmin}_a \int \mathsf{P}_{p_1,p_2,\ldots}(y\,|\,x_1,\ldots,x_V)\cdot\mathsf{L}_2(y,a)\,dy \qquad \text{(for continuous } y\text{)}\end{aligned}$$

This will choose the y with the maximum probability to avoid losses on other y outcomes.

5.2 Decision-theoretic parameter estimation

We can also use these estimators to estimate parameters $p_1, p_2, \ldots$ as a:

$$\delta_{h_1,h_2,\ldots}(\mathcal{D}) = \operatorname*{argmin}_a \mathsf{E}_{p_1,p_2,\ldots\sim\mathsf{P}_{h_1,h_2,\ldots}(p_1,p_2,\ldots\,|\,\mathcal{D})}\,\mathsf{L}_{\mathrm{GE}}(\langle p_1,p_2,\ldots\rangle, a)$$

This uses a **generalization error** loss function, which itself contains another estimator δ for y:

$$\mathsf{L}_{\mathrm{GE}}(\langle p_1,p_2,\ldots\rangle, a) = \mathsf{E}_{y,x_1,\ldots,x_V\sim\mathsf{P}_{p_1,p_2,\ldots}(y,x_1,\ldots,x_V)}\,\mathsf{L}(y,\delta_a(x_1,\ldots,x_V))$$

So, substituting these and using the definition of expected value produces a big marginal:

$$\begin{aligned}\delta_{h_1,h_2,\ldots}(\mathcal{D}) &= \operatorname*{argmin}_a \mathsf{E}_{p_1,p_2,\ldots\sim\mathsf{P}_{h_1,h_2,\ldots}(p_1,p_2,\ldots\,|\,\mathcal{D})}\,\mathsf{E}_{y,x_1,\ldots,x_V\sim\mathsf{P}_{p_1,p_2,\ldots}(y,x_1,\ldots,x_V)}\,\mathsf{L}(y,\delta_a(x_1,\ldots,x_V))\\ &= \operatorname*{argmin}_a \int \mathsf{P}_{h_1,h_2,\ldots}(p_1,p_2,\ldots\,|\,\mathcal{D})\cdot\sum_{y,x_1,\ldots,x_V}\mathsf{P}_{p_1,p_2,\ldots}(y,x_1,\ldots,x_V)\cdot\mathsf{L}(y,\delta_a(x_1,\ldots,x_V))\,dp_1dp_2\ldots\end{aligned}$$

This allows us to define parameters $p_1, p_2, \ldots$ that respect our loss function over y.

5.3 Evaluation and visualization

If we want to evaluate a set of binary estimators for some threshold τ on test data:

$$\delta(x_1,\ldots,x_V) = \begin{cases}1 & \text{if } \mathsf{P}(y{=}1\,|\,x_1,\ldots,x_V) > \tau\\ 0 & \text{otherwise}\end{cases}$$

we can't always compare these estimators because they may be optimal at different thresholds τ.

But we can graph the rate of false positives ($a{=}1, y{=}0$) and true positives ($a{=}1, y{=}1$) over all τ.

This is called a **receiver operator (ROC) curve**.

If one estimator's curve is more bowed out toward $(0, 1)$ than another, we may prefer that estimator.

Sometimes we have a large space of vastly more $y{=}0$ examples (e.g. faces in possible rectangles).

In this case we can graph the **recall** (R) and **precision** (P) over all τ for examples i in $\mathcal{D}$:

$$
\begin{aligned}
P &= \frac{\sum_i [\![a_i{=}1 \wedge y_i{=}1]\!]}{\sum_i [\![a_i{=}1]\!]} \\
R &= \frac{\sum_i [\![a_i{=}1 \wedge y_i{=}1]\!]}{\sum_i [\![y_i{=}1]\!]}
\end{aligned}
$$

This is called a **precision-recall** curve.

Likewise, if one curve is more bowed out toward $(1, 1)$ than another, we may prefer that estimator.

We can also report **F scores**, which are the harmonic mean of recall and precision:

$$
\begin{aligned}
F_1 &= \frac{2}{\frac{1}{P} + \frac{1}{R}} && \text{(harmonic mean: inverse of average of inverses)} \\
&= \frac{2RP}{R + P} && \text{(multiply numerator and denominator by } RP\text{)} \\
&= \frac{2\sum_i [\![a_i{=}1 \wedge y_i{=}1]\!]}{\sum_i [\![a_i{=}1]\!] + \sum_i [\![y_i{=}1]\!]} && \text{(from line 1, substituting definitions)}
\end{aligned}
$$

Sometimes we want to evaluate multiple hypotheses y_i on a common dataset $\mathcal{D}$.

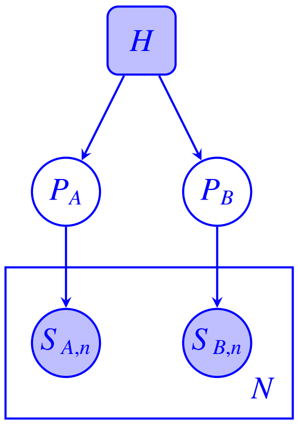
In this case we can report a **false discovery rate** (FDR):

$$
FDR(\tau, \mathcal{D}) = \frac{\sum_i \mathsf{P}(y_i{=}0 \mid \mathcal{D}) \cdot [\![\mathsf{P}(y_i{=}1 \mid \mathcal{D}) > \tau]\!]}{\sum_i [\![\mathsf{P}(y_i{=}1 \mid \mathcal{D}) > \tau]\!]}
$$

5.4 Bayesian vs. frequentist hypothesis testing

Remember earlier we did **Bayesian hypothesis testing**?

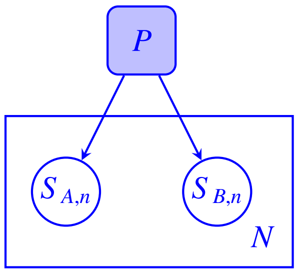
We drew several samples of parameters, and counted the number that satisfied $p_B > p_A$:



You'll also see **frequentist hypothesis testing**, maybe a lot.

In frequentist statistics, parameters aren't random variables, they're fixed properties of data.

We then test hypotheses by drawing samples of *the data* s, then comparing them to the real data $\tilde{s}$:



For example, the data may be classifier scores, and the comparison may be $\frac{\sum_i \llbracket s_{B,i} > s_{A,i} \rrbracket}{N} > \frac{\sum_i \llbracket \tilde{s}_{B,i} > \tilde{s}_{A,i} \rrbracket}{N}$.

An easy way to do this is to randomly swap each pair i of A and B scores in the real data.

This is called **permutation testing**.

It ensures the data are drawn from the same distribution, even if you don't know its parameters.

For this reason, it's called a **non-parametric test**.

5.5 Sample code

Here's sample code for the permutation test:

```
import sys
import numpy
import pandas

SS = pandas.read_csv(sys.argv[1])

numsamples = 1000
randWins = 0
for i in range(numsamples):
  trialsBwins = 0
  for n,(sa,sb) in SS.iterrows():
    if numpy.random.binomial(p=.5,n=1)>=.5: trialsBwins = trialsBwins + (1 if sa>sb else 0)
    else:                                   trialsBwins = trialsBwins + (1 if sb>sa else 0)
  if trialsBwins >= len(SS[ SS['scoreB']==1 ]) - len(SS[ SS['scoreA']==1 ]): randWins = randWins + 1

print( 'Probability of same or better score due to chance: ' + str(randWins/numsamples) )
```

Run on our small set:

```
scoreA,scoreB
0,0
0,0
0,0
0,1
0,1
0,1
1,0
1,1
```

we get a high probability that the results are due to chance:

```
Probability of same or better score due to chance: 0.688
```

If we run it on a larger test set (this is just twice as many of each outcome):

```
scoreA,scoreB
0,0
0,0
0,0
0,0
0,0
0,0
0,1
0,1
0,1
0,1
0,1
0,1
1,0
1,0
1,1
1,1
```

we get a lower probability that the results are due to chance:

```
Probability of same or better score due to chance: 0.64
```