Decision Theory

Lecture 8

Time to make a decision…

Exercise inspired by Mausam, University of Washington, CSE573

State of Nature

Action	Poor market performance **Payoff**	Good market performance **Payoff**
Buy Apple	-1,000	1,700
Buy Google	-2,000	2,100
Buy bonds	10	10

How to invest?

Maximax

Optimism

Select the maximum of the maximum payoff

	State of Nature		**Criterion**
	Poor market performance **Payoff**	Good market performance **Payoff**	Maximum payoff for an action
Action Buy Apple	-1,000	1,700	1,700
Action Buy Google	-2,000	2,100	2,100 ← **Maximax**
Action Buy bonds	10	10	10

Maximin

Pessimism

Select the maximum of the minimum payoffs

		State of Nature		**Criterion**
		Poor market performance **Payoff**	Good market performance **Payoff**	Minimum payoff for an action
Action	Buy Apple	-1,000	1,700	-1,000
	Buy Google	-2,000	2,100	-2,000
	Buy bonds	10	10	10

← **Maximin**

Minimax

Select the minimum maximum regret

		State of Nature				Criterion
		Poor market performance		Good market performance		Maximum regret for an action
		Payoff	**Regret**	**Payoff**	**Regret**	
Action	Buy Apple	-1,000	1,010	1,700	400	1,010
	Buy Google	-2,000	2,010	2,100	0	2,010
	Buy bonds	10	0	10	2,090	2,090

← **Minimax**

Which decision would I regret least?

Regret = Opportunity loss

Next: factor in probabilities of different outcomes

Expected Payoff: Equal likelihood

Action	State of Nature: Poor market performance Payoff	State of Nature: Good market performance Payoff	Criterion: Expected reward/ payoff
Buy Apple	-1,000	1,700	350
Buy Google	-2,000	2,100	50
Buy bonds	10	10	10
State Probability:	0.5	0.5	

Select the highest average payoff ASSUMING all states are of equal probability

Maximum Expected Reward

Expected Payoff

		State of Nature		Criterion
		Poor market performance **Payoff**	Good market performance **Payoff**	Expected reward/ payoff
Action	Buy Apple	-1,000	1,700	-190
	Buy Google	-2,000	2,100	-770
	Buy bonds	10	10	10
State Probability:		0.7	0.3	

Select the highest average payoff assuming state probabilities from prior knowledge

Maximum Expected Reward

Decision making design pattern

1. Define a measure of risk or reward

2. Select the action that optimizes that metric

Notation

$EV(a_i) = V(a_i|s_0)P(s_0) + V(a_i|s_1)P(s_1)$

Expected reward / payoff

State of Nature (s)

	Poor market performance $s = s_0$	Excellent market performance $s = s_1$	**Expected Reward** $EV(a_i)$
Action: Buy Apple $a = a_0$	$V(a_0\|s_0)$ -1,000	$V(a_0\|s_1)$ 1,700	$(0.7)(-1000) + (0.3)(1700)$ $\mathbf{= -190}$
Buy Google $a = a_1$	$V(a_1\|s_0)$ -2,000	$V(a_1\|s_1)$ 2,100	$(0.7)(-2000) + (0.3)(2100)$ $\mathbf{= -770}$
Buy bonds $a = a_2$	$V(a_2\|s_0)$ 10	$V(a_2\|s_1)$ 10	$(0.7)(10) + (0.3)(10)$ $\mathbf{= 10}$

State Probability: $P(s_0) =$ 0.7 $P(s_1) =$ 0.3

Risk = expected loss
(cost)

Loss: $\lambda(a_i|s_j) \triangleq$ Loss incurred by choosing action i and the state of nature being state j

Risk: $R(a_i) = \sum_{j=1}^{N_s} \lambda(a_i|s_j) P(s_j)$
Expected loss

Goal: Select action i for which $R(a_i)$ is minimum

Payoff

State of Nature

Action	Poor market performance	Good market performance
Buy Apple	-1,000	1,700
Buy Google	-2,000	2,100
Buy bonds	10	10

Loss

(here we define loss in terms of opportunity cost)

State of Nature

Action	Poor market performance	Good market performance
Buy Apple	1,010	400
Buy Google	2,010	0
Buy bonds	0	2,090

Investments: loss

$R(a_i) = \lambda(a_i|s_0)P(s_0) + \lambda(a_i|s_1)P(s_1)$

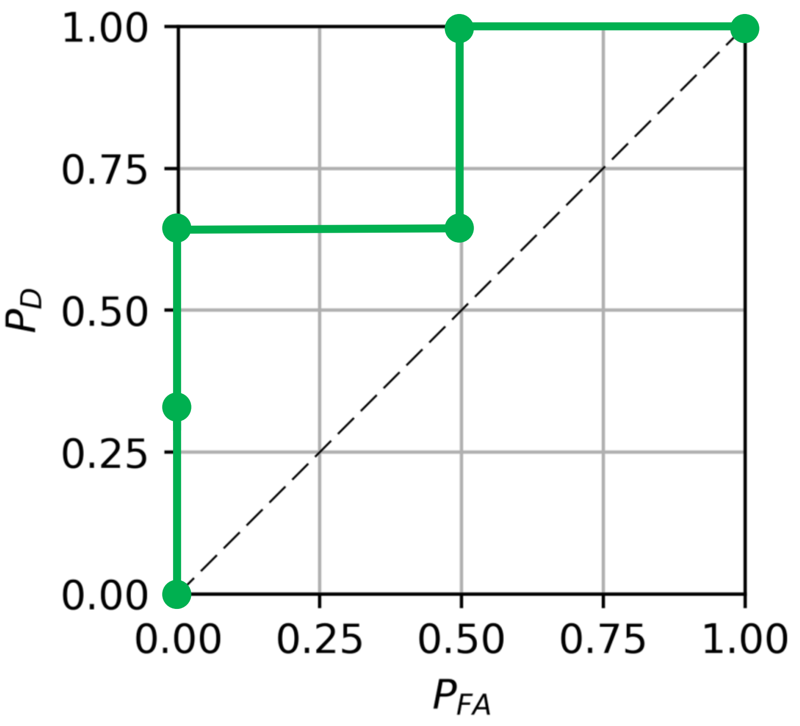
Risk (Expected loss)

State of Nature (s)

		Poor market performance $s = s_0$	Excellent market performance $s = s_1$	**Risk** (Expected Loss) $R(a_i)$
Action	Buy Apple $a = a_0$	$\lambda(a_0\|s_0)$ 1,010	$\lambda(a_0\|s_1)$ 400	$(0.7)(1010) + (0.3)(400)$ $= \mathbf{827}$
	Buy Google $a = a_1$	$\lambda(a_1\|s_0)$ 2,010	$\lambda(a_1\|s_1)$ 0	$(0.7)(2010) + (0.3)(0)$ $= \mathbf{1407}$
	Buy bonds $a = a_2$	$\lambda(a_2\|s_0)$ 0	$\lambda(a_2\|s_1)$ 2,090	$(0.7)(0) + (0.3)(2090)$ $= \mathbf{627}$
State Probability:		$P(s_0) =$ 0.7	$P(s_1) =$ 0.3	

How does this relate to supervised learning?

Where to operate along ROC?



State of Nature

Estimate	Class 0	Class 1
Class 0	$\lambda_{00} = \mathbf{0}$	$\lambda_{01} = \mathbf{100}$ **False negative**
Class 1	$\lambda_{10} = \mathbf{1}$ **False positive**	$\lambda_{11} = \mathbf{0}$

$$\lambda_{ij} = \lambda(a_i | s_j)$$

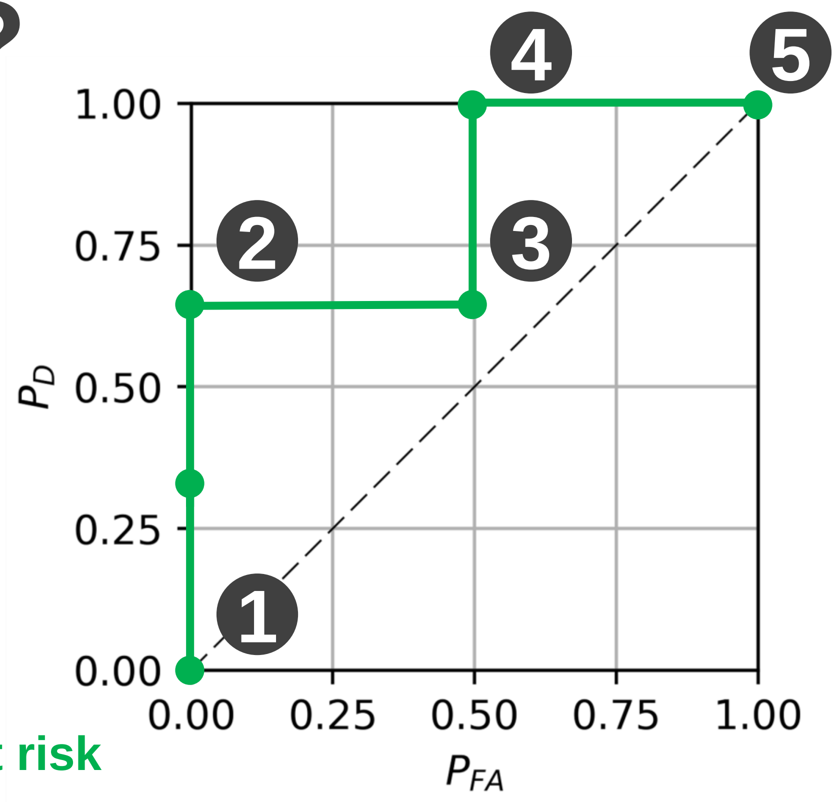
Loss from classifying as class i when state of nature is class j

- Assume our classification problem is landmine detection
- A false alarm wastes some time and resources, but a missed detection may cost a life

Where to operate along ROC?

Action: select operating point i	Probability of false alarm P_{FA}	Probability of missed detection $(1 - Pd)$	Risk $R(a_i)$
1	0	1	100
2	0	0.33	33
3	0.5	0.33	33.5
4	0.5	0	0.5
5	1	0	1

Least risk



$$R(a_i) = \sum_{j=1}^{N_S} \lambda(a_i|s_j)P(s_j)$$

$$R(a_i) = \lambda_{10}\underbrace{P_{FA}(i)}_{\text{Prob of false alarm}} + \lambda_{01}\underbrace{(1 - P_D(i))}_{\text{Prob of missed detection}}$$

State of Nature

Estimate	Class 0	Class 1
Class 0	$\lambda_{00} = 0$	$\lambda_{01} = 100$
Class 1	$\lambda_{10} = 1$	$\lambda_{11} = 0$

Let’s generalize this to any binary classifier

This is how to pick what decision threshold to use for a binary classifier

Defining risk for binary decisions

		State of Nature Class 0 $s = s_0$	Class 1 $s = s_1$
Estimate	Class 0 $a = a_0$	$\lambda(a_0 \vert s_0)$ λ_{00}	$\lambda\,(a_0 \vert s_1)$ λ_{01}
	Class 1 $a = a_1$	$\lambda\,(a_1 \vert s_0)$ λ_{10}	$\lambda\,(a_1 \vert s_1)$ λ_{11}

λ_{ij} = Loss when you classify as class i when state of nature is class j

Probability from classifier (i.e. confidence score)

$$R(a_0|\boldsymbol{x}) = \lambda_{00} P(s_0|\boldsymbol{x}) + \lambda_{01} P(s_1|\boldsymbol{x})$$

$$R(a_1|\boldsymbol{x}) = \lambda_{10} P(s_0|\boldsymbol{x}) + \lambda_{11} P(s_1|\boldsymbol{x})$$

$$P(s_i|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|s_i)P(s_i)}{P(\boldsymbol{x})}$$

1 Define the risk associated with each of the two actions

$$R(a_0|\boldsymbol{x}) = \lambda_{00}P(s_0|\boldsymbol{x}) + \lambda_{01}P(s_1|\boldsymbol{x})$$

$$R(a_1|\boldsymbol{x}) = \lambda_{10}P(s_0|\boldsymbol{x}) + \lambda_{11}P(s_1|\boldsymbol{x})$$

2 Create a decision rule based on the data

If $R(a_0|\boldsymbol{x}) < R(a_1|\boldsymbol{x})$ then a_0 (decide class 0)

Else $R(a_0|\boldsymbol{x}) > R(a_1|\boldsymbol{x})$ then a_1 (decide class 1)

We choose the rule to **minimize the risk**

3 Express this rule in terms of the output from the classifier

$$\lambda_{00}P(s_0|\boldsymbol{x}) + \lambda_{01}P(s_1|\boldsymbol{x}) > \lambda_{10}P(s_0|\boldsymbol{x}) + \lambda_{11}P(s_1|\boldsymbol{x})$$

$$\frac{P(s_1|\boldsymbol{x})}{P(s_0|\boldsymbol{x})} > \frac{\lambda_{10} - \lambda_{00}}{\lambda_{01} - \lambda_{11}} \quad \text{then} \quad a_1$$

This can be applied to any model that outputs posterior probabilities (**discriminative or generative models**)

4 Use Bayes rule to express this as a function of likelihoods

$$\frac{P(s_1|\boldsymbol{x})}{P(s_0|\boldsymbol{x})} > \frac{\lambda_{10} - \lambda_{00}}{\lambda_{01} - \lambda_{11}}$$

$$P(s_i|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|s_i)P(s_i)}{P(\boldsymbol{x})}$$

$$\frac{P(\boldsymbol{x}|s_1)P(s_1)}{P(\boldsymbol{x}|s_0)P(s_0)} > \frac{\lambda_{10} - \lambda_{00}}{\lambda_{01} - \lambda_{11}}$$

then a_1 (decide class 1)

Can easily factor in prior knowledge about the classes

5 The decision rule can be expressed as a **likelihood ratio**

$$\frac{P(\boldsymbol{x}|s_1)}{P(\boldsymbol{x}|s_0)} > \left(\frac{\lambda_{10} - \lambda_{00}}{\lambda_{01} - \lambda_{11}}\right)\frac{P(s_0)}{P(s_1)}$$

then a_1 (decide class 1)

else a_0 (decide class 0)

This can be readily applied to **generative models**

Special case: Minimizing the misclassification rate

$$\frac{P(s_1|\boldsymbol{x})}{P(s_0|\boldsymbol{x})} > \frac{\lambda_{10} - \lambda_{00}}{\lambda_{01} - \lambda_{11}} \quad \text{then} \quad a_1 \quad \text{(decide class 1)}$$

Assume that the loss is only for error, and it's the same for both types of error:

$$\lambda_{10} = \lambda_{01} \quad \text{and} \quad \lambda_{00} = \lambda_{11} = 0$$

Then the decision rule simplifies to the following:

$$\frac{P(s_1|\boldsymbol{x})}{P(s_0|\boldsymbol{x})} > 1 \quad \text{then} \quad a_1 \ \text{(decide class 1)}$$

$$\text{else} \quad a_0 \ \text{(decide class 0)}$$

Pick whichever class is more likely given the data

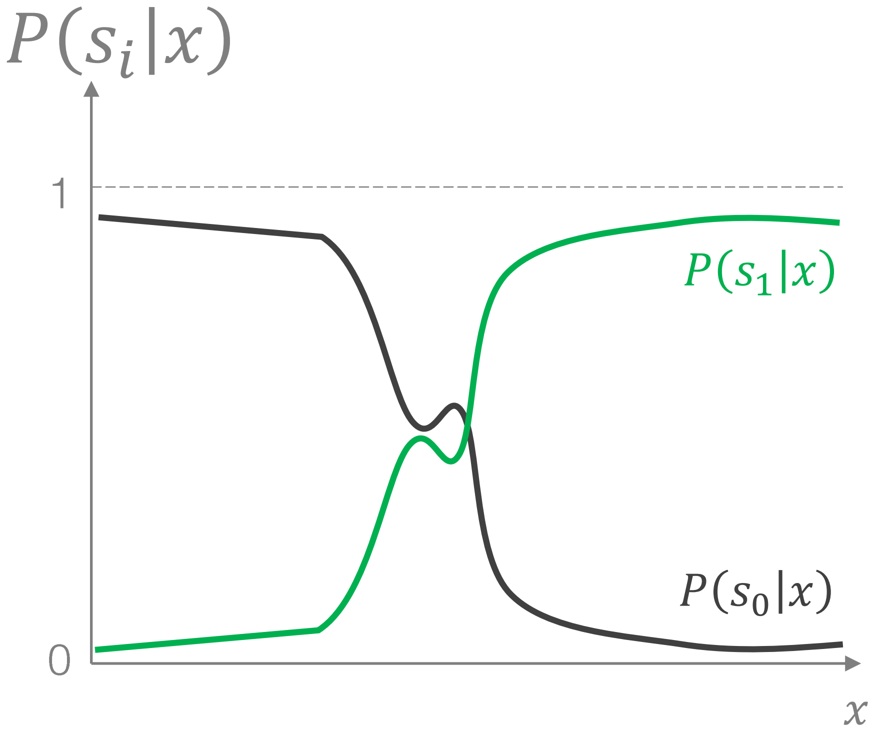
Recall Bayes' Rule

Note: The **evidence** ensures the posterior integrates to 1

$$\underbrace{P(s_i|\boldsymbol{x})}_{\text{Posterior}} = \frac{\overbrace{P(\boldsymbol{x}|s_i)}^{\text{Likelihood}}\overbrace{P(s_i)}^{\text{Prior}}}{\underbrace{P(\boldsymbol{x})}_{\text{Evidence}}}$$

Posterior

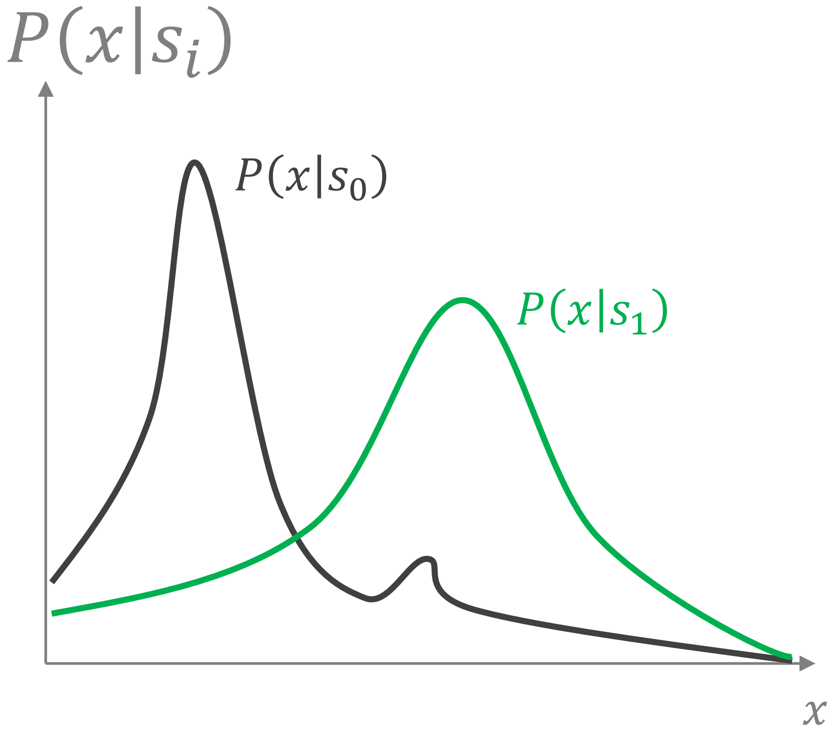
Answers the question: after seeing the data – which class is it most likely to belong to? Summing this across classes equals 1.



Discriminative models estimate this

Likelihood

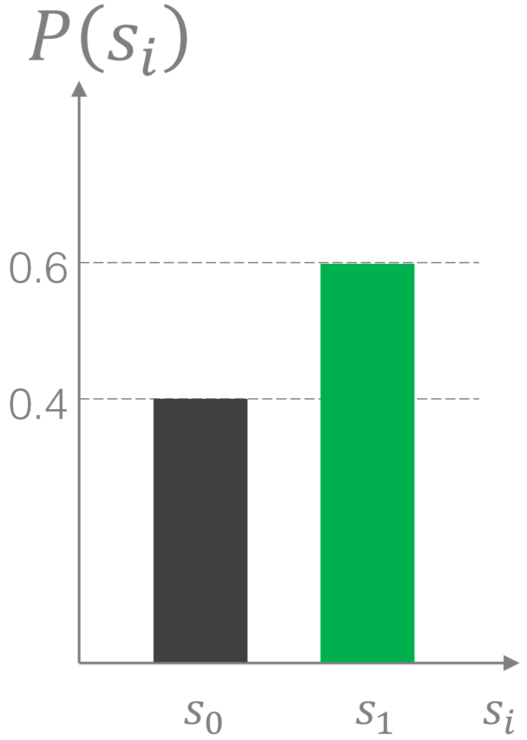
Answers the question: if I knew which class a sample belongs to, how are the data distributed?



Generative models estimate this

Prior

Answers the question: what do I anticipate is the balance between my classes?



Generative and discriminative models

Unobservable state of the world

Data Generating Process

$p(X, Y)$

Target Function for predicting y from x

$f(x) \rightarrow y$

Types of models. We can either model the full data generating process **OR** the target function, the mapping of x to y

If we model this process, it's a **generative model**

- Models $P(x|y)$
- Can be used to generate synthetic data and impute missing values
- Examples: naïve Bayes, linear discriminant analysis, hidden Markov models

If we model this function, it's a **discriminative model**

- Model $P(y|x)$ OR directly map x to y without probabilities
- Often better performance for large sample sizes
- Examples: logistic regression, support vector machines, neural networks, k nearest neighbors

Takeaways

To make a decision:

1. Define a measure of risk or reward
2. Select the action that optimizes that metric

Decision theory informs how to operate supervised learning algorithms in practice

Decision theory incorporates relative importance of different error types

Generative models estimate $P(x|y)$, while discriminative models estimate $P(y|x)$