Pattern Recognition

Bayesian Decision Theory

Recognition

The model

Let two random variables be given:

- The first one is typically discrete (i.e. $k \in K$) and is called “class”
- The second one is often continuous ($x \in \mathbb{R}^n$) and is called “observation”

Let the joint probability distribution $p(x, k)$ be “given”.
As k is discrete it is often specified by $p(x, k) = p(k) \cdot p(x|k)$

The recognition task: given x, estimate k.

Usual problems (questions):

- How to estimate k from x ? (**today**)
- The joint probability is not always explicitly specified.
- The set K is sometimes huge (remember the Hopfield-Networks)

Idea – a game

Somebody samples a pair (x, k) according to a p.d. $p(x, k)$

He keeps k hidden and presents x to **you**

You decide for some k^* according to a chosen **decision strategy**

Somebody penalizes your decision according to a **Loss-function**, i.e. he compares your decision to the "true" hidden k

You know both $p(x, k)$ and the Loss-function (how does he compare)

Your goal is to design the decision strategy in order to pay as less as possible in average.

Bayesian Risk

Notations:

The **decision set** D . Note: it needs not to coincide with K !!!
Examples – the decision “I don’t know”, “surely not this class” etc.

Decision strategy (mapping) $e : X \to D$

Loss-function $C : D \times K \to \mathbb{R}$

The **Bayesian Risk** – the expected loss:

$$R(e) = \sum_x \sum_k p(x,k) \cdot C\Big(e(x), k\Big) \to \min_e$$

(should be minimized with respect to the decision strategy)

Some special cases

General:

$$R(e) = \sum_x \sum_k p(x,k) \cdot C\Big(e(x), k\Big) \to \min_e$$

Almost always:
decisions can be made for different x independently (the set of decision strategies is not restricted). Then:

$$R\Big(e(x)\Big) = \sum_k p(x,k) \cdot C\Big(e(x), k\Big) \to \min_{e(x)}$$

Very often:
the decision set coincides with the set of classes, i.e. $D = K$

$$k^* = \arg\min_k \sum_{k'} p(x,k') \cdot C(k,k')$$

Maximum A-posteriori Decision (MAP)

The Loss is the simplest one:

$$C(k, k') = \mathbb{1}(k \neq k')$$

i.e. we pay 1 if the answer is not the true class, no matter what error we make.

From that follows:

$$\begin{aligned} R(k) &= \sum_{k'} p(k'|x) \cdot \mathbb{1}(k \neq k') = \\ &= \sum_{k'} p(k'|x) - p(k|x) = 1 - p(k|x) \to \min_k \end{aligned}$$

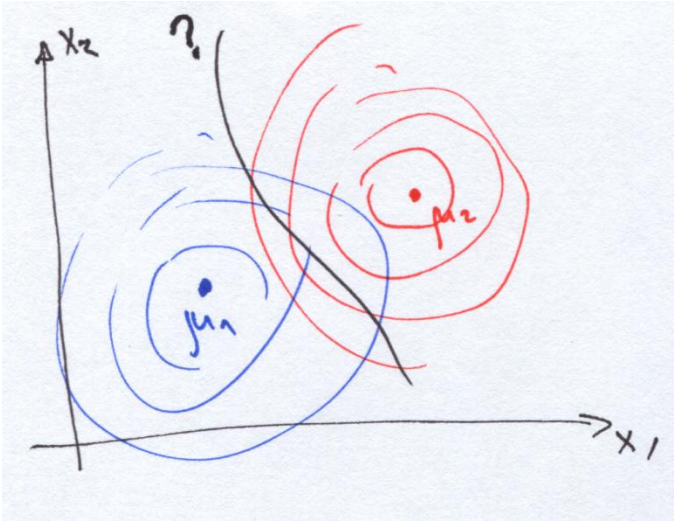
$$p(k|x) \to \max_k$$

A MAP example

Let $K = \{1, 2\}$, $x \in \mathbb{R}^2$, $p(k)$ be given. Conditional probability distributions for observations given classes are Gaussians:

$$p(x|k) = \frac{1}{2\pi\sigma_k^2} \exp\left[-\frac{\|x - \mu_k\|^2}{2\sigma_k^2}\right]$$

The loss-function is $\mathbb{1}(k \neq k')$,
i.e. we wont MAP.



The decision strategy (the mapping $e : X \to K$) partitions the input space into two regions: the one corresponding to the first and the one corresponding to the second class. How this partition looks like?

A MAP example

For a particular x we decide for 1, if

$$p(1) \cdot \frac{1}{2\pi\sigma_1^2} \exp\left[-\frac{\|x-\mu_1\|^2}{2\sigma_1^2}\right] > p(2) \cdot \frac{1}{2\pi\sigma_2^2} \exp\left[-\frac{\|x-\mu_2\|^2}{2\sigma_2^2}\right]$$

Special case (for simplicity) $\sigma_1 = \sigma_2$
→ the decision strategy is (derivation on the board)

$$\langle x, \mu_2 - \mu_1 \rangle > const$$

→ a linear classifier – the hyperplane that is orthogonal to $\mu_2 - \mu_1$

More classes, equal σ and $p(k)$ → Voronoi-diagram
More classes, equal σ and different $p(k)$ → Fischer-classifier
Two classes, different σ → a quadratic curve
etc.

Decision with rejection

The decision set is $D = K \cup \{rw\}$, i.e. extended by a special decision "I don't know". The loss-function is

$$C(d,k) = \begin{cases} \mathbb{1}(d \neq k) & \text{wenn} \quad d \in K \\ \varepsilon & \text{wenn} \quad d = rw \end{cases}$$

– we pay a (reasonable) penalty if we are lazy to decide.

Case-by-case analysis:

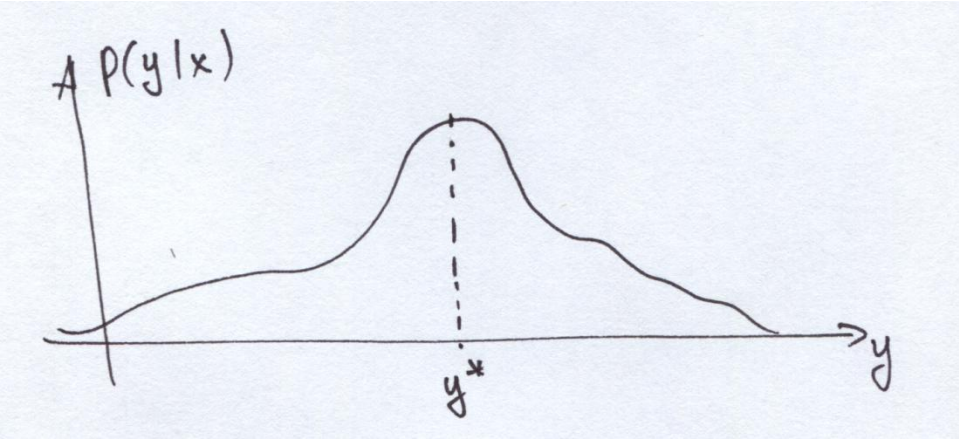
1. We decide for a class $d \in K$, decision is MAP $d = k^*$, the loss for this is $1 - p(k^*|x)$
2. We decide to reject $d = rw$, the loss for this is ε

→ Compare $p(k^*|x)$ with $1 - \varepsilon$ and decide for the variant with greater value.

Other simple loss-functions

Let the set of classes be **structured**

Example:
The probability distribution is $p(x, y)$ with observations x and **continuous** hidden value $y \in \mathbb{R}$. Suppose, we know $p(y|x)$ for a given x for which we would like to infer y^* .



The Bayesian Risk reads:

$$R\Big(d(x)\Big) = \int_{-\infty}^{\infty} p(y|x) \cdot C\Big(d(x), y\Big)\, dy$$

Other simple loss-functions

Simple delta-loss-function → MAP (not interesting anymore)

Loss may account for differences between the decision and the "true" hidden value, for instance $C(d, y) = (d - y)^2$ i.e. we pay depending on the **distance**. Than (see board again):

$$
\begin{aligned}
e(x) &= \arg\min_d \int_{-\infty}^{\infty} p(y|x) \cdot (d - y)^2 \, dy \\
&= \int_{-\infty}^{\infty} y \cdot p(y|x) \, dy = \mathbb{E}_{p(y|x)} y
\end{aligned}
$$

Other choices:
$C(d, y) = |d - y|$, $C(d, y) = \mathbb{I}(|d - y| > \delta)$, combination with "rejection" etc.

Additive loss-functions – an example

A – Aufgaben, B – Berater

	A1	A2	A3	A4	...
B1	1	1	1	1	...
B2	1	0	0	1	...
B3	1	0	1	0	...
B4	0	0	1	1	...
B5	1	1	1	0	...
B6	0	1	1	1	...
...	...	...	...	...	...
	*	*	*	*	...

Variante 1:

- Wähle den **besten** Berater (und vergesse alle anderen)
- Übernehme seine Antworten

Variante 2:

- Betrachte jede Aufgabe extra
- Schaue (gewichtet), was **alle** Berater dazu sagen

„Berater“ sind Zustandsfolgen,

„Aufgaben“ sind statistische Variablen (z.B. Zeitpunkte in HMM)

Die entscheidende „Besonderheit“:
die Menge der Entscheidungen (Klassen) ist strukturiert – die Menge der Zustandsfolgen

Additive loss-functions

Die Klasse ist ein Vektor $\bar{k} = (k_1, k_2, \ldots, k_n) \in K^n$, die Entscheidungsmenge sei $D = K^n$
Die a-posteriori Wahrscheinlichkeit $p(k_1, k_2, \ldots, k_n|x)$ sei „bekannt“.

Variante 1: MAP, d.h. $C(\bar{k}, \bar{k}') = \mathbb{1}(\bar{k} \neq \bar{k}')$

$$\bar{k}^* = (k_1, k_2, \ldots, k_n)^* = \arg\max_{\bar{k}} p(k_1, k_2, \ldots, k_n|x)$$

Die Kostenfunktion berücksichtigt nicht, in wieweit sich die Vektoren unterscheiden.

Variante 2: Kostenfunktionen gibt es für jeden Element k_i:

$$C(\bar{k}, \bar{k}') = \sum_i c_i(k_i, k_i')$$

$$R(\bar{k}) = \sum_{\bar{k}'} \left[p(\bar{k}'|x) \cdot \sum_i c_i(k_i, k_i') \right] = \sum_i \sum_{\bar{k}'} c_i(k_i, k_i') \cdot p(\bar{k}'|x) =$$

$$= \sum_i \sum_k \sum_{\bar{k}' : k_i' = k} c_i(k_i, k) \cdot p(\bar{k}'|x) =$$

$$= \sum_i \sum_k c_i(k_i, k) \sum_{\bar{k}' : k_i' = k} p(\bar{k}'|x) = \sum_i \sum_k c_i(k_i, k) p(k_i' = k|x)$$

Additive loss-functions

$$R(\bar{k}) = \sum_i \sum_k c_i(k_i, k) p(k'_i = k|x) \to \min_{\bar{k}}$$

$$\sum_k c_i(k_i, k) p(k'_i = k|x) \to \min_{k_i} \quad \forall i$$

1) Man berechne

$$p(k_i = k) = \sum_{\bar{k}: k_i = k} p(\bar{k}) \quad \forall i, k$$

2) Man treffe die Entscheidung für alle Elemente „unabhängig“

$$k_i^* = \arg\min_k \sum_{k'} p(k'_i = k) \cdot c_i(k, k')$$

Spezialfall: $c(k, k') = \mathbb{1}(k \neq k')$

$C(\bar{k}, \bar{k}') = \sum_i \mathbb{1}(k_i \neq k'_i)$ heißt **Hamming-Abstand**
– die Anzahl der falsch klassifizierten Variablen

$\Rightarrow$ **Max-Marginal Entscheidung** $k_i^* = \arg\max_k p(k'_i = k)$