Multiclass Classification

General Classification

Given $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ training examples where $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \{1, \ldots, C\}$

Goal: Given new query point $\mathbf{x}$, predict its label y

Some problems are inherently binary:

- SPAM vs HAM

Some problems are inherently multiclass:

- Is there a car/truck/motorcycle/pedestrian in this image?
- Should the self-driving car speed up, slow down, hold speed?

Inherently Multiclass Classification

Some models are inherently multiclass:

Naïve Bayes:

For each class k, estimate $p(y = c \mid \mathbf{x})$ for each $k = 1, \ldots, C$

Assign $\mathbf{x}$ to the class with highest associated probability:

$$\hat{y} = \text{argmax}_c \; p(y = c \mid \mathbf{x})$$

Inherently Multiclass Classification

Some models are inherently multiclass:

K-Nearest Neighbors:

- Find $\mathcal{N}_K(\mathbf{x})$, the K training examples in training data that are "nearest" to $\mathbf{x}$
- Assign $\hat{y}$ the majority label in the neighborhood $\mathcal{N}_K(\mathbf{x})$

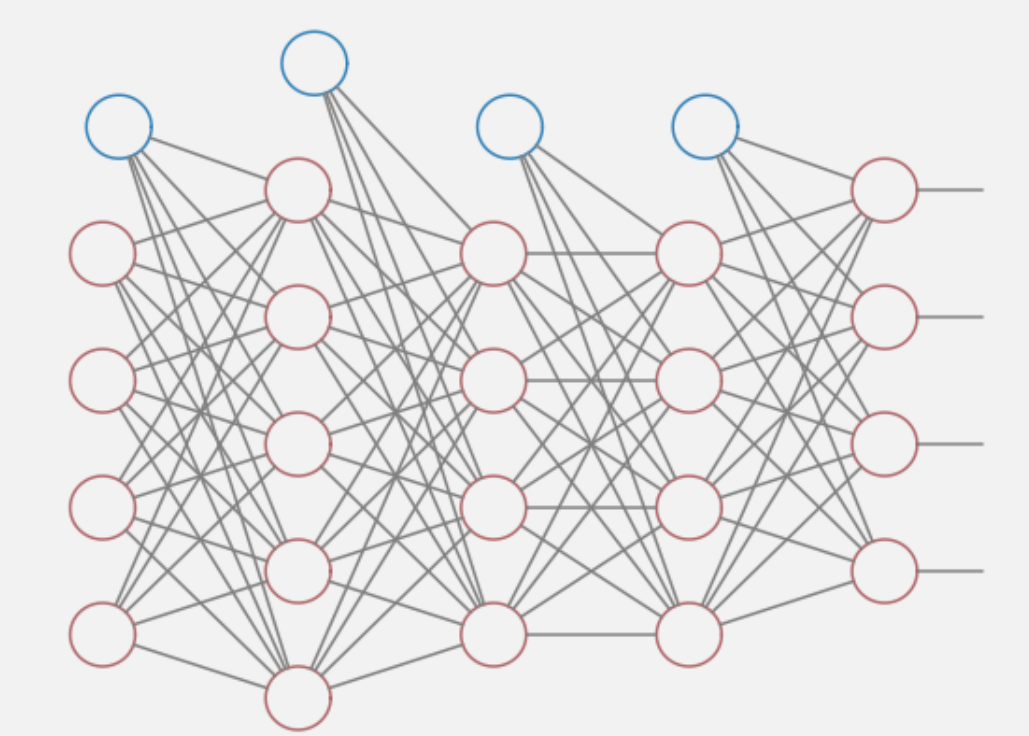
Also had decent probabilistic interpretation

$$p(y = c \mid \mathbf{x}) = \frac{1}{K} \sum_{i \in \mathcal{N}_K} I(y_i = c) \qquad \hat{y} = \operatorname{argmax}_c \, p(y = c \mid \mathbf{x})$$

Inherently Multiclass Classification

Some models are inherently multiclass:

Neural Networks:



Inherently Multiclass Classification

Some models are inherently multiclass:

Logistic Regression:

For binary classification we learned weights such that

$$\log \frac{p(y=1 \mid \mathbf{x})}{p(y=0 \mid \mathbf{x})} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p = \boldsymbol{\beta}^T \mathbf{x}$$

$$\log \frac{p(y=1 \mid \mathbf{x})}{1 - p(y=1 \mid \mathbf{x})} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p = \boldsymbol{\beta}^T \mathbf{x}$$

With

$$p(y=1 \mid \mathbf{x}) = \text{sigm}(\boldsymbol{\beta}^T \mathbf{x}) \qquad p(y=0 \mid \mathbf{x}) = 1 - \text{sigm}(\boldsymbol{\beta}^T \mathbf{x})$$

Inherently Multiclass Classification

For Logistic Regression with more than two classes:

PROB X BELONGS TO c=1

PROB X BELONGS TO c=C

$$\log \frac{p(y=1 \mid \mathbf{x})}{p(y=C \mid \mathbf{x})} = \beta_{10} + \beta_{11}x_1 + \beta_{12}x_2 + \cdots + \beta_{1p}x_p = \boldsymbol{\beta}_1^T \mathbf{x}$$

$$\log \frac{p(y=2 \mid \mathbf{x})}{p(y=C \mid \mathbf{x})} = \beta_{20} + \beta_{21}x_1 + \beta_{22}x_2 + \cdots + \beta_{2p}x_p = \boldsymbol{\beta}_2^T \mathbf{x}$$

$$\vdots$$

$$\log \frac{p(y=C-1 \mid \mathbf{x})}{p(y=C \mid \mathbf{x})} = \beta_{C-1,0} + \beta_{C-1,1}x_1 + \cdots + \beta_{C-1,p}x_p = \boldsymbol{\beta}_{C-1}^T \mathbf{x}$$

Inherently Multiclass Classification

For Logistic Regression with more than two classes:

$$\log \frac{p(y=1 \mid \mathbf{x})}{p(y=C \mid \mathbf{x})} = \beta_{10} + \beta_{11}x_1 + \beta_{12}x_2 + \cdots + \beta_{1p}x_p = \boldsymbol{\beta}_1^T \mathbf{x}$$

$$\log \frac{p(y=2 \mid \mathbf{x})}{p(y=C \mid \mathbf{x})} = \beta_{20} + \beta_{21}x_1 + \beta_{22}x_2 + \cdots + \beta_{2p}x_p = \boldsymbol{\beta}_2^T \mathbf{x}$$

$$\vdots$$

$$\log \frac{p(y=C-1 \mid \mathbf{x})}{p(y=C \mid \mathbf{x})} = \beta_{C-1,0} + \beta_{C-1,1}x_1 + \cdots + \beta_{C-1,p}x_p = \boldsymbol{\beta}_{C-1}^T \mathbf{x}$$

With

$$p(y=c \mid \mathbf{x}) = \frac{\exp(\boldsymbol{\beta}_c^T \mathbf{x})}{1 + \sum_{c=1}^{C-1} \exp(\boldsymbol{\beta}_c^T \mathbf{x})} \qquad p(y=C \mid \mathbf{x}) = \frac{1}{1 + \sum_{c=1}^{C-1} \exp(\boldsymbol{\beta}_c^T \mathbf{x})}$$

Multiclass Classification

Some methods can are commonly implemented as binary classification

- Logistic Regression

Other methods that we've seen (or will see) are inherently binary:

- The Perceptron
- Support Vector Machines (ehh, kinda)
- Ensembled Decision Stumps

It would be nice if we could take our favorite binary classifiers and leverage them to do multiclass classification

Binary to Multiclass Classification

Example Data:

<name=Cindy, age=5, sex=F> , 
<name=Marcia, age=15, sex=F> , 
<name=Bobby, age=6, sex=M> , 
<name=Jan, age=12, sex=F> , 
<name=Peter, age=13, sex=M> , 

We'll look at two strategies:

- **One-vs-All** or **OVA**
- **All-Pairs** or **AP**

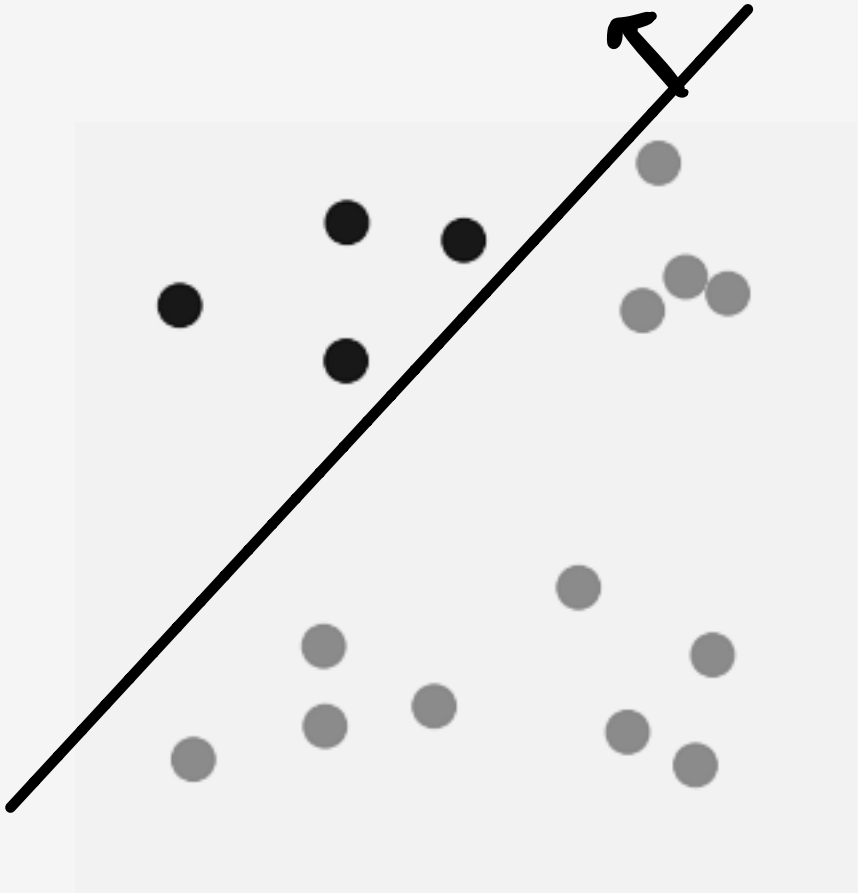
One-vs-All

Build C binary classifiers of the form *Class c* vs *Class Not-c*

One-vs-All

Build C binary classifiers of the form *Class c* vs *Class Not-c*

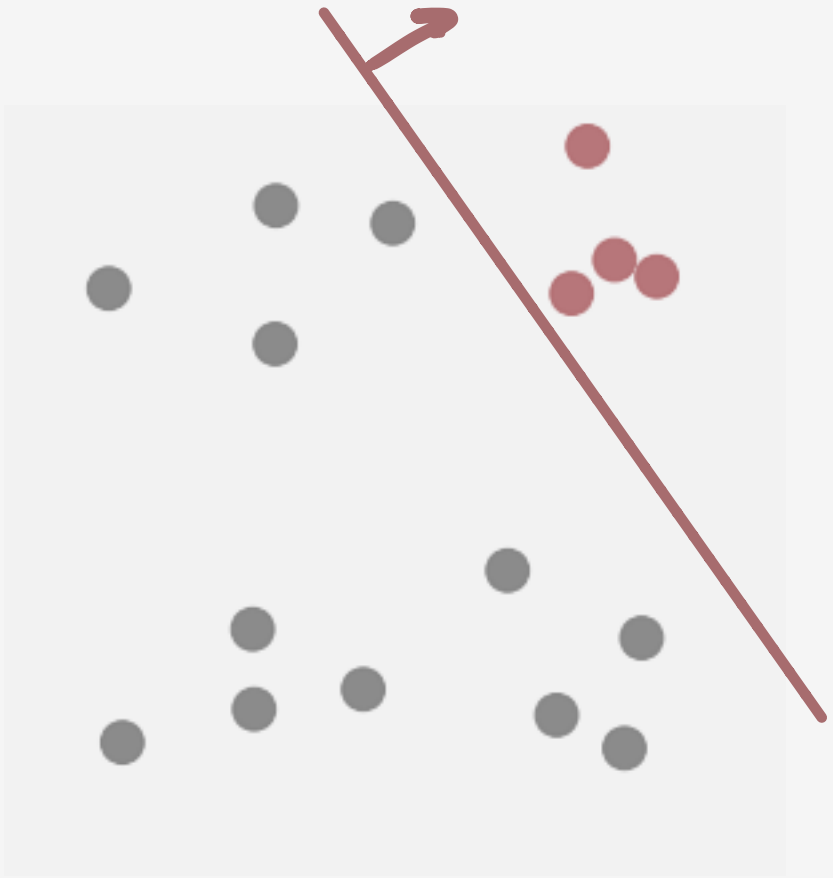
- Black vs Not-Black



One-vs-All

Build C binary classifiers of the form *Class c* vs *Class Not-c*

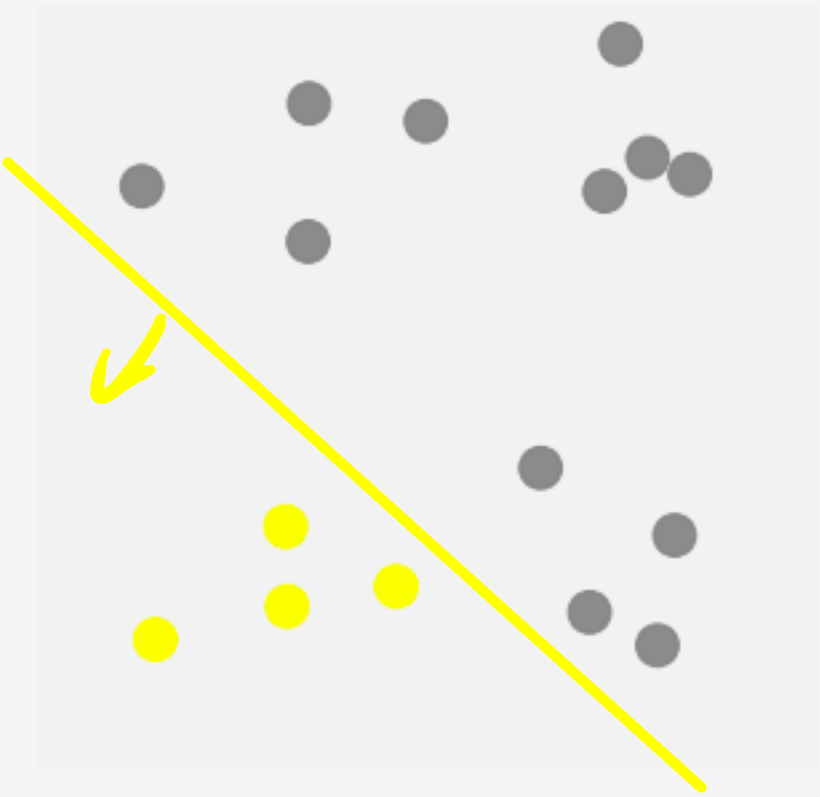
- Red vs Not-Red



One-vs-All

Build C binary classifiers of the form *Class c* vs *Class Not-c*

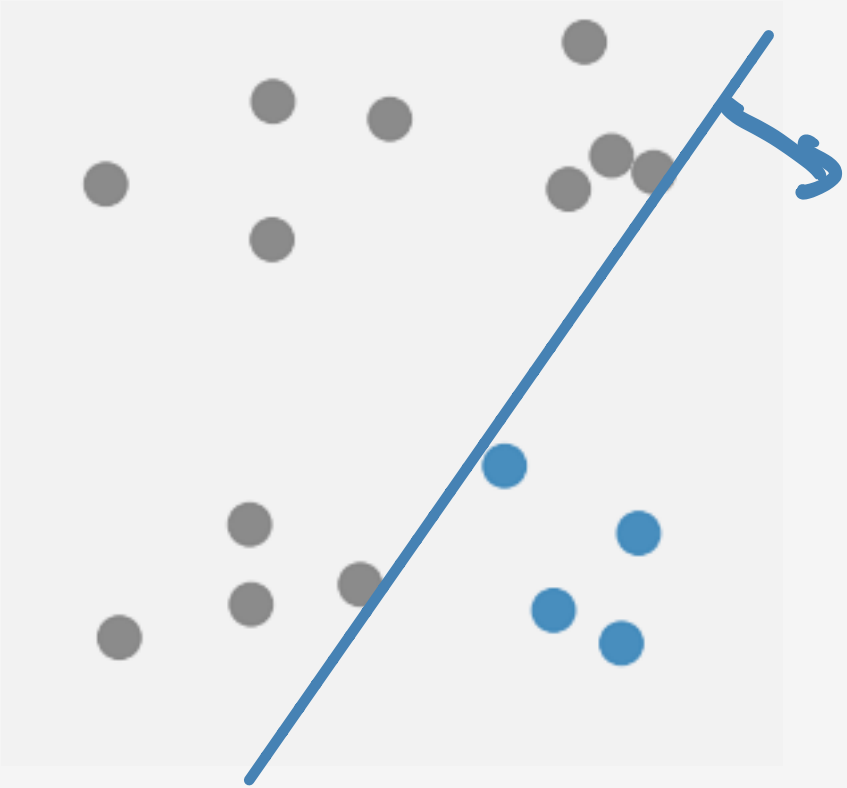
- Yellow vs Not-Yellow



One-vs-All

Build C binary classifiers of the form *Class c* vs *Class Not-c*

- Blue vs Not-Blue



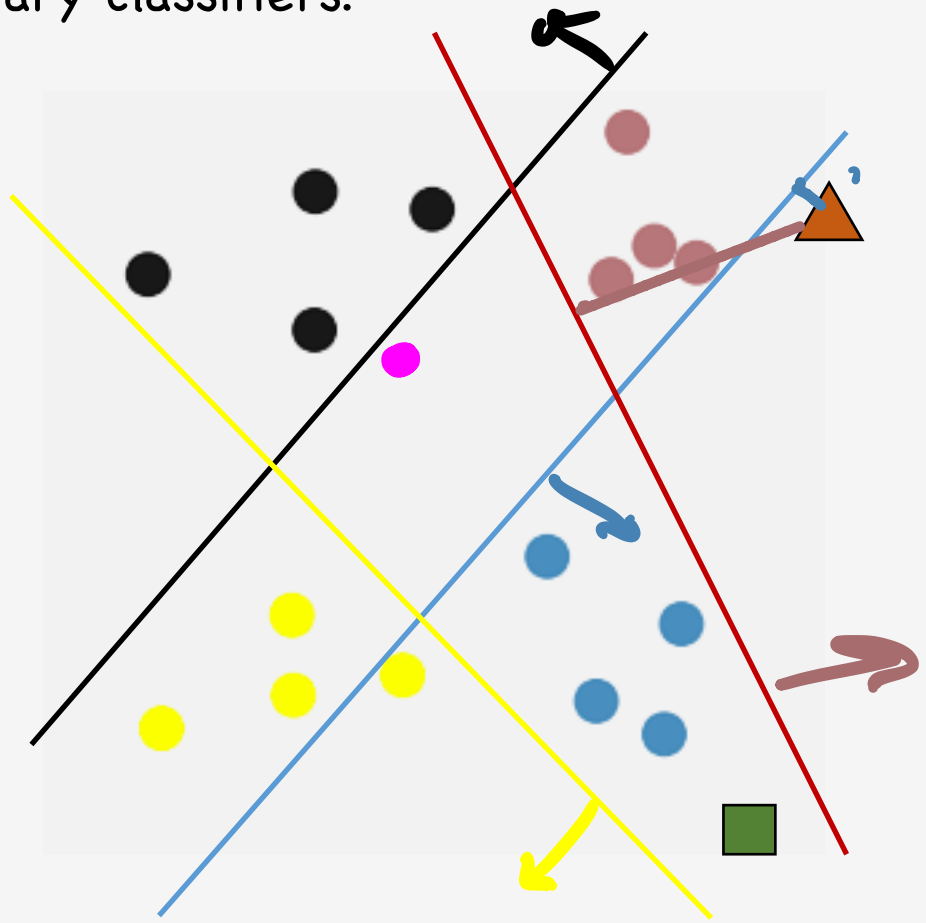
One-vs-All

Build C binary classifiers of the form *Class c* vs *Class Not-c*

For a new query point, evaluate with each of the C binary classifiers.

Predict class with highest confidence (if available)

- Predict ■ ⇒ $\tilde{y}$ = BLUE
- Predict ▲ ⇒ $\hat{y}$ = RED



One-vs-All: Tabular Form

Build C binary classifiers of the form *Class c* vs *Class not-c*

Helpful to think of as a table of labels and classifiers

BLACK vs NOT BLACK

	■	■	■	■
$\mathbf{x}_1$ ■	$-$	$+$	$-$	$-$
$\mathbf{x}_2$ ■	$-$	$-$	$+$	$-$
$\mathbf{x}_3$ ■	$-$	$-$	$-$	$+$
$\mathbf{x}_4$ ■	$-$	$+$	$-$	$-$
$\mathbf{x}_5$ ■	$+$	$-$	$-$	$-$
	h_1	h_2	h_3	h_4

Evaluating each of C classifiers gives a vector encoding: $\mathbf{x} \rightarrow [0,\ 1,\ 0,\ 0,$ 

One-vs-All

Question: Can you see any potential pitfalls for the OVA method?

One-vs-All

Question: Can you see any potential pitfalls for the OVA method?

A Big One: If you start with a balanced training data, you immediately create imbalanced data

Remedies:

- **Undersample** the not-C class
- **Oversample** is the is-C class
- **Undersampled Ensembling**

All-Pairs

Build $\binom{C}{2}$ binary classifiers of the form Class c_i vs Class c_j

All-Pairs: Tabular Form

Build $\binom{C}{2}$ binary classifiers of the form Class c_i vs Class c_j

	■ v ■	■ v ■	■ v ■	■ v ■	■ v ■	■ v ■
$\mathbf{x}_1$ ■	$-$			$-$		$-$
$\mathbf{x}_2$ ■		$-$	$+$			$+$
$\mathbf{x}_3$ ■			$-$	$+$	$-$	
$\mathbf{x}_4$ ■	$-$			$-$		$-$
$\mathbf{x}_5$ ■	$+$	$+$			$+$	
	h_{12}	h_{13}	h_{34}	h_{42}	h_{14}	h_{32}

- Build one binary classifier for each pair of labels
- Define h_{ij} when $i \to +$ and $j \to -$. **Note:** $h_{ij} = -h_{ji}$
- Classify with $H(\mathbf{x}) = \text{argmax}_i \left(\sum_j h_{ij}(\mathbf{x}) \right)$

One-vs-All vs All-Pairs

$\alpha = 1.5$ n = # TRAINING EX's

Time Complexity:

Suppose training time for a single classifier is $\mathcal{O}(n^\alpha)$ and testing time for single $\mathbf{x}$ is $\mathcal{O}(T)$

	Train	Test
OVA	$\mathcal{O}(Cn^\alpha)$	$\mathcal{O}(CT)$
APs	$\mathcal{O}(C^2(n/C)^\alpha)$	$\mathcal{O}(C^2T)$

One-vs-All vs All-Pairs

Time Complexity:

Suppose training time for a single classifier is $\mathcal{O}(n^\alpha)$ and testing time for single $\mathbf{x}$ is $\mathcal{O}(T)$

	Train	Test
OVA	$\mathcal{O}(Cn^\alpha)$	$\mathcal{O}(CT)$
APs	$\mathcal{O}(C^2(n/C)^\alpha)$	$\mathcal{O}(C^2T)$

- All-Pairs wins at Training Time
- One-vs-All wins at Test Time
- All-Pairs generally more accurate (caveat: I read this on the internet)

Error-Correcting Output Codes

Another popular scheme comes from the theory of error-correcting codes

Consider the following binary coding matrix, with columns corresponding to N binary classifiers

Class	h_0	h_1	h_2	h_3	h_4
0	1	0	1	0	1
1	0	0	1	1	1
2	1	1	0	0	0
3	1	1	1	1	0

$x \rightarrow [1\ 1\ 1\ 0\ 0]$

The binary encoding of each class is called a codeword

For query point, predict with each of N classifiers to get predicted codeword

Predict class where codeword is closest to associated row of the coding matrix

Error-Correcting Output Codes

Class	h_0	h_1	h_2	h_3	h_4
0	1	0	1	0	1
1	0	0	1	1	1
2	1	1	0	0	0
3	1	1	1	1	0

Question: What do we mean by "closest"?

Error-Correcting Output Codes

Class	h_0	h_1	h_2	h_3	h_4
0	1	0	1	0	1
1	0	0	1	1	1
2	1	1	0	0	0
3	1	1	1	1	0

Question: What do we mean by "closest"?

Answer: Measure distance in terms of **Hamming Distance**

$$d(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{n} |x_i - z_i|$$

Error-Correcting Output Codes

Class	h_0	h_1	h_2	h_3	h_4
0	1	0	1	0	1
1	0	0	1	1	1
2	1	1	0	0	0
3	1	1	1	1	0

Question: Why is such a scheme a good idea?

Error-Correcting Output Codes

Class	h_0	h_1	h_2	h_3	h_4
0	1	0	1	0	1
1	0	0	1	1	1
2	1	1	0	0	0
3	1	1	1	1	0

Question: Why is such a scheme a good idea?

Answer: You can correct errors

Error-Correcting Output Codes

Example: Suppose you want to predict whether name is of Korean, German, or Argentine descent

Data:

Korean	German	Argentine
Park	Mann	Puig
Kim	Berndt	Borges
Kwon	Grass	Cortazar

Codebook:

Class	h_0	h_1	h_2	h_3
Korean	1	0	0	1
German	0	0	1	0
Argentine	1	1	1	0

Error-Correcting Output Codes

Example: Suppose you want to predict whether name is of Korean, German, or Argentine descent

Data:

Korean	German	Argentine
Park	Mann	Puig
Kim	Berndt	Borges
Kwon	Grass	Cortazar

Codebook:

Class	h_0	h_1	h_2	h_3
Korean	1	0	0	1
German	0	0	1	0
Argentine	1	1	1	0

	h_0	h_1	h_2	h_3
Park	1	0	0	1
Kim	1	0	0	1
Kwon	1	0	0	1
Mann	0	0	1	0
Berndt	0	0	1	0
Gross	0	0	1	0
Puig	1	1	1	0
Borges	1	1	1	0
Cortazar	1	1	1	0

Error-Correcting Output Codes

Example: Suppose you want to predict whether name is of Korean, German, or Argentine descent

Data:

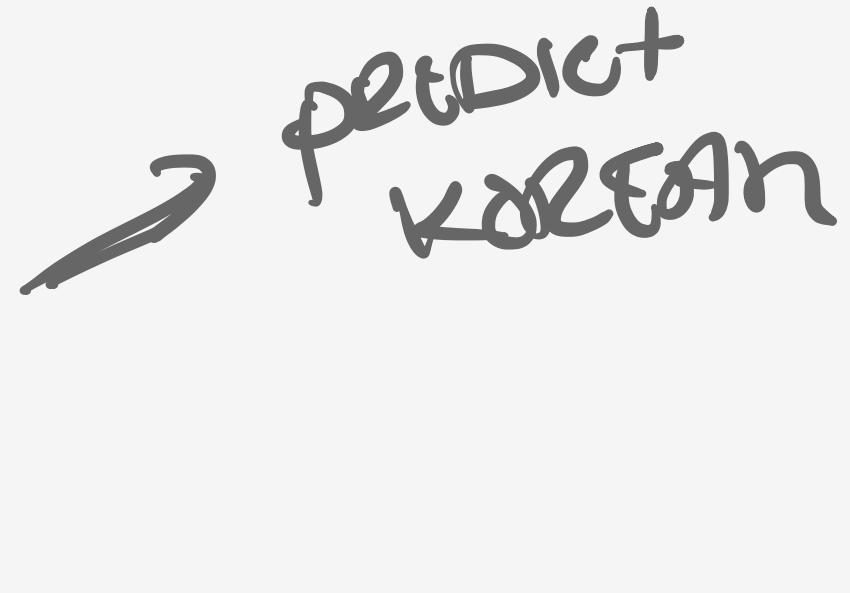
Korean	German	Argentine
Park	Mann	Puig
Kim	Berndt	Borges
Kwon	Grass	Cortazar

Suppose we predict $\hat{y} = (0, 0, 0, 1)$

Which class do we predict?

Codebook:

Class	h_0	h_1	h_2	h_3
Korean	1	0	0	1
German	0	0	1	0
Argentine	1	1	1	0



Error-Correcting Output Codes

Question: What is the codebook for the One-vs-All Scheme with three classes?

Follow-Up Question: How many errors can One-vs-All correct?

Error-Correcting Output Codes

Question: How many useful codewords can you have for a three-class classifier?

There are 8 possible hypotheses:

Class	h_0	h_1	h_2	h_3	h_4	h_5	h_6	h_7
0	1	0	0	1	0	1	0	1
1	0	1	0	1	1	0	0	1
2	0	0	1	0	1	1	0	1

Error-Correcting Output Codes

Question: How many useful hypotheses can you have for a three-class classifier?

There are 8 possible hypotheses:

Class	h_0	h_1	h_2	h_3	h_4	h_5	h_6	h_7
0	1	0	0	1	0	1	0	1
1	0	1	0	1	1	0	0	1
2	0	0	1	0	1	1	0	1

Only three of the hypotheses are useful, corresponding to the OVA scheme

Here we have a minimal Hamming Distance of 2, which means we can't correct any errors ☹

Error-Correcting Output Codes

Question: What is the general formula for the number of useful hypotheses with C classes?

- There are _____ hypotheses that don't distinguish between classes
- There are _____ hypotheses that are complements of each other

Thus we conclude that for a C-class problem there are _____ useful hypotheses

Error-Correcting Output Codes

Question: What is the general formula for the number of useful hypotheses with C classes?

- There are 2 hypotheses that don't distinguish between classes
- There are 2^{C-1} hypotheses that are complements of each other

Thus we conclude that for a C-class problem there are $2^{C-1}-1$ useful hypotheses

Error-Correcting Output Codes

It turns out that the more classes you have, the more useful an ECOC scheme becomes

Example: With C = 10 classes you can use a codebook with 15 hypotheses that gives a minimal Hamming Distance of d=7

This means we can correct up to _____ errors

In general, with a minimal Hamming Distance of d we can correct up to _____ errors

Error-Correcting Output Codes

It turns out that the more classes you have, the more useful an ECOC scheme becomes

Example: With C = 10 classes you can use a codebook with 15 hypotheses that gives a minimal Hamming Distance of d=7

This means we can correct up to three errors

In general, with a minimal Hamming Distance of d we can correct up to $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors

Error-Correcting Output Codes

It turns out that the more classes you have, the more useful an ECOC scheme becomes

Example: With C = 10 classes you can use a codebook with 15 hypotheses that gives a minimal Hamming Distance of d=7

Class	Code Word														
	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
0	1	1	0	0	0	0	1	0	1	0	0	1	1	0	1
1	0	0	1	1	1	1	0	1	0	1	1	0	0	1	0
2	1	0	0	1	0	0	0	1	1	1	1	0	1	0	1
3	0	0	1	1	0	1	1	1	0	0	0	0	1	0	1
4	1	1	1	0	1	0	1	1	0	0	1	0	0	0	1
5	0	1	0	0	1	1	0	1	1	1	0	0	0	0	1
6	1	0	1	1	1	0	0	0	0	1	0	1	0	0	1
7	0	0	0	1	1	1	1	0	1	0	1	1	0	0	1
8	1	1	0	1	0	1	1	0	0	1	0	0	0	1	1
9	0	1	1	1	0	0	0	0	1	0	1	0	0	1	1