# Multiclass Classification

Alan Ritter

(many slides from Greg Durrett, Vivek Srikumar, Stanford CS231n)

# This Lecture

- Multiclass fundamentals
- Feature extraction
- Multiclass logistic regression
- Multiclass SVM
- Optimization

# Multiclass Fundamentals

# Text Classification

## A Cancer Conundrum: Too Many Drug Trials, Too Few Patients

Breakthroughs in immunotherapy and a rush to develop profitable new treatments have brought a crush of clinical trials scrambling for patients.

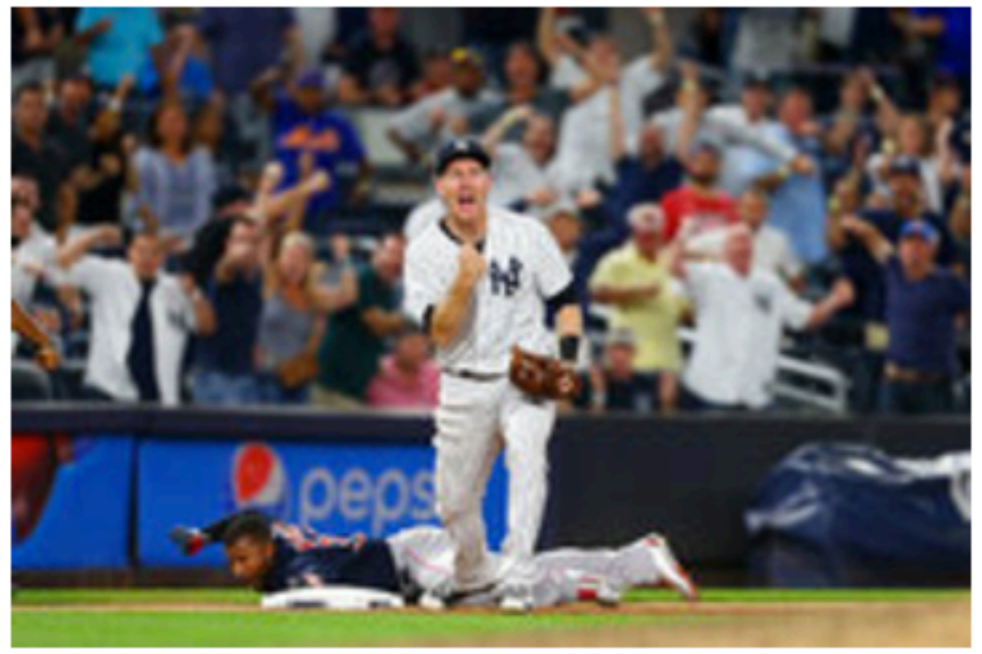
By GINA KOLATA

→ Health

## Yankees and Mets Are on Opposite Tracks This Subway Series

As they meet for a four-game series, the Yankees are playing for a postseason spot, and the most the Mets can hope for is to play spoiler.

By FILIP BONDY



→ Sports

~20 classes

# Image Classification

→ Dog

→ Car

- Thousands of classes (ImageNet)

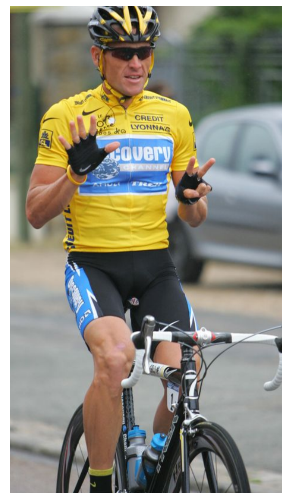
# Entity Linking

# Entity Linking

Although he originally won the event, the United States Anti-Doping Agency announced in August 2012 that they had disqualified **Armstrong** from his seven consecutive Tour de France wins from 1999–2005.

# Entity Linking

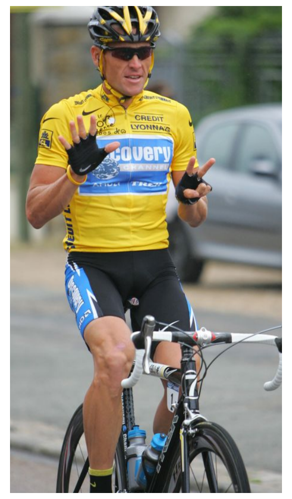
Although he originally won the event, the United States Anti-Doping Agency announced in August 2012 that they had disqualified **Armstrong** from his seven consecutive Tour de France wins from 1999–2005.



Lance Edward Armstrong is an American former professional road cyclist

# Entity Linking

Although he originally won the event, the United States Anti-Doping Agency announced in August 2012 that they had disqualified **Armstrong** from his seven consecutive Tour de France wins from 1999–2005.

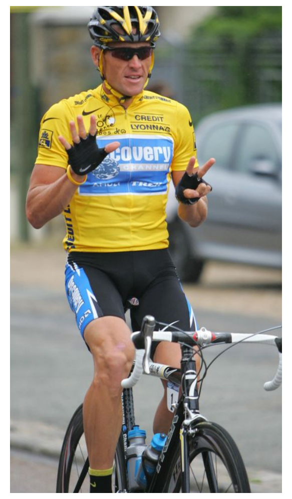


Lance Edward Armstrong is an American former professional road cyclist

Armstrong County is a county in Pennsylvania…

# Entity Linking

Although he originally won the event, the United States Anti-Doping Agency announced in August 2012 that they had disqualified **Armstrong** from his seven consecutive Tour de France wins from 1999–2005.



Lance Edward Armstrong is an American former professional road cyclist

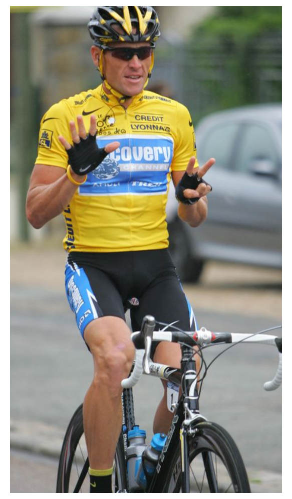
Armstrong County is a county in Pennsylvania…

?

?

# Entity Linking

Although he originally won the event, the United States Anti-Doping Agency announced in August 2012 that they had disqualified **Armstrong** from his seven consecutive Tour de France wins from 1999–2005.



Lance Edward Armstrong is an American former professional road cyclist

Armstrong County is a county in Pennsylvania…

? ?

- 4,500,000 classes (all articles in Wikipedia)

# Reading Comprehension

One day, James thought he would go into town and see what kind of trouble he could get into. He went to the grocery store and pulled all the pudding off the shelves and ate two jars. Then he walked to the fast food restaurant and ordered 15 bags of fries. He didn't pay, and instead headed home.

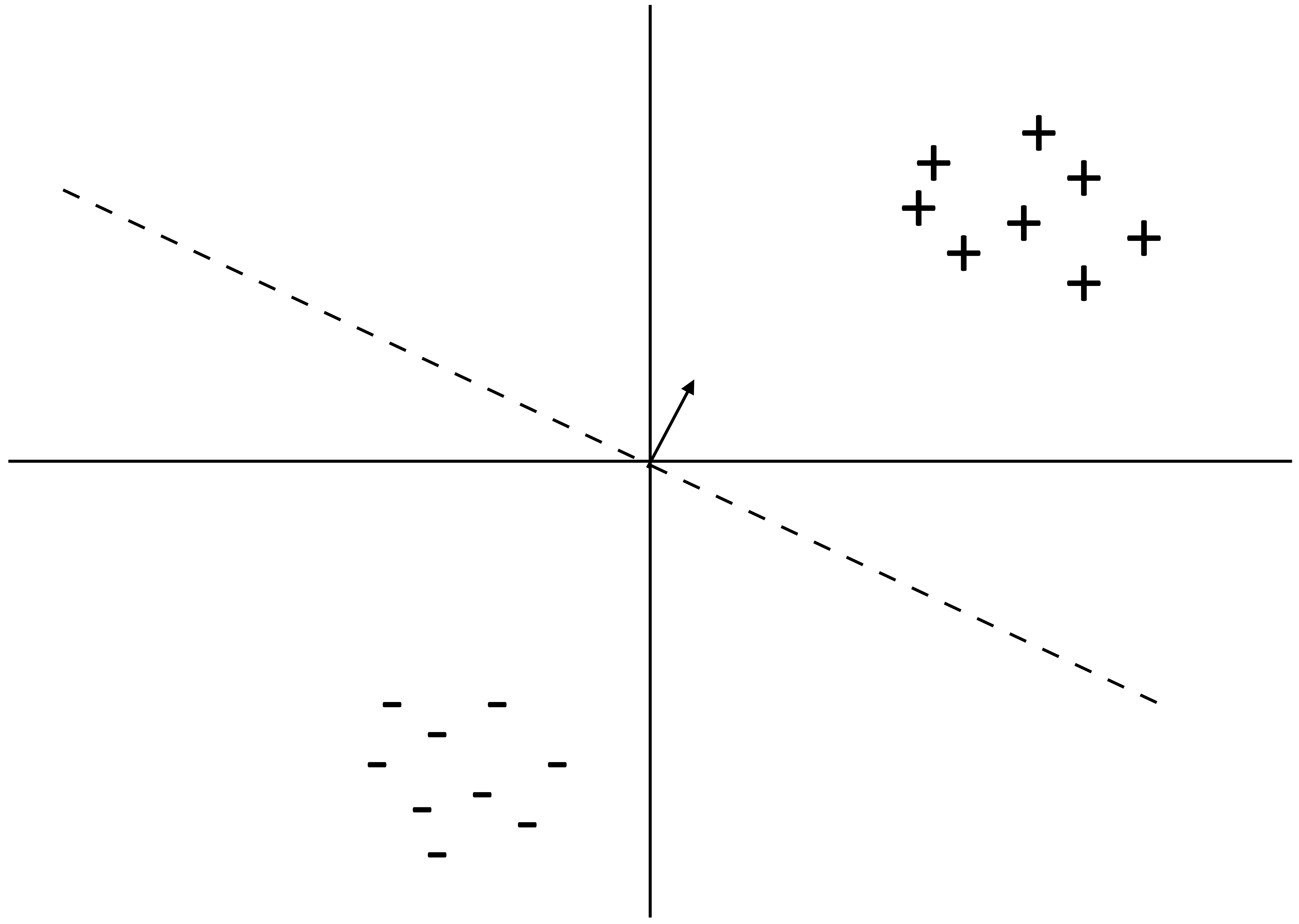
3) Where did James go after he went to the grocery store?
A) his deck
B) his freezer
C) a fast food restaurant
D) his room

- Multiple choice questions, 4 classes (but classes change per example)

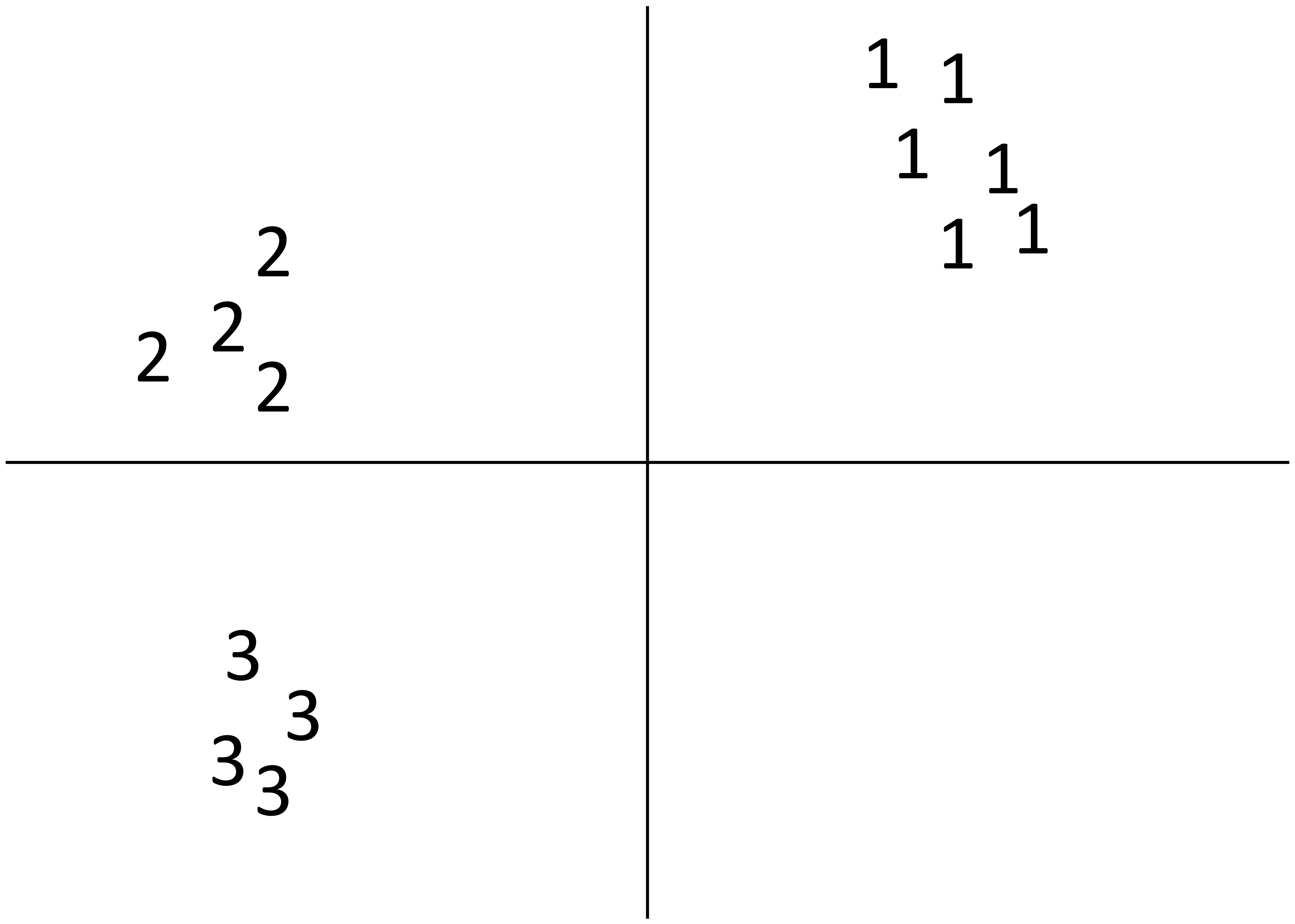
Richardson (2013)

# Binary Classification

- Binary classification: one weight vector defines positive and negative classes

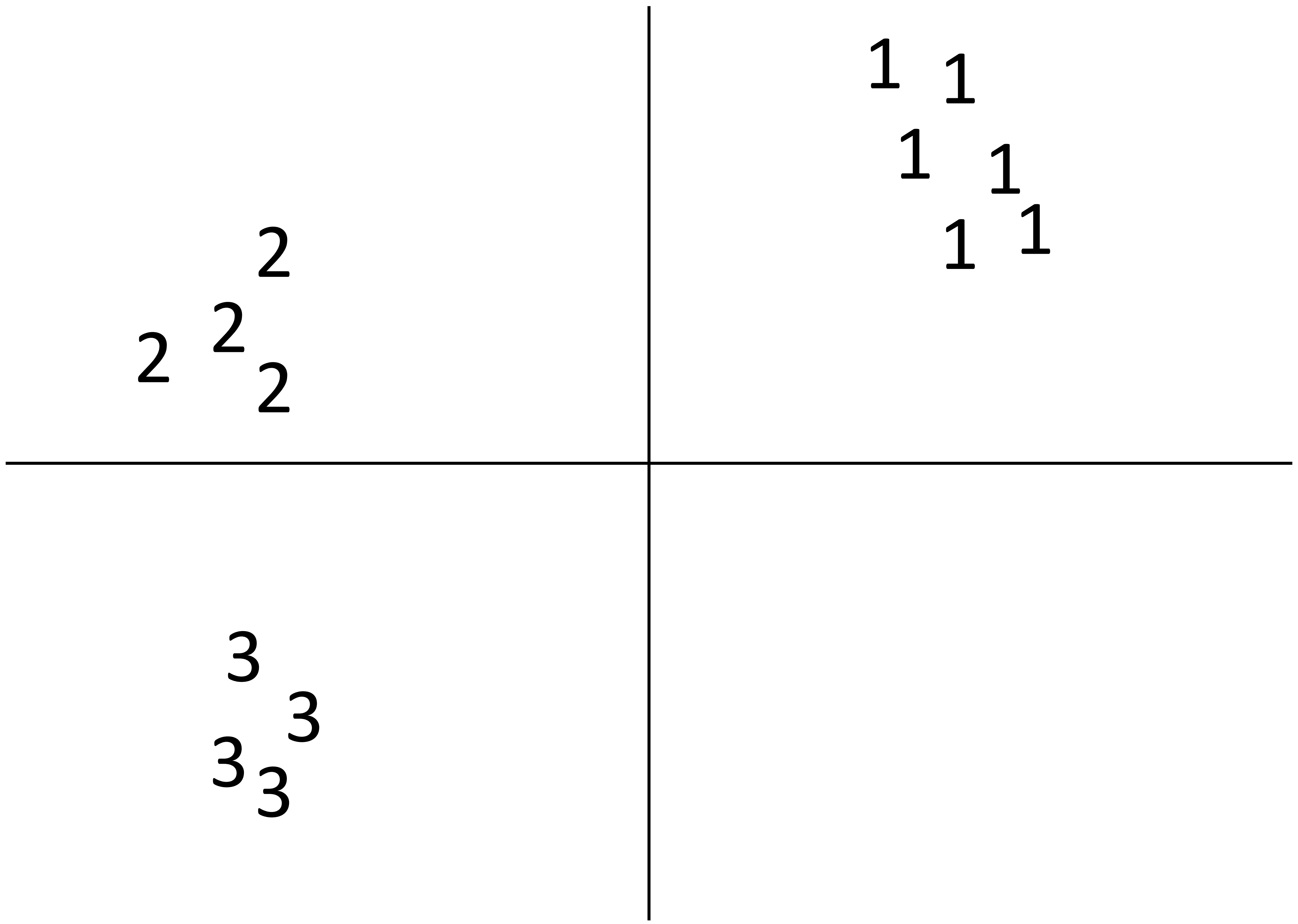


# Multiclass Classification



# Multiclass Classification

- Can we just use binary classifiers here?

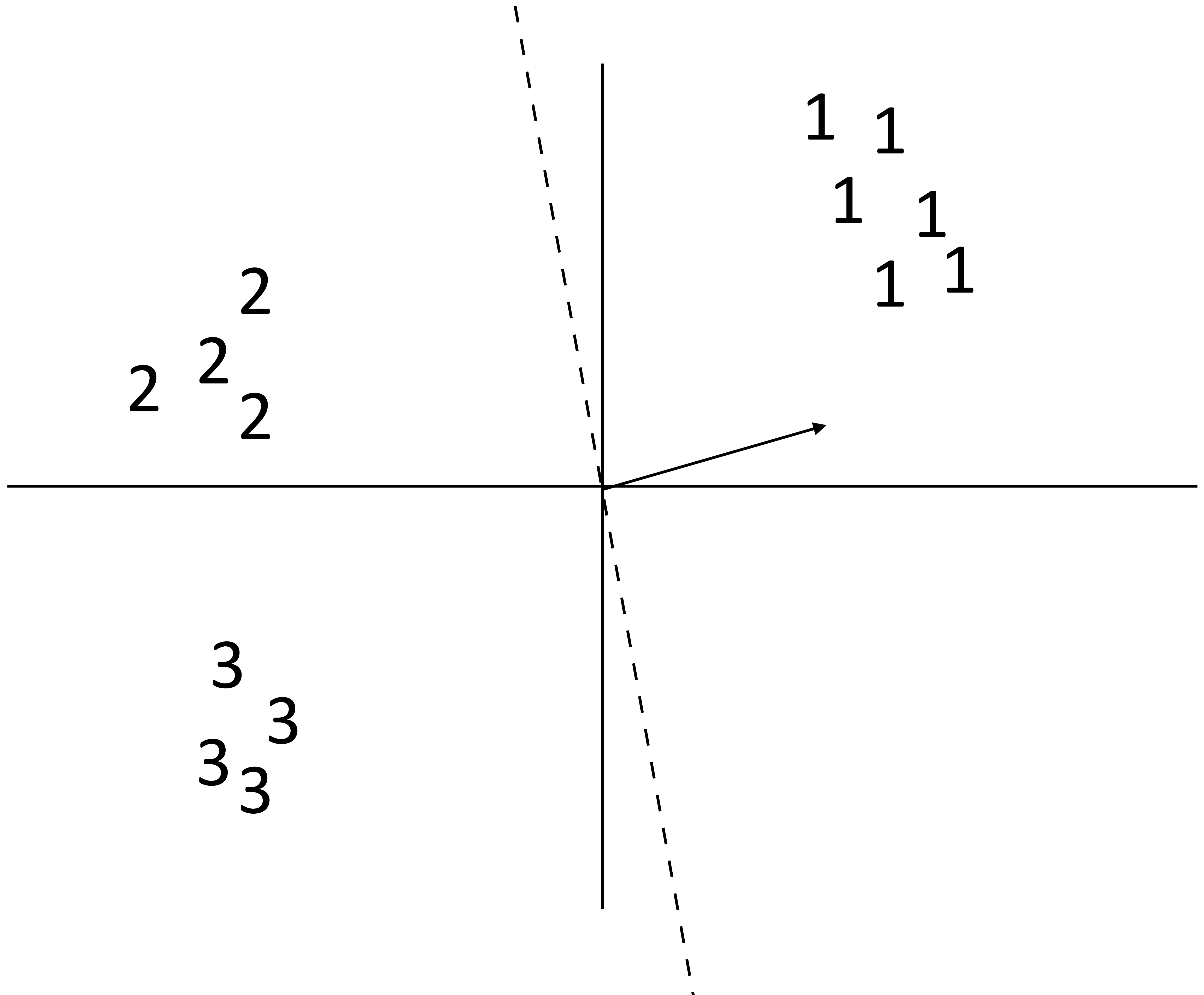


# Multiclass Classification

- One-vs-all: train $k$ classifiers, one to distinguish each class from all the rest

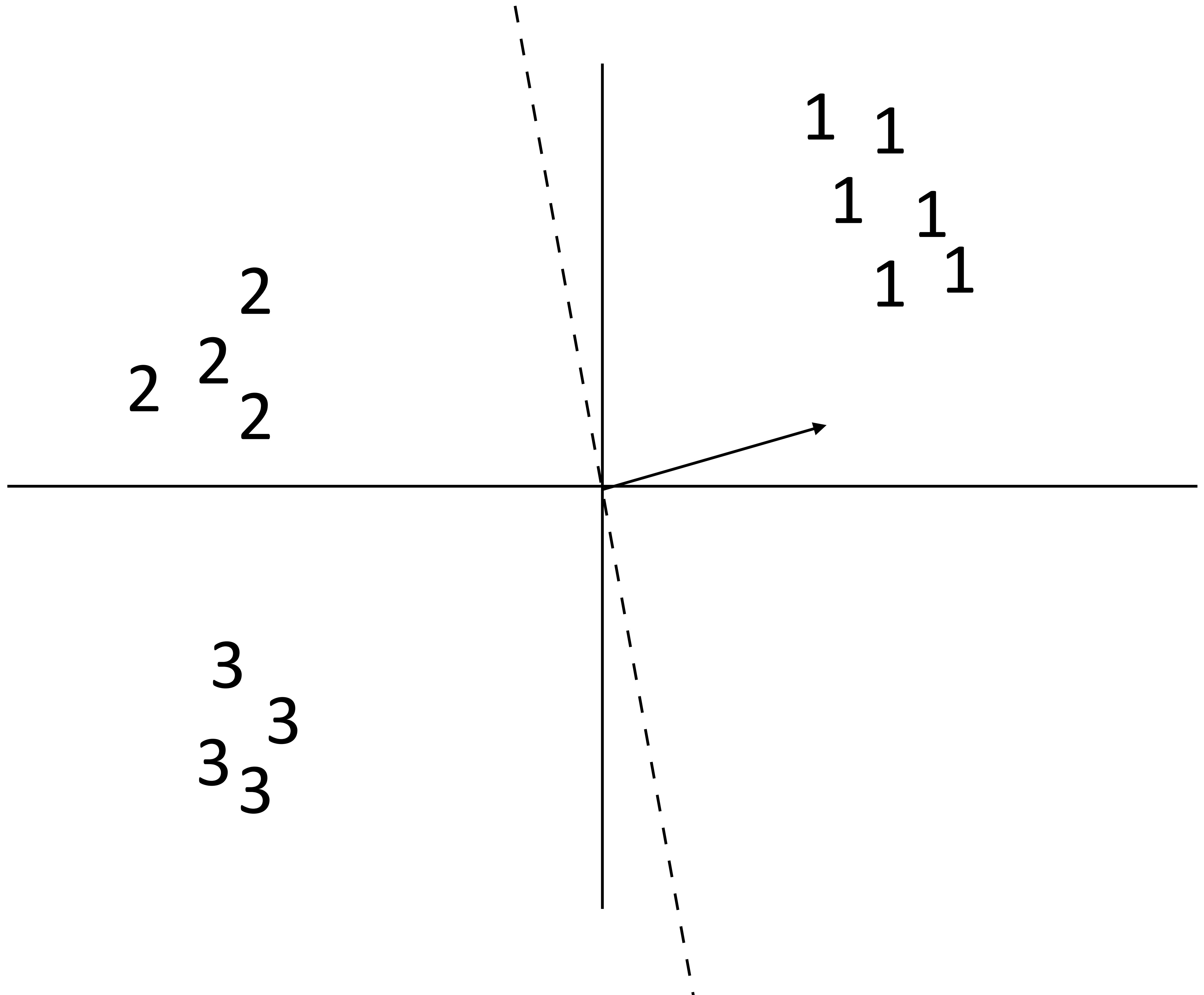
# Multiclass Classification

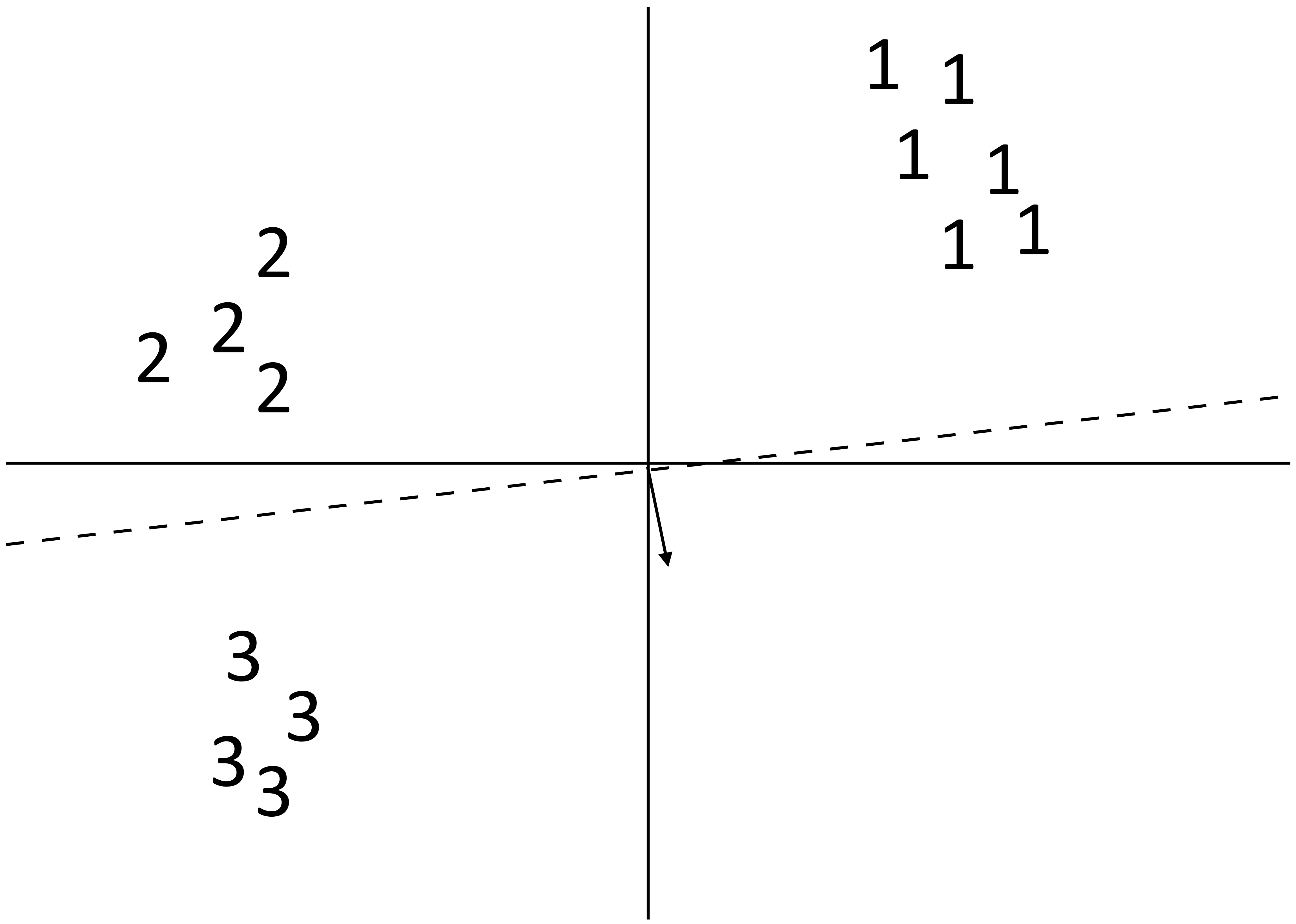
- One-vs-all: train *k* classifiers, one to distinguish each class from all the rest



# Multiclass Classification

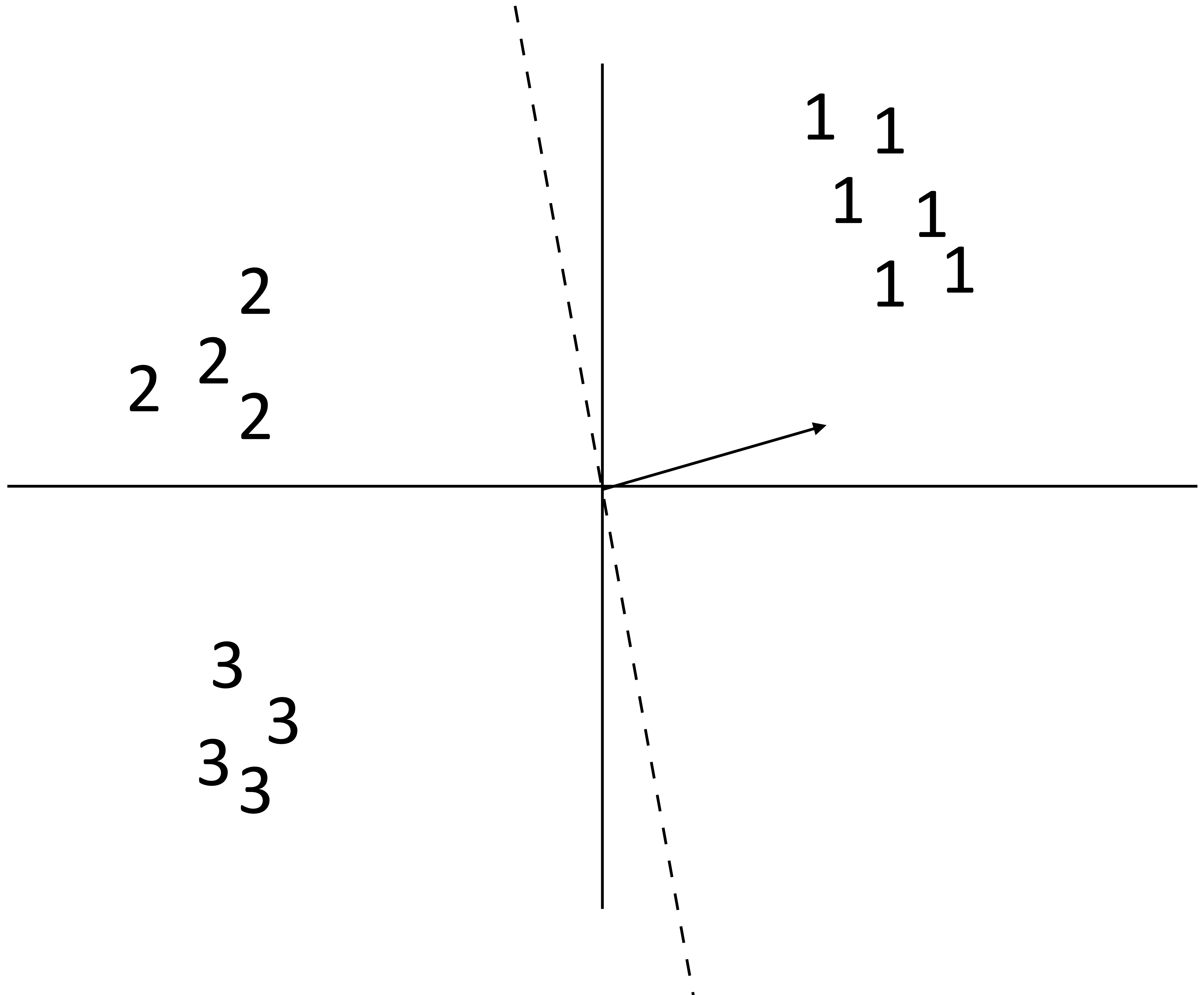
- One-vs-all: train *k* classifiers, one to distinguish each class from all the rest

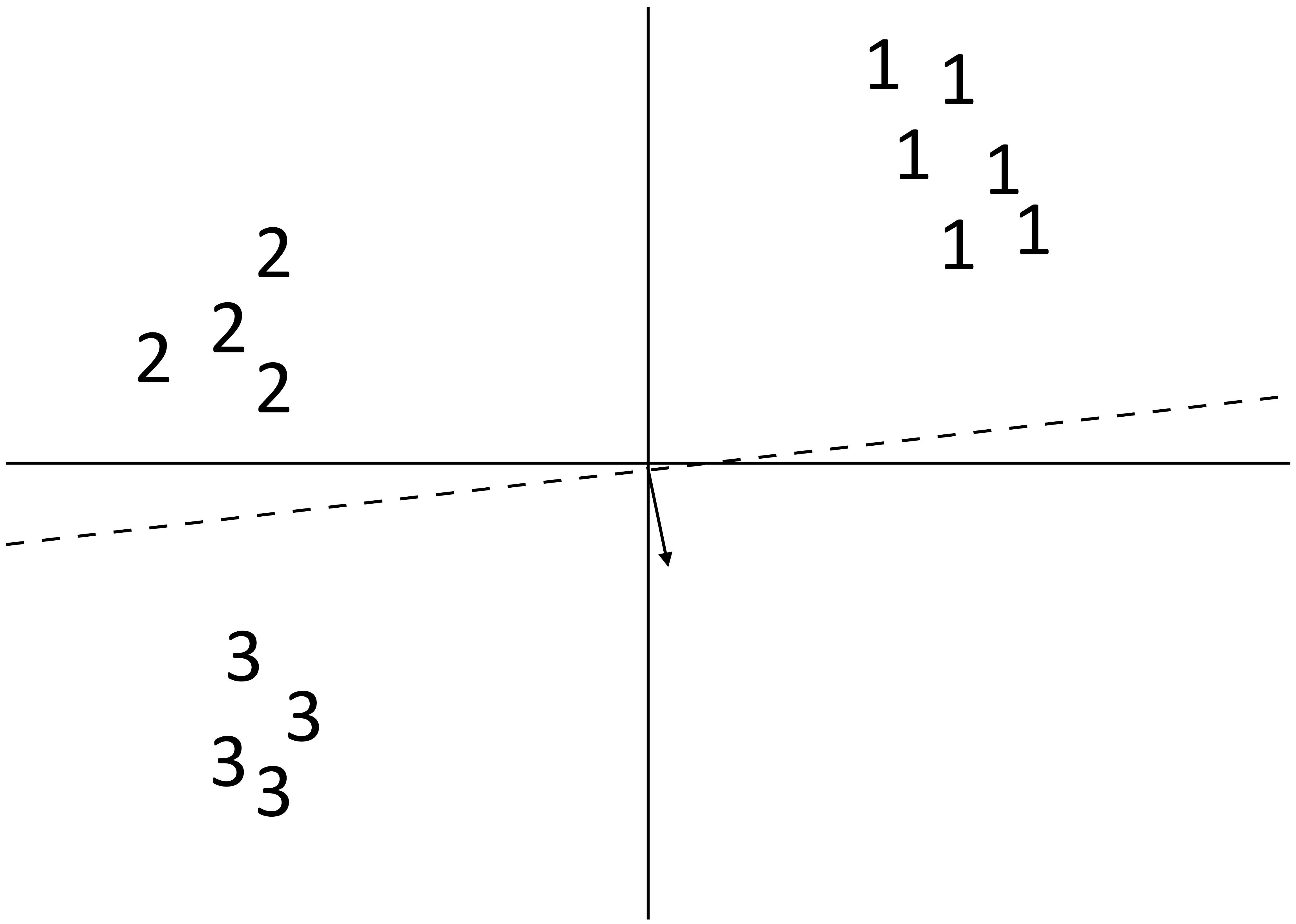


# Multiclass Classification

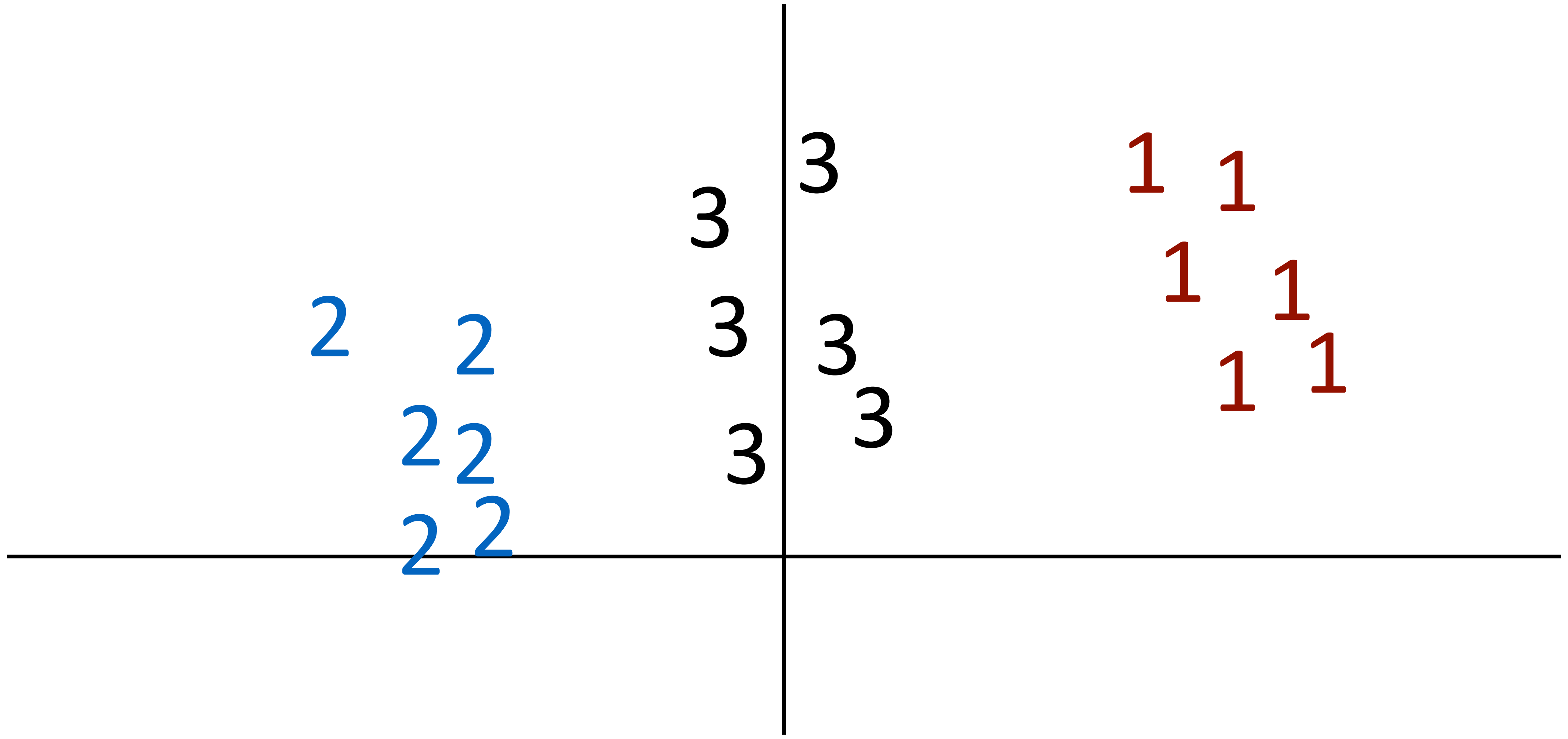
- One-vs-all: train *k* classifiers, one to distinguish each class from all the rest
- How do we reconcile multiple positive predictions? Highest score?



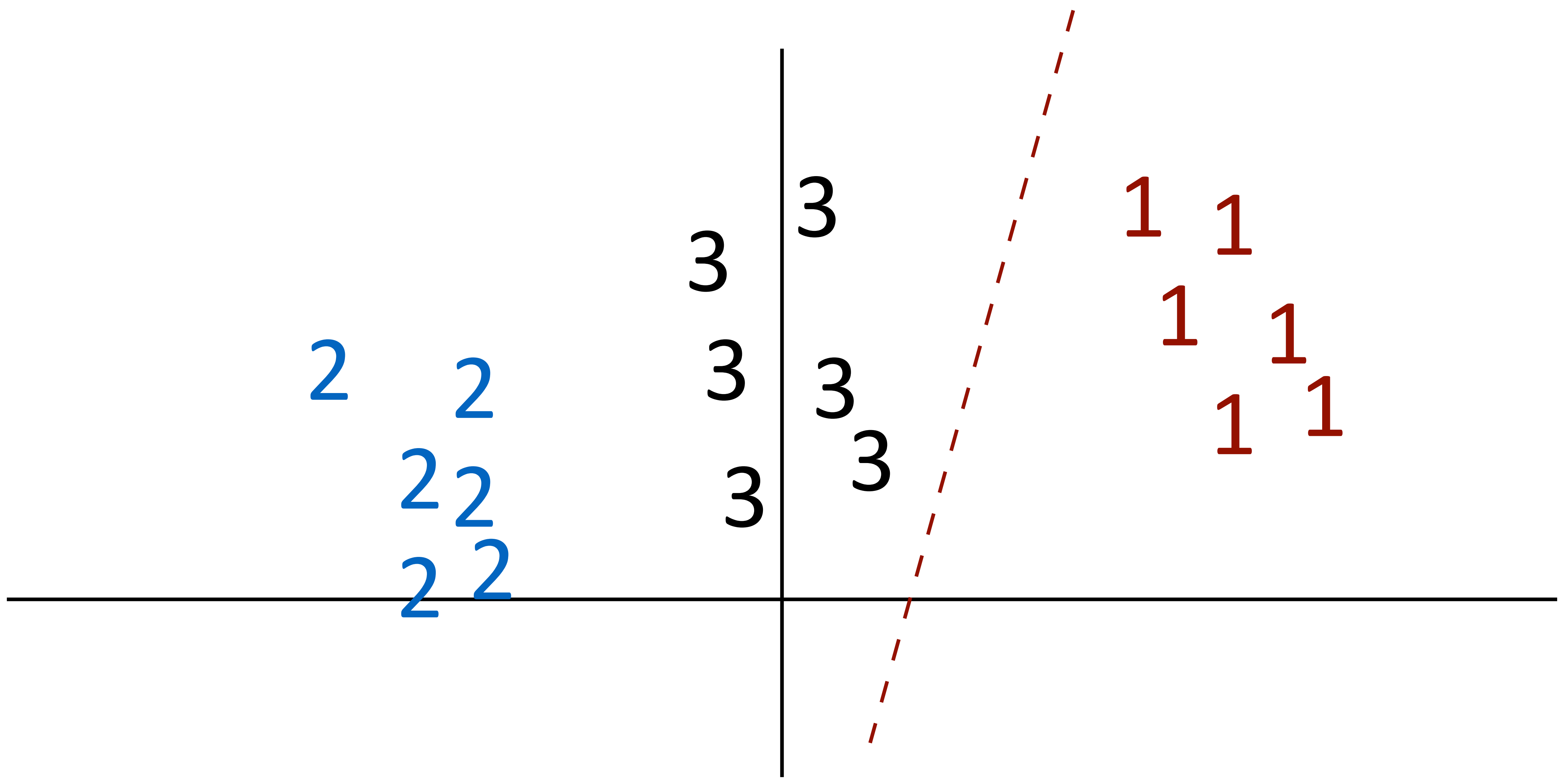
# Multiclass Classification

- Not all classes may even be separable using this approach



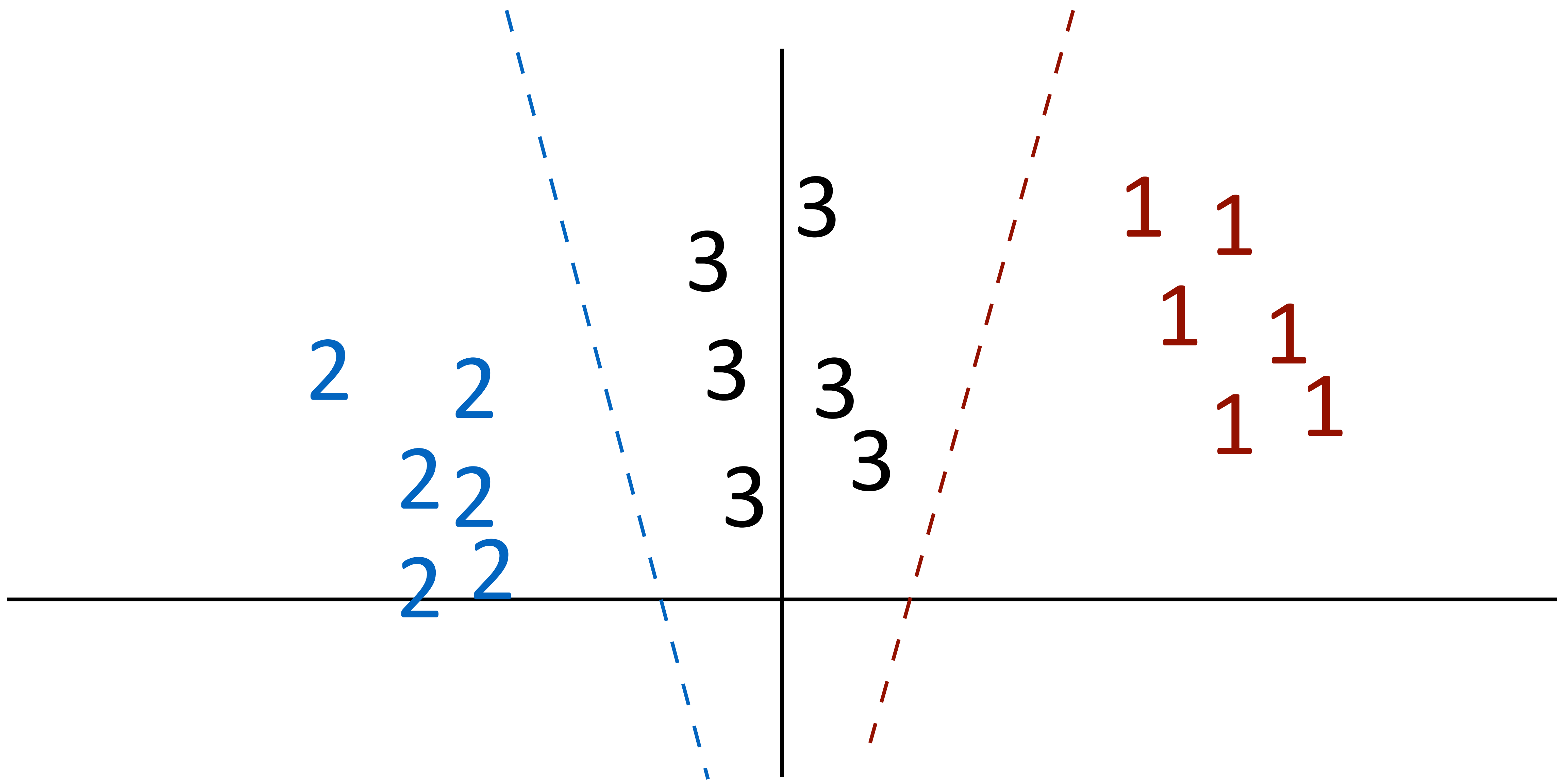
# Multiclass Classification

- Not all classes may even be separable using this approach



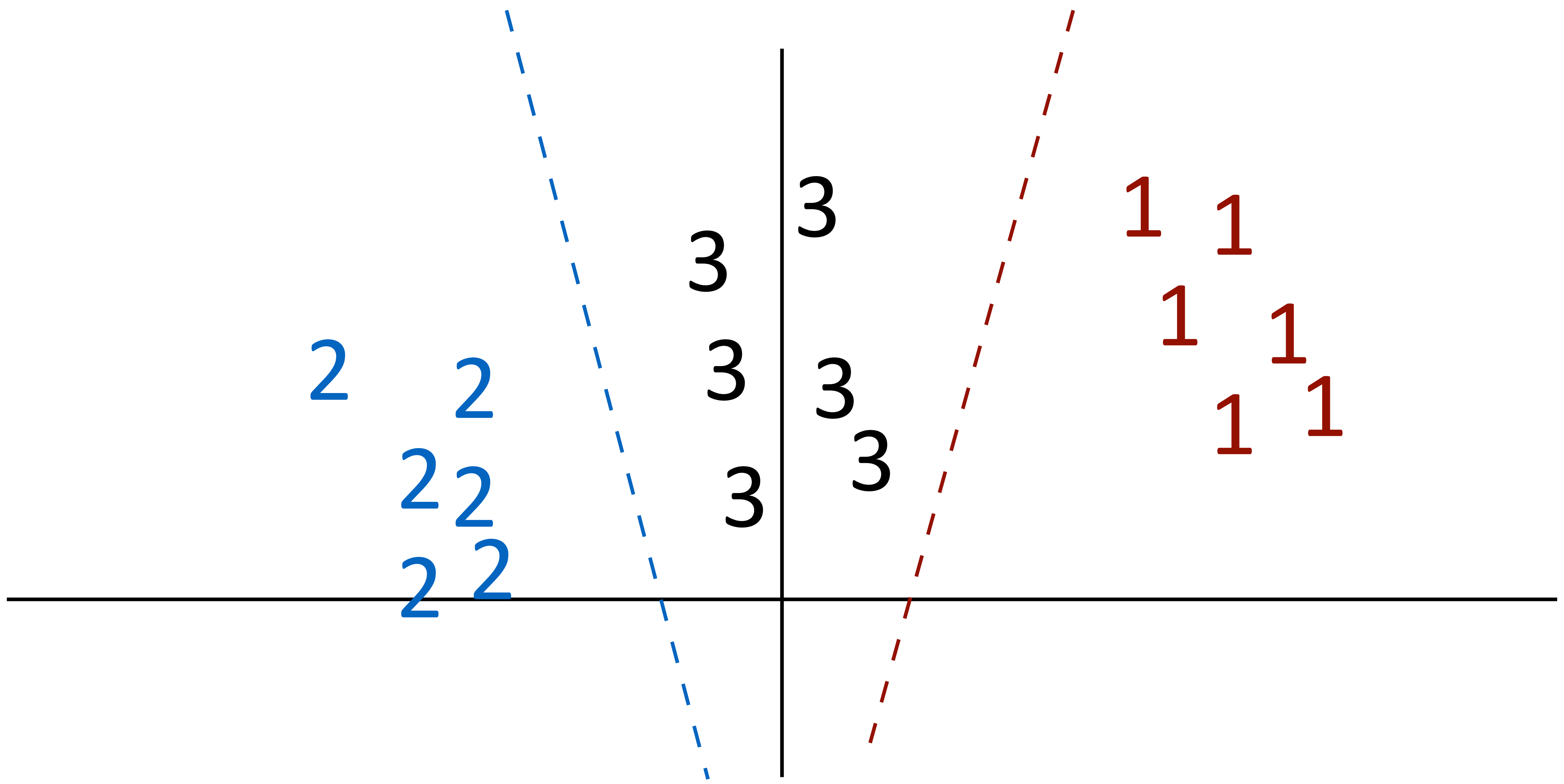
# Multiclass Classification

- Not all classes may even be separable using this approach



# Multiclass Classification

- Not all classes may even be separable using this approach



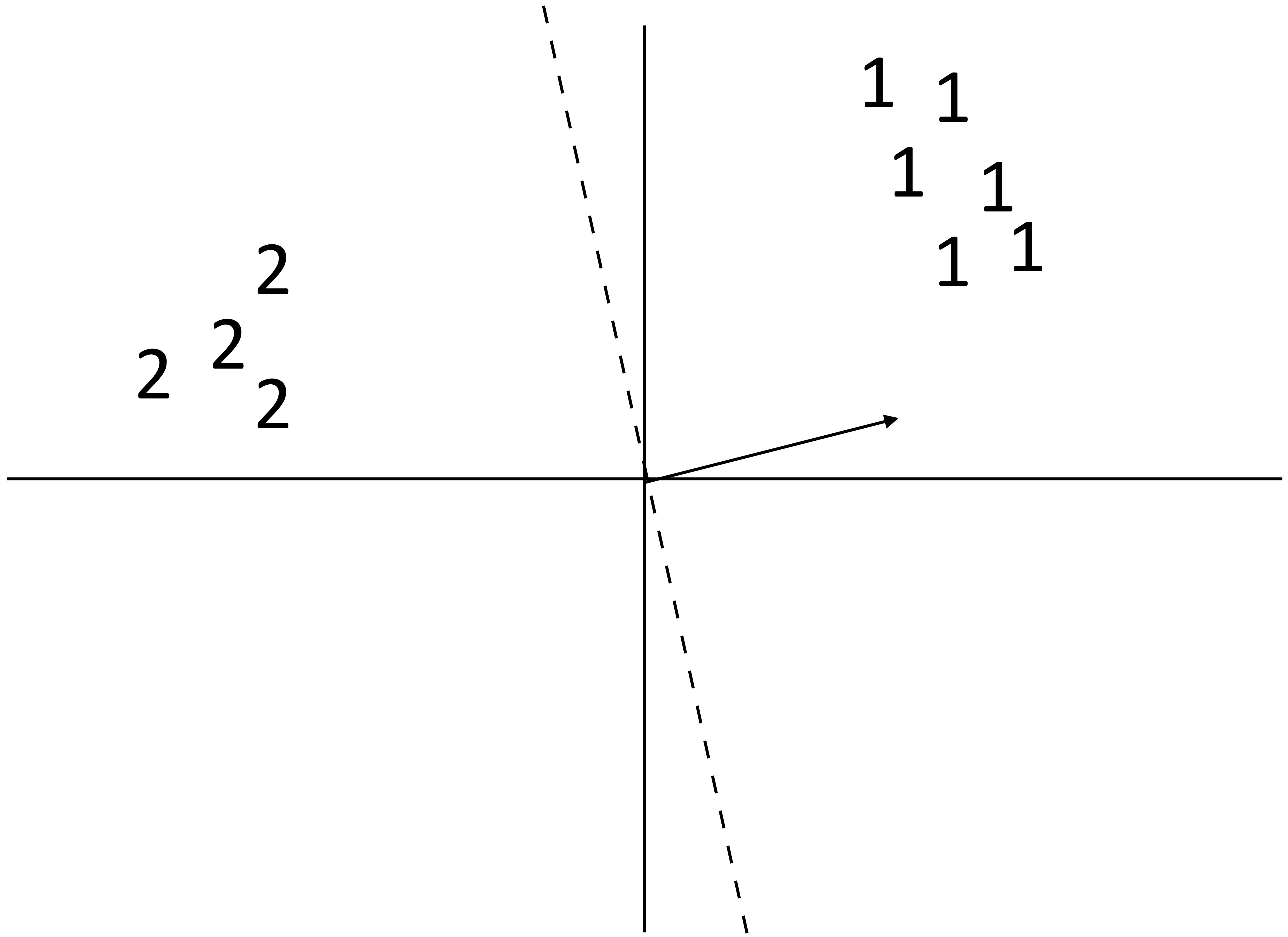

- Can separate 1 from 2+3 and 2 from 1+3 but not 3 from the others (with these features)

# Multiclass Classification

- All-vs-all: train n(n-1)/2 classifiers to differentiate each pair of classes

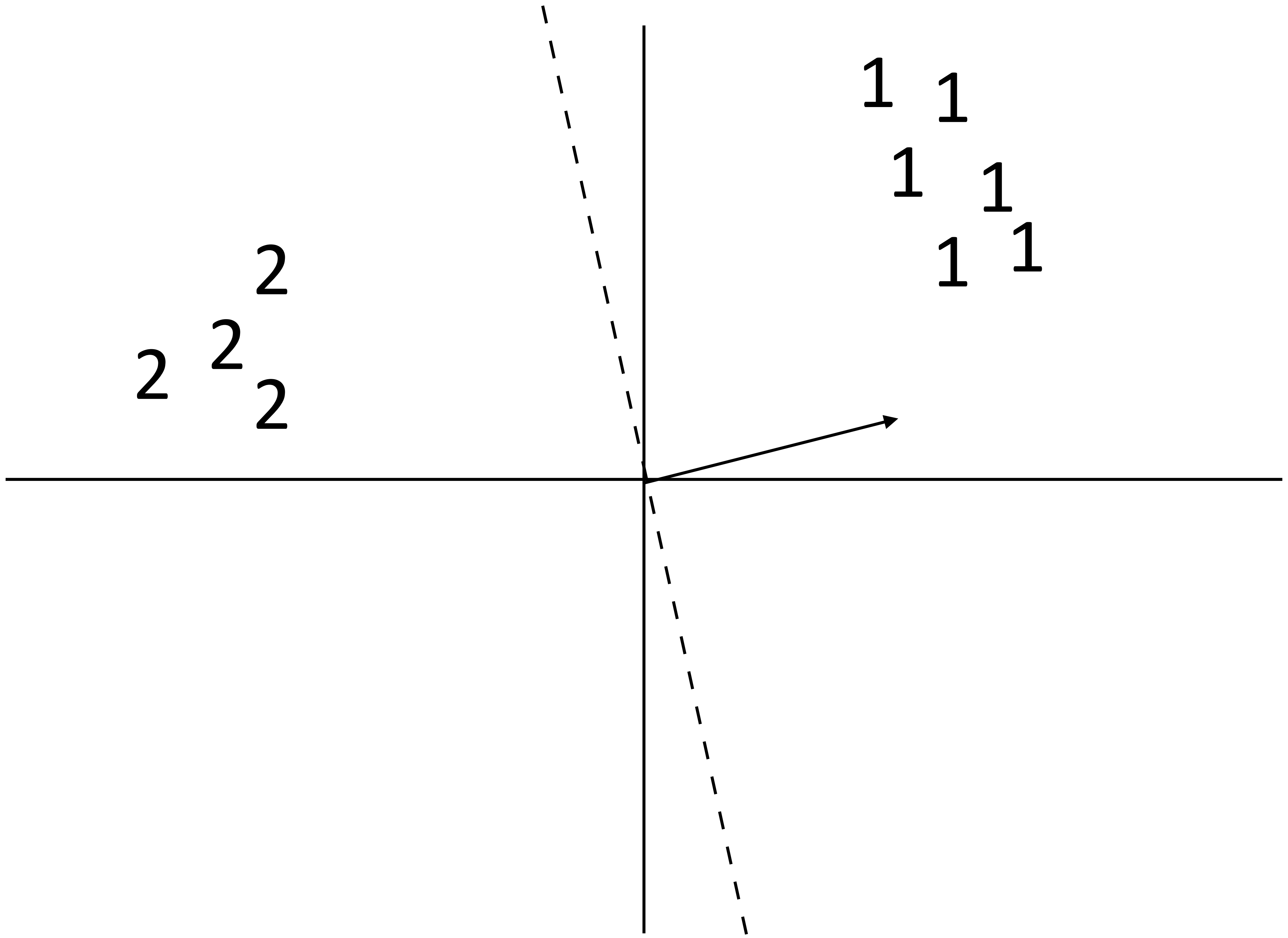
# Multiclass Classification

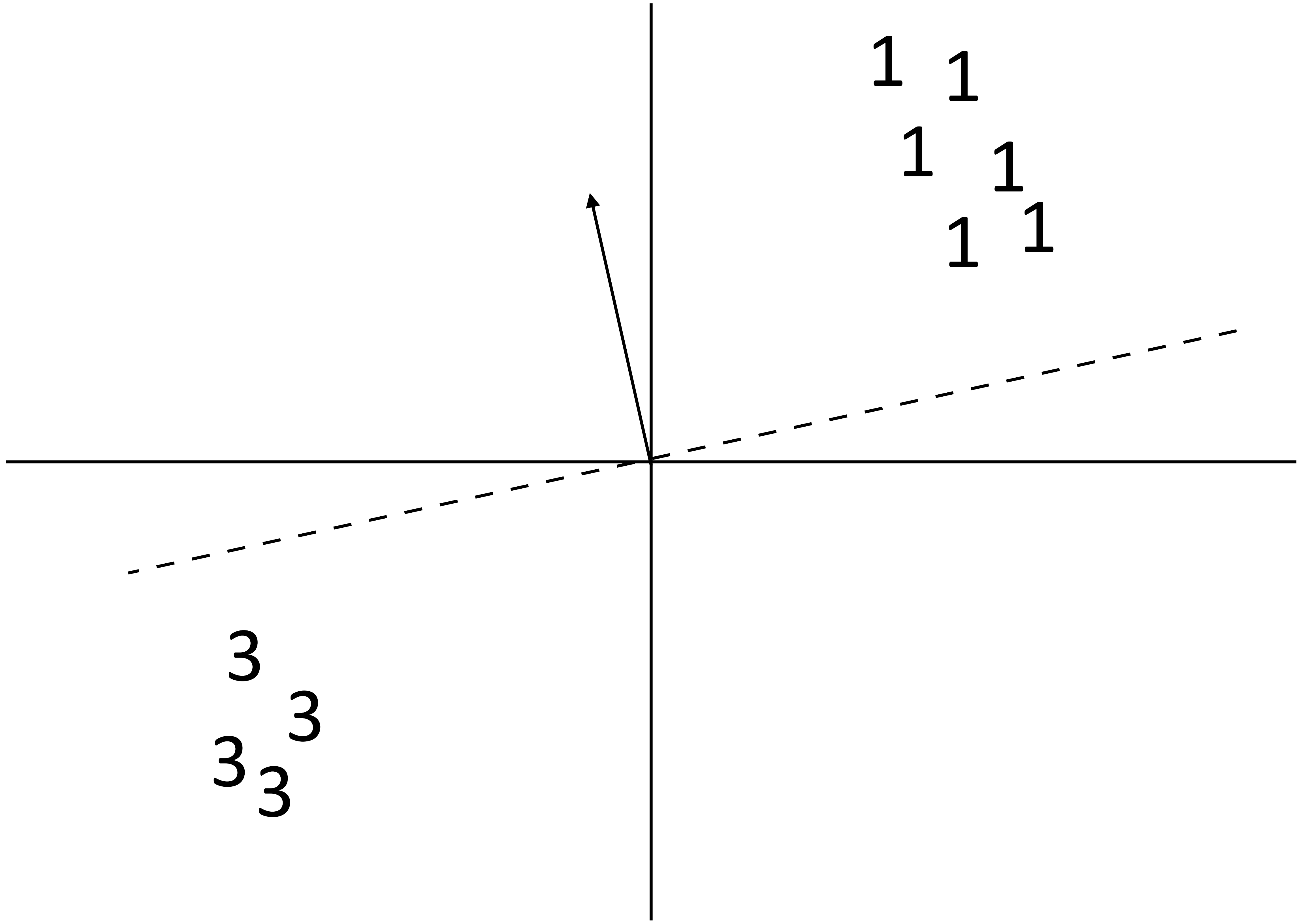
- All-vs-all: train n(n-1)/2 classifiers to differentiate each pair of classes



# Multiclass Classification

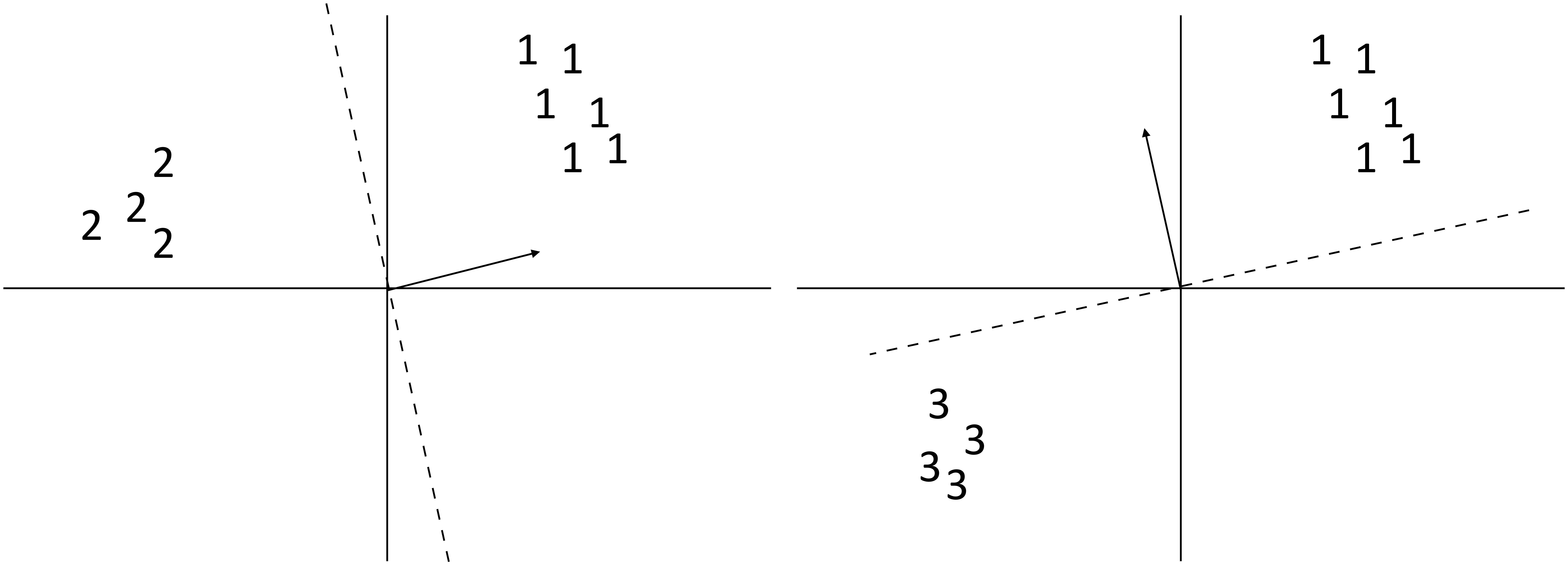
- All-vs-all: train n(n-1)/2 classifiers to differentiate each pair of classes



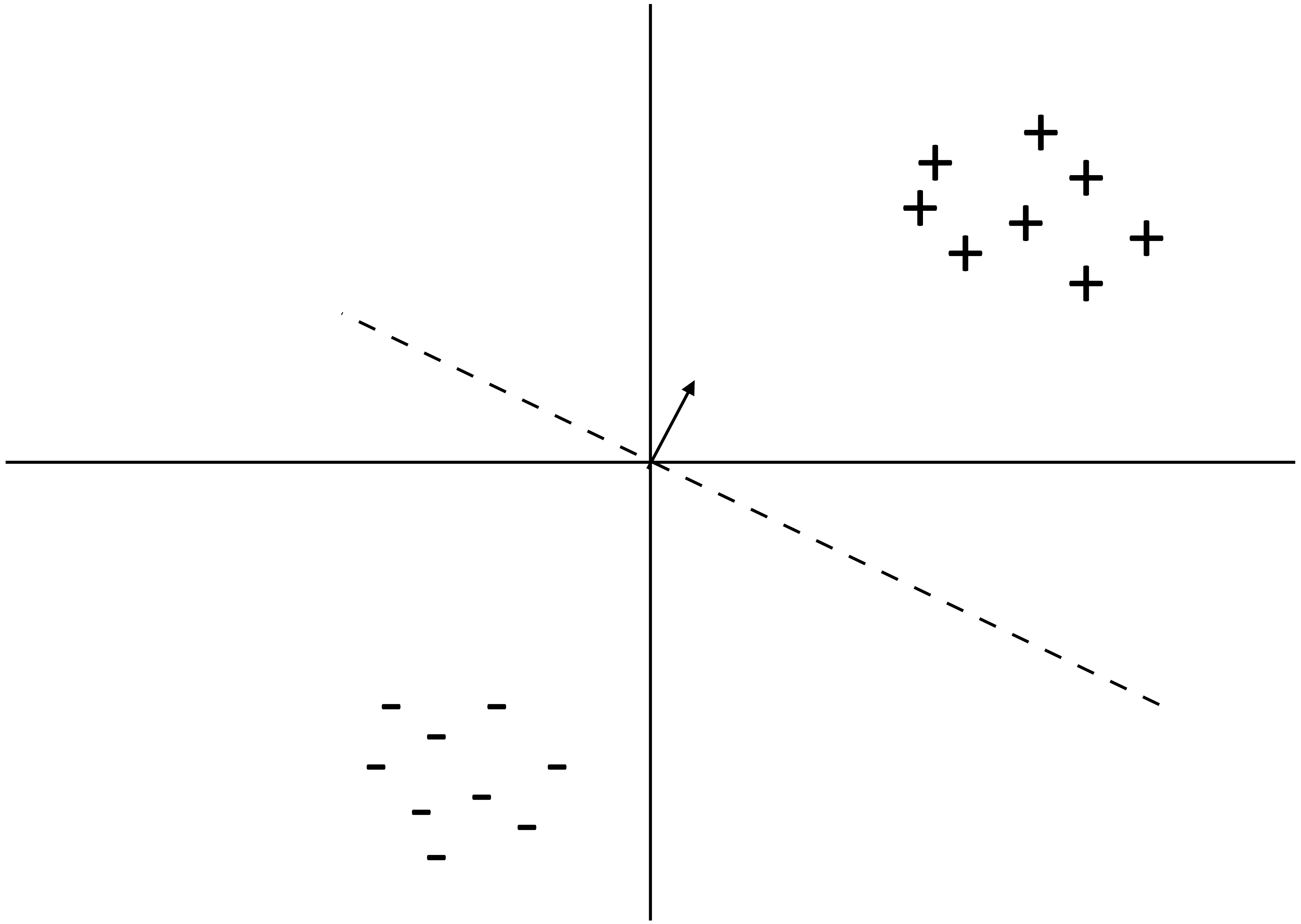
# Multiclass Classification

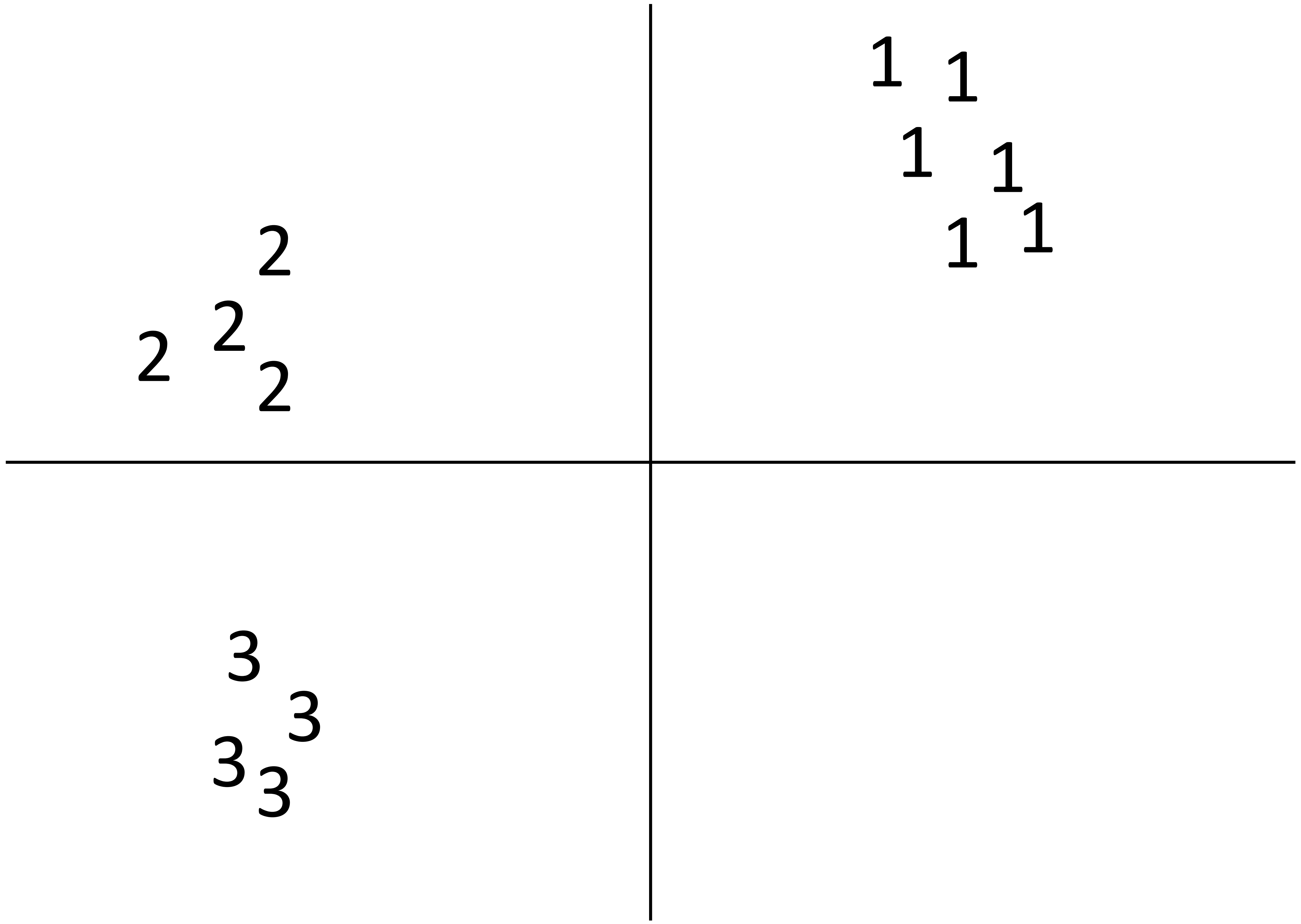
- All-vs-all: train n(n-1)/2 classifiers to differentiate each pair of classes
- Again, how to reconcile?



# Multiclass Classification

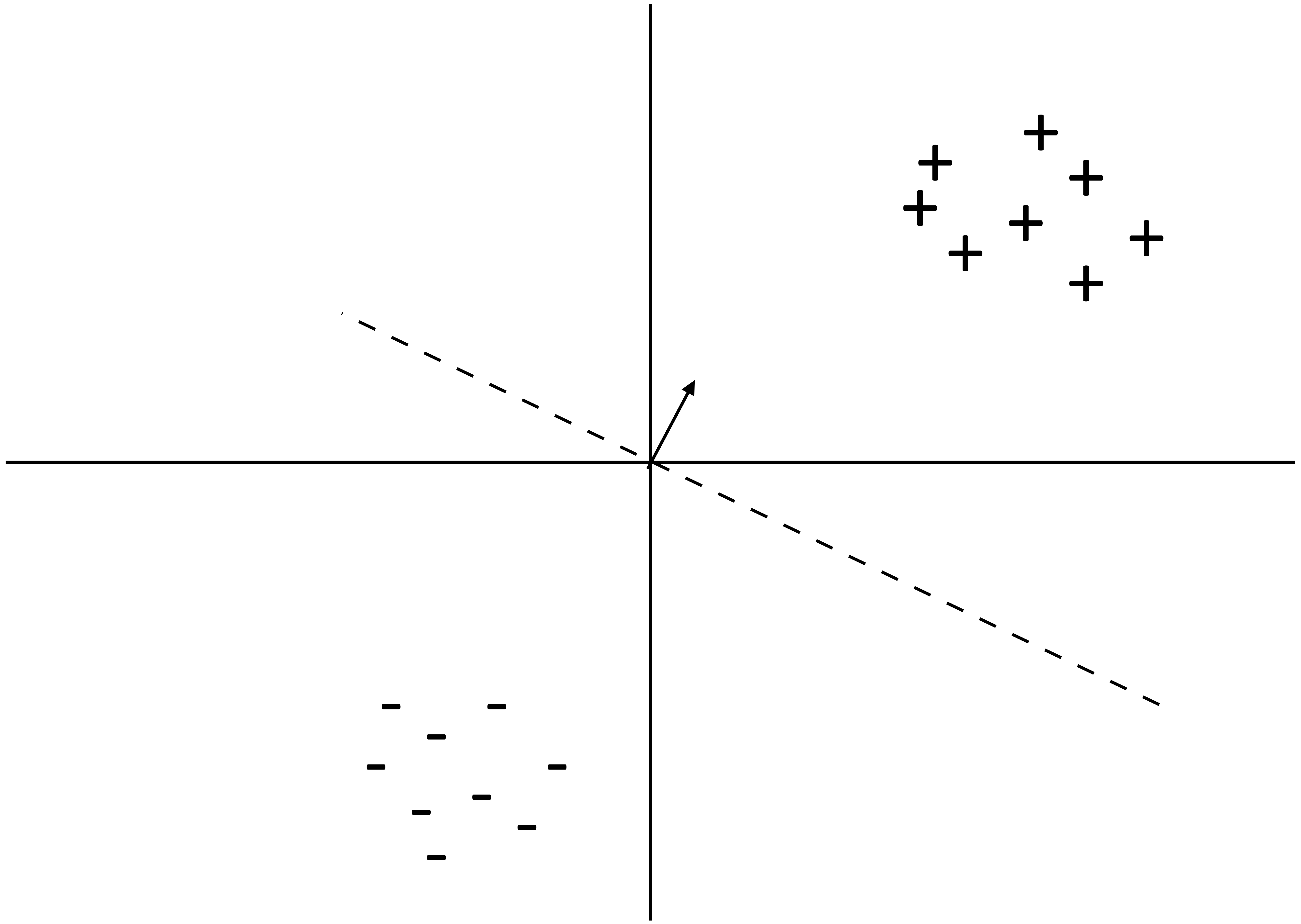
- Binary classification: one weight vector defines both classes

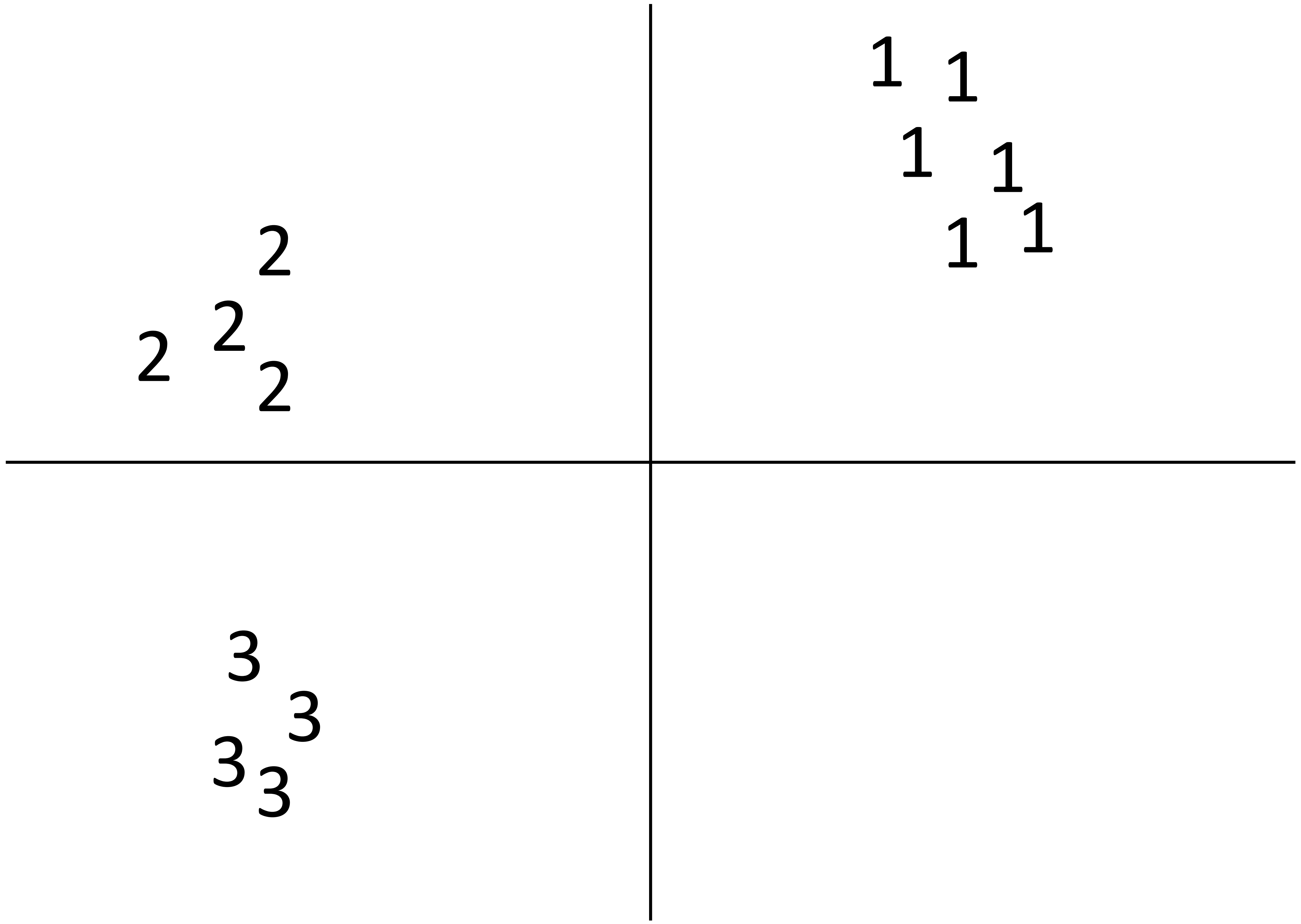


# Multiclass Classification

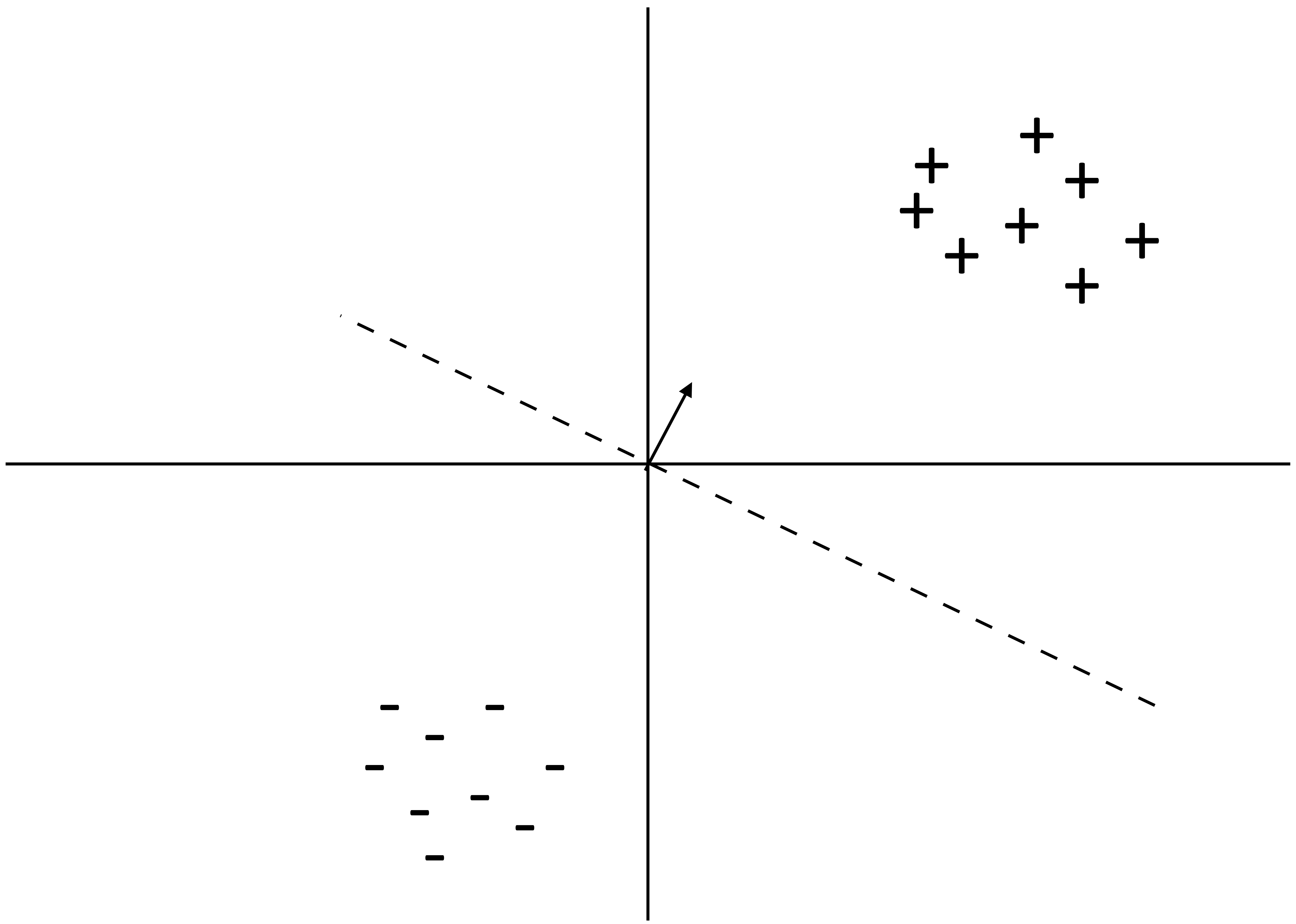
- Binary classification: one weight vector defines both classes
- Multiclass classification: different weights and/or features per class

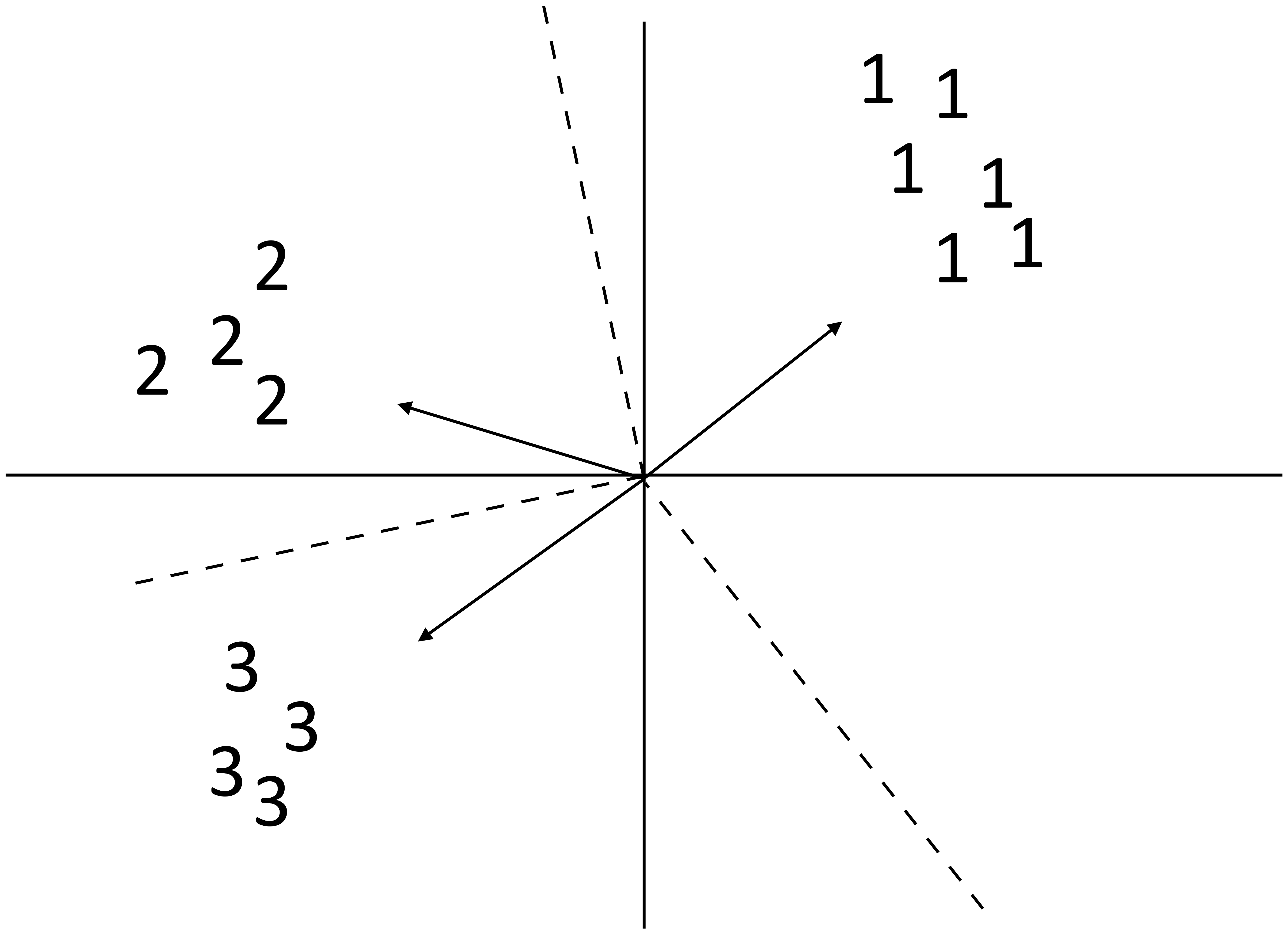




# Multiclass Classification

- Binary classification: one weight vector defines both classes
- Multiclass classification: different weights and/or features per class



# Multiclass Classification

# Multiclass Classification

- Formally: instead of two labels, we have an output space $\mathcal{Y}$ containing a number of possible classes
  - Same machinery that we'll use later for exponentially large output spaces, including sequences and trees

# Multiclass Classification

- Formally: instead of two labels, we have an output space $\mathcal{Y}$ containing a number of possible classes
  - Same machinery that we'll use later for exponentially large output spaces, including sequences and trees

- Decision rule: $\text{argmax}_{y \in \mathcal{Y}} w^\top f(x, y)$

# Multiclass Classification

- Formally: instead of two labels, we have an output space $\mathcal{Y}$ containing a number of possible classes
  - Same machinery that we'll use later for exponentially large output spaces, including sequences and trees
- Decision rule: $\text{argmax}_{y \in \mathcal{Y}} w^\top f(x, y)$ ← features depend on choice of label now! note: this isn't the gold label

# Multiclass Classification

- Formally: instead of two labels, we have an output space $\mathcal{Y}$ containing a number of possible classes
  - Same machinery that we'll use later for exponentially large output spaces, including sequences and trees
- Decision rule: $\text{argmax}_{y \in \mathcal{Y}} w^\top f(x, y)$ ← features depend on choice of label now! note: this isn't the gold label
  - Multiple feature vectors, one weight vector

# Multiclass Classification

- Formally: instead of two labels, we have an output space $\mathcal{Y}$ containing a number of possible classes
  - Same machinery that we'll use later for exponentially large output spaces, including sequences and trees
- Decision rule: $\text{argmax}_{y \in \mathcal{Y}} w^\top f(x, y)$ ← features depend on choice of label now! note: this isn't the gold label
  - Multiple feature vectors, one weight vector
  - Can also have one weight vector per class: $\text{argmax}_{y \in \mathcal{Y}} w_y^\top f(x)$

# Multiclass Classification

- Formally: instead of two labels, we have an output space $\mathcal{Y}$ containing a number of possible classes
  - Same machinery that we'll use later for exponentially large output spaces, including sequences and trees
- Decision rule: $\text{argmax}_{y \in \mathcal{Y}} w^\top f(x, y)$ ← features depend on choice of label now! note: this isn't the gold label
  - Multiple feature vectors, one weight vector
  - Can also have one weight vector per class: $\text{argmax}_{y \in \mathcal{Y}} w_y^\top f(x)$
  - The single weight vector approach will generalize to structured output spaces, whereas per-class weight vectors won't

# Feature Extraction

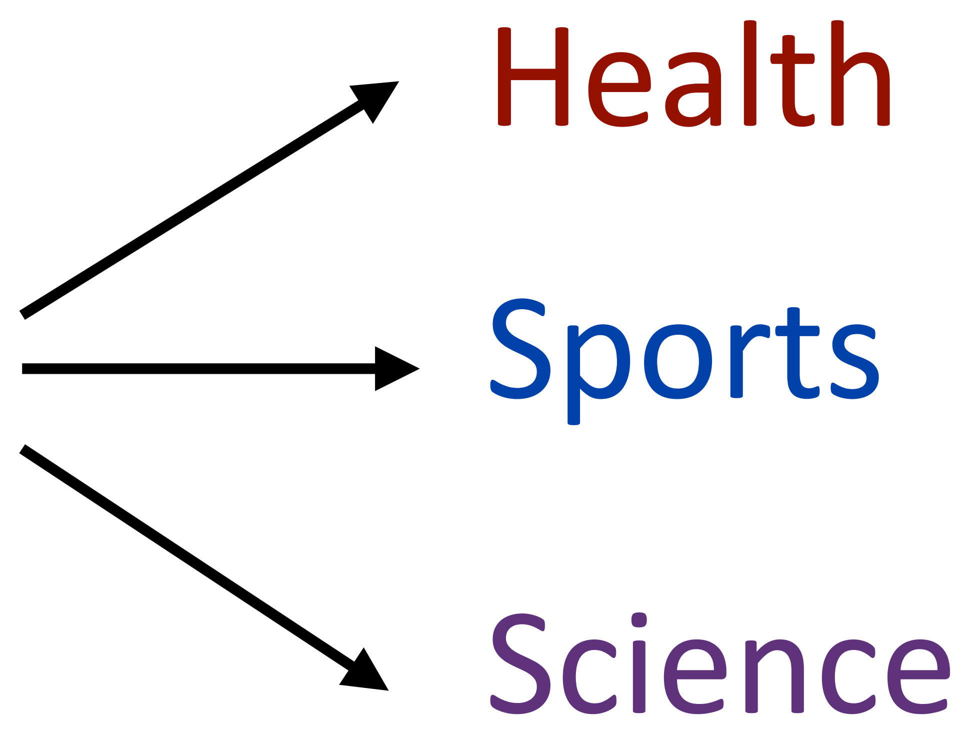
# Block Feature Vectors

- Decision rule: $\mathrm{argmax}_{y \in \mathcal{Y}} w^\top f(x, y)$

# Block Feature Vectors

- Decision rule: $\text{argmax}_{y \in \mathcal{Y}} w^\top f(x, y)$

*too many drug trials, too few patients*



# Block Feature Vectors

- Decision rule: $\text{argmax}_{y \in \mathcal{Y}} w^\top f(x, y)$

*too many drug trials, too few patients*

Health

Sports

Science

- Base feature function:

# Block Feature Vectors

- Decision rule: $\mathrm{argmax}_{y \in \mathcal{Y}} w^\top f(x, y)$

*too many drug trials, too few patients* → Health / Sports / Science

- Base feature function:

$f(x)$ = I[contains *drug*], I[contains *patients*], I[contains *baseball*]

# Block Feature Vectors

- Decision rule: $\text{argmax}_{y \in \mathcal{Y}} w^\top f(x, y)$

*too many drug trials, too few patients* → Health / Sports / Science

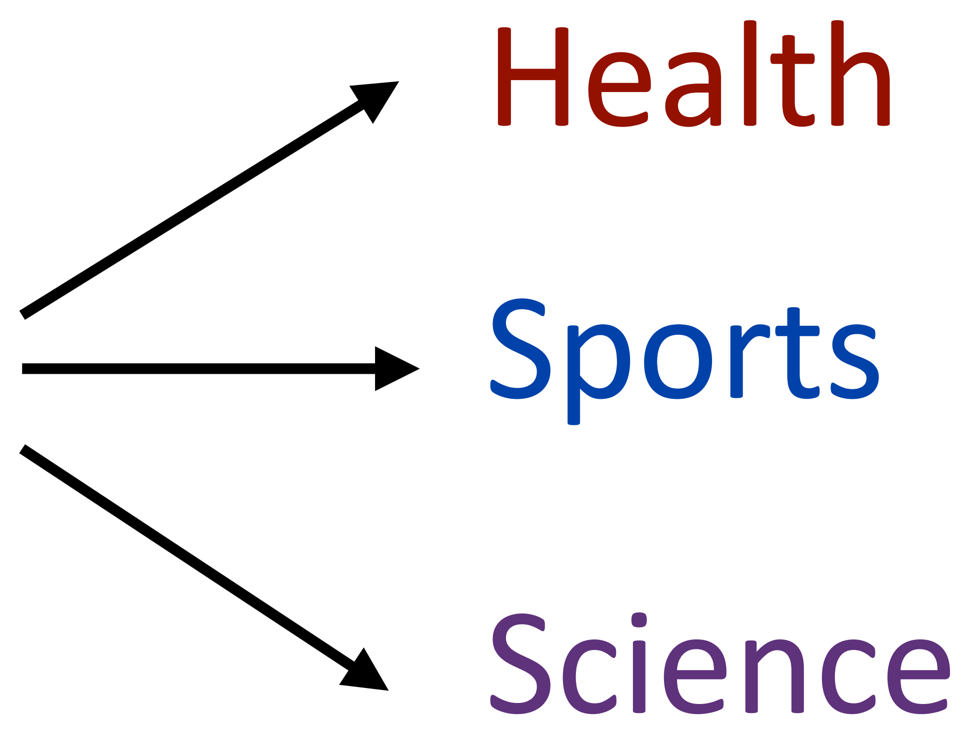
- Base feature function:

$f(x)$ = I[contains *drug*], I[contains *patients*], I[contains *baseball*] = [1, 1, 0]

# Block Feature Vectors

- Decision rule: $\text{argmax}_{y \in \mathcal{Y}} w^\top f(x, y)$

*too many drug trials, too few patients*




- Base feature function:

$f(x)$ = I[contains *drug*], I[contains *patients*], I[contains *baseball*] = [1, 1, 0]

$f(x, y =$ Health $) =$

# Block Feature Vectors

- Decision rule: $\text{argmax}_{y \in \mathcal{Y}} w^\top f(x, y)$

*too many drug trials, too few patients* → Health / Sports / Science

- Base feature function:

$f(x)$ = I[contains *drug*], I[contains *patients*], I[contains *baseball*] = [1, 1, 0]

$f(x, y =$ Health $) =$ [1, 1, 0, 0, 0, 0, 0, 0, 0]

# Block Feature Vectors

- Decision rule: $\text{argmax}_{y \in \mathcal{Y}} w^\top f(x, y)$

*too many drug trials, too few patients* → Health / Sports / Science

- Base feature function:

$f(x)$ = I[contains *drug*], I[contains *patients*], I[contains *baseball*] = [1, 1, 0]

feature vector blocks for each label

$f(x, y =$ Health $) =$ [1, 1, 0, 0, 0, 0, 0, 0, 0]

# Block Feature Vectors

- Decision rule: $\text{argmax}_{y \in \mathcal{Y}} w^\top f(x, y)$

*too many drug trials, too few patients* → Health / Sports / Science

- Base feature function:

$f(x)$ = I[contains *drug*], I[contains *patients*], I[contains *baseball*] = [1, 1, 0]

feature vector blocks for each label

$f(x, y =$ Health $) =$ [1, 1, 0, 0, 0, 0, 0, 0, 0]

$f(x, y =$ Sports $) =$ [0, 0, 0, 1, 1, 0, 0, 0, 0]

# Block Feature Vectors

- Decision rule: $\text{argmax}_{y \in \mathcal{Y}} w^\top f(x, y)$

*too many drug trials, too few patients* → Health, Sports, Science

- Base feature function:

$f(x)$ = I[contains *drug*], I[contains *patients*], I[contains *baseball*] = [1, 1, 0]

feature vector blocks for each label

$f(x, y =$ Health $) =$ [1, 1, 0, 0, 0, 0, 0, 0, 0]

I[contains *drug* & label = Health]

$f(x, y =$ Sports $) =$ [0, 0, 0, 1, 1, 0, 0, 0, 0]

# Block Feature Vectors

- Decision rule: $\text{argmax}_{y \in \mathcal{Y}} w^\top f(x, y)$

*too many drug trials, too few patients* → Health / Sports / Science

- Base feature function:

$f(x)$ = I[contains *drug*], I[contains *patients*], I[contains *baseball*] = [1, 1, 0]

feature vector blocks for each label

$f(x, y =$ Health $) =$ [1, 1, 0, 0, 0, 0, 0, 0, 0]    I[contains *drug* & label = Health]

$f(x, y =$ Sports $) =$ [0, 0, 0, 1, 1, 0, 0, 0, 0]

- Equivalent to having three weight vectors in this case

# Making Decisions

*too many drug trials, too few patients* → Health / Sports / Science

$f(x)$ = I[contains *drug*], I[contains *patients*], I[contains *baseball*]

$f(x, y =$ Health $) =$ [1, 1, 0, 0, 0, 0, 0, 0, 0]

$f(x, y =$ Sports $) =$ [0, 0, 0, 1, 1, 0, 0, 0, 0]

# Making Decisions

*too many drug trials, too few patients* → Health / Sports / Science

$f(x)$ = I[contains *drug*], I[contains *patients*], I[contains *baseball*]

$f(x, y =$ Health $) =$ [1, 1, 0, 0, 0, 0, 0, 0, 0]

$f(x, y =$ Sports $) =$ [0, 0, 0, 1, 1, 0, 0, 0, 0]

$w =$ [+2.1, +2.3, -5, -2.1, -3.8, 0, +1.1, -1.7, -1.3]

# Making Decisions

*too many drug trials, too few patients* → Health / Sports / Science

$f(x)$ = I[contains *drug*], I[contains *patients*], I[contains *baseball*]

$f(x, y =$ Health $) =$ [1, 1, 0, 0, 0, 0, 0, 0, 0]

$f(x, y =$ Sports $) =$ [0, 0, 0, 1, 1, 0, 0, 0, 0]

"word drug in Science article" = +1.1

$w =$ [+2.1, +2.3, -5, -2.1, -3.8, 0, +1.1, -1.7, -1.3]

# Making Decisions

*too many drug trials, too few patients* → Health / Sports / Science

$f(x)$ = I[contains *drug*], I[contains *patients*], I[contains *baseball*]

$f(x, y =$ Health $) =$ [1, 1, 0, 0, 0, 0, 0, 0, 0]

$f(x, y =$ Sports $) =$ [0, 0, 0, 1, 1, 0, 0, 0, 0]

"word drug in Science article" = +1.1

$w =$ [+2.1, +2.3, -5, -2.1, -3.8, 0, +1.1, -1.7, -1.3]

$w^\top f(x, y)$ =

# Making Decisions

*too many drug trials, too few patients* → Health / Sports / Science

$f(x)$ = I[contains *drug*], I[contains *patients*], I[contains *baseball*]

$f(x, y =$ Health $) =$ [1, 1, 0, 0, 0, 0, 0, 0, 0]

$f(x, y =$ Sports $) =$ [0, 0, 0, 1, 1, 0, 0, 0, 0]

"word drug in Science article" = +1.1

$w =$ [+2.1, +2.3, -5, -2.1, -3.8, 0, +1.1, -1.7, -1.3]

$w^\top f(x, y)$ = Health: +4.4 Sports: -5.9 Science: -0.6

# Making Decisions

*too many drug trials, too few patients* → Health / Sports / Science

$f(x)$ = I[contains *drug*], I[contains *patients*], I[contains *baseball*]

$f(x, y =$ Health $) =$ [1, 1, 0, 0, 0, 0, 0, 0, 0]

$f(x, y =$ Sports $) =$ [0, 0, 0, 1, 1, 0, 0, 0, 0]

"word drug in Science article" = +1.1

$w =$ [+2.1, +2.3, -5, -2.1, -3.8, 0, +1.1, -1.7, -1.3]

$w^\top f(x, y)$ = Health: +4.4 Sports: -5.9 Science: -0.6

↖ argmax

# Another example: POS tagging

*blocks*

# Another example: POS tagging

*the router* *blocks* *the packets*

# Another example: POS tagging

*the router* *blocks* *the packets*

NNS
VBZ
NN
DT
…

# Another example: POS tagging

- Classify *blocks* as one of 36 POS tags

*the router* *blocks* *the packets*

NNS
VBZ
NN
DT
...

# Another example: POS tagging

- Classify *blocks* as one of 36 POS tags
- Example *x*: sentence with a word (in this case, *blocks*) highlighted

*the router* *blocks* *the packets*

NNS
VBZ
NN
DT
...

# Another example: POS tagging

- Classify *blocks* as one of 36 POS tags
- Example *x*: sentence with a word (in this case, *blocks*) highlighted
- Extract features with respect to this word:

*the router* *blocks* *the packets*

NNS
VBZ
NN
DT
…

# Another example: POS tagging

- Classify *blocks* as one of 36 POS tags
- Example *x*: sentence with a word (in this case, *blocks*) highlighted
- Extract features with respect to this word:

*the router* *blocks* *the packets*

NNS
VBZ
NN
DT
...

*f*(*x*, *y*=VBZ) = I[curr_word=*blocks* & tag = VBZ],
I[prev_word=*router* & tag = VBZ]
I[next_word=*the* & tag = VBZ]
I[curr_suffix=*s* & tag = VBZ]

# Another example: POS tagging

- Classify *blocks* as one of 36 POS tags
- Example *x*: sentence with a word (in this case, *blocks*) highlighted
- Extract features with respect to this word:

*the router* *blocks* *the packets*

NNS
VBZ
NN
DT
…

*f*(*x*, *y*=VBZ) = I[curr_word=*blocks* & tag = VBZ],
I[prev_word=*router* & tag = VBZ]
I[next_word=*the* & tag = VBZ]
I[curr_suffix=*s* & tag = VBZ]

not saying that *the* is tagged as VBZ! saying that *the* follows the VBZ word

# Multiclass Logistic Regression

# Multiclass Logistic Regression

$$P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$$

sum over output
space to normalize

# Multiclass Logistic Regression

$$P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$$

sum over output space to normalize

- Compare to binary:

$$P(y=1|x) = \frac{\exp(w^\top f(x))}{1+\exp(w^\top f(x))}$$

# Multiclass Logistic Regression

$$P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$$

sum over output space to normalize

- Compare to binary:

$$P(y=1|x) = \frac{\exp(w^\top f(x))}{1+\exp(w^\top f(x))}$$

negative class implicitly had $f(x, y=0)$ = the zero vector

# Multiclass Logistic Regression

$$P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$$

sum over output space to normalize

# Multiclass Logistic Regression

$$P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$$

Softmax function

sum over output space to normalize

# Multiclass Logistic Regression

$$P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$$

Softmax function

sum over output space to normalize

Why? Interpret raw classifier scores as **probabilities**

# Multiclass Logistic Regression

$$P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$$

Softmax function

sum over output space to normalize

Why? Interpret raw classifier scores as **probabilities**

*too many drug trials, too few patients*

# Multiclass Logistic Regression

$$P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$$

Softmax function

sum over output space to normalize

Why? Interpret raw classifier scores as **probabilities**

*too many drug trials, too few patients*

Health: +2.2

Sports: +3.1

Science: -0.6

$w^\top f(x,y)$

# Multiclass Logistic Regression

$$P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$$

Softmax function

sum over output space to normalize

Why? Interpret raw classifier scores as **probabilities**

*too many drug trials, too few patients*

Health: +2.2

Sports: +3.1

Science: -0.6

$w^\top f(x,y)$

exp →

probabilities must be >= 0

| |
|---|
| 6.05 |
| 22.2 |
| 0.55 |

unnormalized probabilities

# Multiclass Logistic Regression

$$P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$$

Softmax function

sum over output space to normalize

Why? Interpret raw classifier scores as **probabilities**

*too many drug trials, too few patients*

| $w^\top f(x,y)$ | exp → | probabilities must be >= 0 (unnormalized probabilities) | normalize → | probabilities must sum to 1 (probabilities) |
|---|---|---|---|---|
| Health: +2.2 | | 6.05 | | 0.21 |
| Sports: +3.1 | | 22.2 | | 0.77 |
| Science: -0.6 | | 0.55 | | 0.02 |

# Multiclass Logistic Regression

Softmax function

$$P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$$

sum over output space to normalize

Why? Interpret raw classifier scores as **probabilities**

*too many drug trials, too few patients*

Health: +2.2

Sports: +3.1

Science: -0.6

$w^\top f(x,y)$

exp →

probabilities must be >= 0

6.05

22.2

0.55

unnormalized probabilities

normalize →

probabilities must sum to 1

0.21

0.77

0.02

probabilities

1.00

0.00

0.00

correct (gold) probabilities

# Multiclass Logistic Regression

$$P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$$

Softmax function

sum over output space to normalize

Why? Interpret raw classifier scores as **probabilities**

*too many drug trials, too few patients*

| $w^\top f(x,y)$ | exp → unnormalized probabilities (probabilities must be >= 0) | normalize → probabilities (probabilities must sum to 1) | compare ↔ correct (gold) probabilities |
|---|---|---|---|
| Health: +2.2 | 6.05 | 0.21 | 1.00 |
| Sports: +3.1 | 22.2 | 0.77 | 0.00 |
| Science: -0.6 | 0.55 | 0.02 | 0.00 |

$$\mathcal{L}(x_j, y_j^*) = \log P(y_j^*|x_j)$$

# Multiclass Logistic Regression

$$P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$$

Softmax function

sum over output space to normalize

Why? Interpret raw classifier scores as **probabilities**

*too many drug trials, too few patients*

| $w^\top f(x,y)$ | | probabilities must be >= 0 (unnormalized probabilities) | | probabilities must sum to 1 (probabilities) | | correct (gold) probabilities |
|---|---|---|---|---|---|---|
| Health: +2.2 | exp → | 6.05 | normalize → | 0.21 | log(0.21) = - 1.56 | 1.00 |
| Sports: +3.1 | | 22.2 | | 0.77 | ← compare → | 0.00 |
| Science: -0.6 | | 0.55 | | 0.02 | $\mathcal{L}(x_j, y_j^*) = \log P(y_j^*|x_j)$ | 0.00 |

# Multiclass Logistic Regression

$$P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$$

sum over output space to normalize

# Multiclass Logistic Regression

$$P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$$

sum over output space to normalize

- Training: maximize $\mathcal{L}(x,y) = \sum_{j=1}^{n} \log P(y_j^*|x_j)$

# Multiclass Logistic Regression

$$P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$$

sum over output space to normalize

- Training: maximize $\mathcal{L}(x,y) = \sum_{j=1}^{n} \log P(y_j^*|x_j)$

$$= \sum_{j=1}^{n} \left( w^\top f(x_j, y_j^*) - \log \sum_{y} \exp(w^\top f(x_j, y)) \right)$$

# Training

- Multiclass logistic regression $P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$

- Likelihood $\mathcal{L}(x_j, y_j^*) = w^\top f(x_j, y_j^*) - \log \sum_y \exp(w^\top f(x_j, y))$

# Training

- Multiclass logistic regression $P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$

- Likelihood $\mathcal{L}(x_j, y_j^*) = w^\top f(x_j, y_j^*) - \log \sum_y \exp(w^\top f(x_j, y))$

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \frac{\sum_y f_i(x_j, y) \exp(w^\top f(x_j, y))}{\sum_y \exp(w^\top f(x_j, y))}$$

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \sum_y f_i(x_j, y) P_w(y|x_j)$$

# Training

- Multiclass logistic regression $P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$
- Likelihood $\mathcal{L}(x_j, y_j^*) = w^\top f(x_j, y_j^*) - \log \sum_y \exp(w^\top f(x_j, y))$

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \frac{\sum_y f_i(x_j, y) \exp(w^\top f(x_j, y))}{\sum_y \exp(w^\top f(x_j, y))}$$

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \sum_y f_i(x_j, y) P_w(y|x_j)$$

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \mathbb{E}_y[f_i(x_j, y)]$$

# Training

- Multiclass logistic regression $P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$

- Likelihood $\mathcal{L}(x_j, y_j^*) = w^\top f(x_j, y_j^*) - \log \sum_y \exp(w^\top f(x_j, y))$

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \frac{\sum_y f_i(x_j, y) \exp(w^\top f(x_j, y))}{\sum_y \exp(w^\top f(x_j, y))}$$

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \sum_y f_i(x_j, y) P_w(y|x_j)$$

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \mathbb{E}_y[f_i(x_j, y)]$$

gold feature value

# Training

- Multiclass logistic regression $P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$

- Likelihood $\mathcal{L}(x_j, y_j^*) = w^\top f(x_j, y_j^*) - \log \sum_y \exp(w^\top f(x_j, y))$

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \frac{\sum_y f_i(x_j, y)\exp(w^\top f(x_j, y))}{\sum_y \exp(w^\top f(x_j, y))}$$

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \sum_y f_i(x_j, y) P_w(y|x_j)$$

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \mathbb{E}_y[f_i(x_j, y)]$$

gold feature value

model's expectation of feature value

# Training

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \sum_y f_i(x_j, y) P_w(y|x_j)$$

*too many drug trials, too few patients*

*y** = Health

$f(x, y =$ Health $) =$ [1, 1, 0, 0, 0, 0, 0, 0, 0]

$f(x, y =$ Sports $) =$ [0, 0, 0, 1, 1, 0, 0, 0, 0]

# Training

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \sum_y f_i(x_j, y)P_w(y|x_j)$$

*too many drug trials, too few patients*

$y^*$ = Health

$f(x, y =$ Health $) =$ [1, 1, 0, 0, 0, 0, 0, 0, 0]

$P_w(y|x)$ = [0.21, 0.77, 0.02]

$f(x, y =$ Sports $) =$ [0, 0, 0, 1, 1, 0, 0, 0, 0]

# Training

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \sum_y f_i(x_j, y) P_w(y|x_j)$$

*too many drug trials, too few patients*

$y^*$ = Health

$f(x, y =$ Health $) =$ [1, 1, 0, 0, 0, 0, 0, 0, 0]

$f(x, y =$ Sports $) =$ [0, 0, 0, 1, 1, 0, 0, 0, 0]

$P_w(y|x)$ = [0.21, 0.77, 0.02]

gradient:

# Training

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \sum_y f_i(x_j, y) P_w(y|x_j)$$

*too many drug trials, too few patients*

$y^*$ = Health

$f(x, y =$ Health $) =$ [1, 1, 0, 0, 0, 0, 0, 0, 0]

$P_w(y|x)$ = [0.21, 0.77, 0.02]

$f(x, y =$ Sports $) =$ [0, 0, 0, 1, 1, 0, 0, 0, 0]

gradient: [1, 1, 0, 0, 0, 0, 0, 0, 0]

# Training

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \sum_y f_i(x_j, y) P_w(y|x_j)$$

*too many drug trials, too few patients*

$y^*$ = Health

$f(x, y =$ Health $) =$ [1, 1, 0, 0, 0, 0, 0, 0, 0]

$P_w(y|x)$ = [0.21, 0.77, 0.02]

$f(x, y =$ Sports $) =$ [0, 0, 0, 1, 1, 0, 0, 0, 0]

gradient: [1, 1, 0, 0, 0, 0, 0, 0, 0] — 0.21 [1, 1, 0, 0, 0, 0, 0, 0, 0]

# Training

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \sum_y f_i(x_j, y) P_w(y|x_j)$$

*too many drug trials, too few patients*

$y^*$ = Health

$f(x, y = \text{Health}) = [1, 1, 0, 0, 0, 0, 0, 0, 0]$

$P_w(y|x) = [0.21, 0.77, 0.02]$

$f(x, y = \text{Sports}) = [0, 0, 0, 1, 1, 0, 0, 0, 0]$

gradient: [1, 1, 0, 0, 0, 0, 0, 0, 0] — 0.21 [1, 1, 0, 0, 0, 0, 0, 0, 0]

— 0.77 [0, 0, 0, 1, 1, 0, 0, 0, 0] — 0.02 [0, 0, 0, 0, 0, 0, 1, 1, 0]

# Training

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \sum_y f_i(x_j, y) P_w(y|x_j)$$

*too many drug trials, too few patients*

$y^*$ = Health

$f(x, y =$ Health $) =$ [1, 1, 0, 0, 0, 0, 0, 0, 0]

$P_w(y|x)$ = [0.21, 0.77, 0.02]

$f(x, y =$ Sports $) =$ [0, 0, 0, 1, 1, 0, 0, 0, 0]

gradient: [1, 1, 0, 0, 0, 0, 0, 0, 0] — 0.21 [1, 1, 0, 0, 0, 0, 0, 0, 0]
— 0.77 [0, 0, 0, 1, 1, 0, 0, 0, 0] — 0.02 [0, 0, 0, 0, 0, 0, 1, 1, 0]

= [0.79, 0.79, 0, -0.77, -0.77, 0, -0.02, -0.02, 0]

# Training

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \sum_y f_i(x_j, y) P_w(y|x_j)$$

*too many drug trials, too few patients*

$y^*$ = Health

$f(x, y = \text{Health}) = [1, 1, 0, 0, 0, 0, 0, 0, 0]$

$P_w(y|x) = [0.21, 0.77, 0.02]$

$f(x, y = \text{Sports}) = [0, 0, 0, 1, 1, 0, 0, 0, 0]$

gradient: [1, 1, 0, 0, 0, 0, 0, 0, 0] — 0.21 [1, 1, 0, 0, 0, 0, 0, 0, 0]
— 0.77 [0, 0, 0, 1, 1, 0, 0, 0, 0] — 0.02 [0, 0, 0, 0, 0, 0, 1, 1, 0]

= [0.79, 0.79, 0, -0.77, -0.77, 0, -0.02, -0.02, 0]

update $w^\top$:

# Training

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \sum_y f_i(x_j, y) P_w(y|x_j)$$

*too many drug trials, too few patients*

$y^*$ = Health

$f(x, y =$ Health $) =$ [1, 1, 0, 0, 0, 0, 0, 0, 0]

$f(x, y =$ Sports $) =$ [0, 0, 0, 1, 1, 0, 0, 0, 0]

$P_w(y|x)$ = [0.21, 0.77, 0.02]

gradient: [1, 1, 0, 0, 0, 0, 0, 0, 0] — 0.21 [1, 1, 0, 0, 0, 0, 0, 0, 0]
— 0.77 [0, 0, 0, 1, 1, 0, 0, 0, 0] — 0.02 [0, 0, 0, 0, 0, 0, 1, 1, 0]

= [0.79, 0.79, 0, -0.77, -0.77, 0, -0.02, -0.02, 0]

update $w^\top$:
[1.3, 0.9, -5, 3.2, -0.1, 0, 1.1, -1.7, -1.3]

# Training

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \sum_y f_i(x_j, y) P_w(y|x_j)$$

*too many drug trials, too few patients*

$y^*$ = Health

$f(x, y =$ Health $) =$ [1, 1, 0, 0, 0, 0, 0, 0, 0]

$P_w(y|x)$ = [0.21, 0.77, 0.02]

$f(x, y =$ Sports $) =$ [0, 0, 0, 1, 1, 0, 0, 0, 0]

gradient: [1, 1, 0, 0, 0, 0, 0, 0, 0] — 0.21 [1, 1, 0, 0, 0, 0, 0, 0, 0]
— 0.77 [0, 0, 0, 1, 1, 0, 0, 0, 0] — 0.02 [0, 0, 0, 0, 0, 0, 1, 1, 0]

= [0.79, 0.79, 0, -0.77, -0.77, 0, -0.02, -0.02, 0]

update $w^\top$:

[1.3, 0.9, -5, 3.2, -0.1, 0, 1.1, -1.7, -1.3] + [0.79, 0.79, 0, -0.77, -0.77, 0, -0.02, -0.02, 0]

# Training

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \sum_y f_i(x_j, y) P_w(y|x_j)$$

*too many drug trials, too few patients*

$y^*$ = Health

$f(x, y = \text{Health}) = [1, 1, 0, 0, 0, 0, 0, 0, 0]$

$P_w(y|x) = [0.21, 0.77, 0.02]$

$f(x, y = \text{Sports}) = [0, 0, 0, 1, 1, 0, 0, 0, 0]$

gradient: [1, 1, 0, 0, 0, 0, 0, 0, 0] — 0.21 [1, 1, 0, 0, 0, 0, 0, 0, 0]
— 0.77 [0, 0, 0, 1, 1, 0, 0, 0, 0] — 0.02 [0, 0, 0, 0, 0, 0, 1, 1, 0]

= [0.79, 0.79, 0, -0.77, -0.77, 0, -0.02, -0.02, 0]

update $w^\top$:
[1.3, 0.9, -5, 3.2, -0.1, 0, 1.1, -1.7, -1.3] + [0.79, 0.79, 0, -0.77, -0.77, 0, -0.02, -0.02, 0]
= [2.09, 1.69, 0, 2.43, -0.87, 0, 1.08, -1.72, 0]

# Training

$$\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \sum_y f_i(x_j, y) P_w(y|x_j)$$

*too many drug trials, too few patients*

$y^*$ = Health

$f(x, y =$ Health $) =$ [1, 1, 0, 0, 0, 0, 0, 0, 0]

$P_w(y|x)$ = [0.21, 0.77, 0.02]

$f(x, y =$ Sports $) =$ [0, 0, 0, 1, 1, 0, 0, 0, 0]

gradient: [1, 1, 0, 0, 0, 0, 0, 0, 0] — 0.21 [1, 1, 0, 0, 0, 0, 0, 0, 0]
— 0.77 [0, 0, 0, 1, 1, 0, 0, 0, 0] — 0.02 [0, 0, 0, 0, 0, 0, 1, 1, 0]

= [0.79, 0.79, 0, -0.77, -0.77, 0, -0.02, -0.02, 0]

update $w^\top$:
[1.3, 0.9, -5, 3.2, -0.1, 0, 1.1, -1.7, -1.3] + [0.79, 0.79, 0, -0.77, -0.77, 0, -0.02, -0.02, 0]
= [2.09, 1.69, 0, 2.43, -0.87, 0, 1.08, -1.72, 0]

↘ new $P_w(y|x)$ = [0.89, 0.10, 0.01]

# Logistic Regression: Summary

- Model: $P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$

# Logistic Regression: Summary

- Model: $P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$

- Inference: $\mathrm{argmax}_y P_w(y|x)$

# Logistic Regression: Summary

- Model: $P_w(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$

- Inference: $\mathrm{argmax}_y P_w(y|x)$

- Learning: gradient ascent on the discriminative log-likelihood

$$f(x,y^*) - \mathbb{E}_y[f(x,y)] = f(x,y^*) - \sum_y [P_w(y|x) f(x,y)]$$

"towards gold feature value, away from expectation of feature value"

# Multiclass SVM

# Soft Margin SVM

# Soft Margin SVM

Minimize $\lambda \|w\|_2^2 + \sum_{j=1}^{m} \xi_j$ ← slack variables > 0 iff example is support vector

# Soft Margin SVM

Minimize $\lambda \|w\|_2^2 + \sum_{j=1}^{m} \xi_j$

slack variables > 0 iff example is support vector

s.t. $\forall j \quad \xi_j \geq 0$

# Soft Margin SVM

Minimize $\lambda\|w\|_2^2 + \sum_{j=1}^{m} \xi_j$ ← slack variables > 0 iff example is support vector

s.t. $\forall j \quad \xi_j \geq 0$

$\forall j \quad (2y_j - 1)(w^\top x_j) \geq 1 - \xi_j$

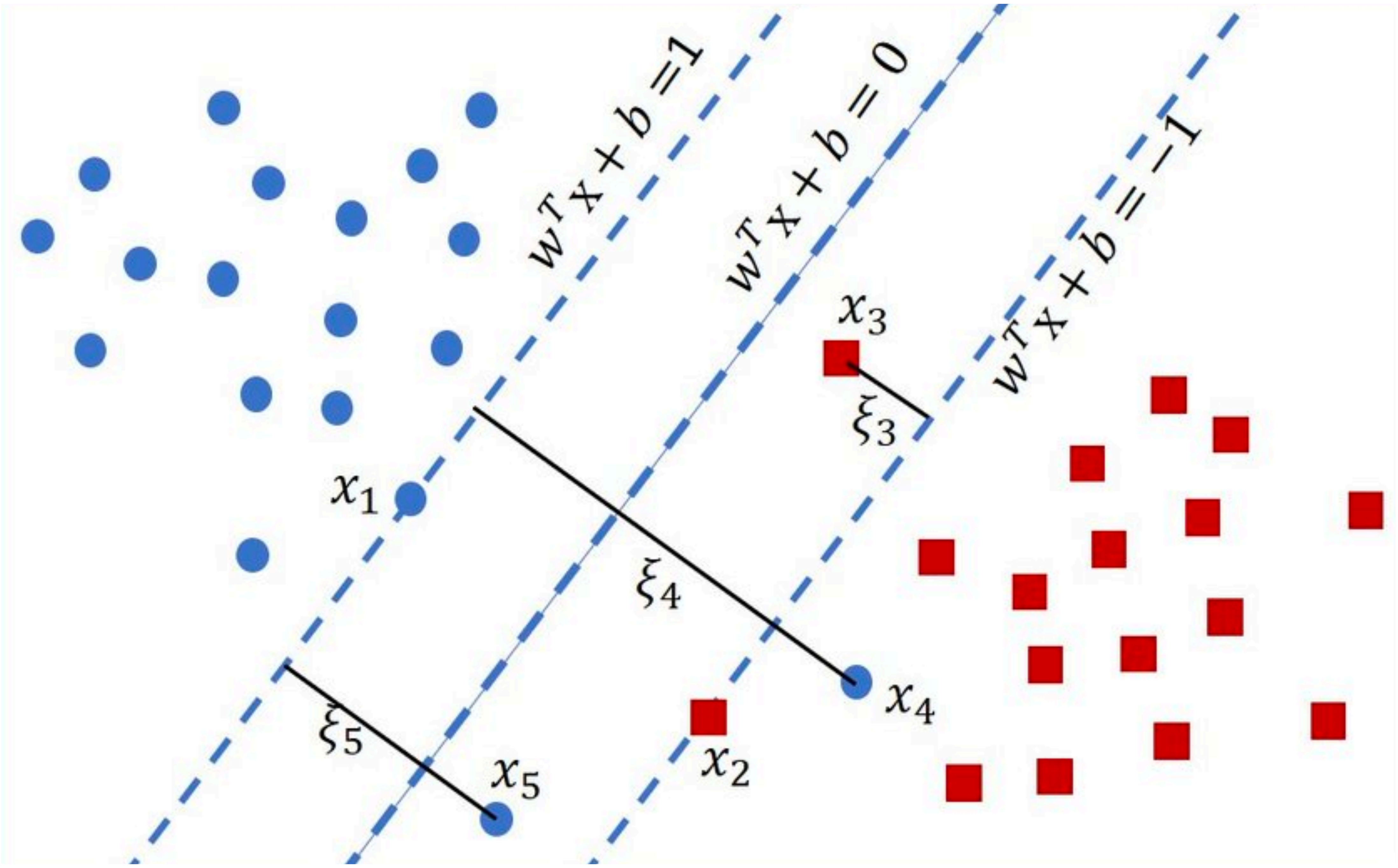



Image credit: Lang Van Tran

# Multiclass SVM

Minimize $\lambda \|w\|_2^2 + \sum_{j=1}^{m} \xi_j$ ← slack variables > 0 iff example is support vector

s.t. $\forall j \quad \xi_j \geq 0$

$\forall j \quad (2y_j - 1)(w^\top x_j) \geq 1 - \xi_j$

# Multiclass SVM

Minimize $\lambda\|w\|_2^2 + \sum_{j=1}^{m} \xi_j$

slack variables > 0 iff example is support vector

s.t. $\forall j \quad \xi_j \geq 0$

~~$\forall j \quad (2y_j - 1)(w^\top x_j) \geq 1 - \xi_j$~~

# Multiclass SVM

Minimize $\lambda\|w\|_2^2 + \sum_{j=1}^{m} \xi_j$ ← slack variables > 0 iff example is support vector

s.t. $\forall j \quad \xi_j \geq 0$

~~$\forall j \quad (2y_j - 1)(w^\top x_j) \geq 1 - \xi_j$~~

$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$

# Multiclass SVM

Minimize $\lambda\|w\|_2^2 + \sum_{j=1}^{m} \xi_j$ ← slack variables > 0 iff example is support vector

s.t. $\forall j \quad \xi_j \geq 0$

~~$\forall j \quad (2y_j - 1)(w^\top x_j) \geq 1 - \xi_j$~~

$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$

Correct prediction now has to beat every other class

# Multiclass SVM

Minimize $\lambda \|w\|_2^2 + \sum_{j=1}^{m} \xi_j$

slack variables > 0 iff example is support vector

s.t. $\forall j \quad \xi_j \geq 0$

~~$\forall j \quad (2y_j - 1)(w^\top x_j) \geq 1 - \xi_j$~~

$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$

Correct prediction now has to beat every other class

Score comparison is more explicit now

# Multiclass SVM

Minimize $\lambda\|w\|_2^2 + \sum_{j=1}^{m} \xi_j$ ← slack variables > 0 iff example is support vector

s.t. $\forall j \quad \xi_j \geq 0$

~~$\forall j \quad (2y_j - 1)(w^\top x_j) \geq 1 - \xi_j$~~

$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$

Correct prediction now has to beat every other class

Score comparison is more explicit now

The 1 that was here is replaced by a loss function

# Training (loss-augmented)

# Training (loss-augmented)

- Are all decisions equally costly?

# Training (loss-augmented)

- Are all decisions equally costly?

*too many drug trials, too few patients* → Health / Sports / Science

# Training (loss-augmented)

- Are all decisions equally costly?

*too many drug trials, too few patients* → Health / Sports / Science

Predicted Sports: bad error

# Training (loss-augmented)

- Are all decisions equally costly?

*too many drug trials, too few patients* → Health / Sports / Science

Predicted Sports: bad error

Predicted Science: not so bad

# Training (loss-augmented)

- Are all decisions equally costly?

*too many drug trials, too few patients* → Health / Sports / Science

Predicted Sports: bad error

Predicted Science: not so bad

- We can define a loss function $\ell(y, y^*)$

# Training (loss-augmented)

- Are all decisions equally costly?

*too many drug trials, too few patients* → Health / Sports / Science

Predicted Sports: bad error

Predicted Science: not so bad

- We can define a loss function $\ell(y, y^*)$

$$\ell(\text{Sports}, \text{Health}) = 3$$

# Training (loss-augmented)

- Are all decisions equally costly?

*too many drug trials, too few patients* → Health / Sports / Science

Predicted Sports: bad error

Predicted Science: not so bad

- We can define a loss function $\ell(y, y^*)$

$$\ell(\text{Sports}, \text{Health}) = 3$$

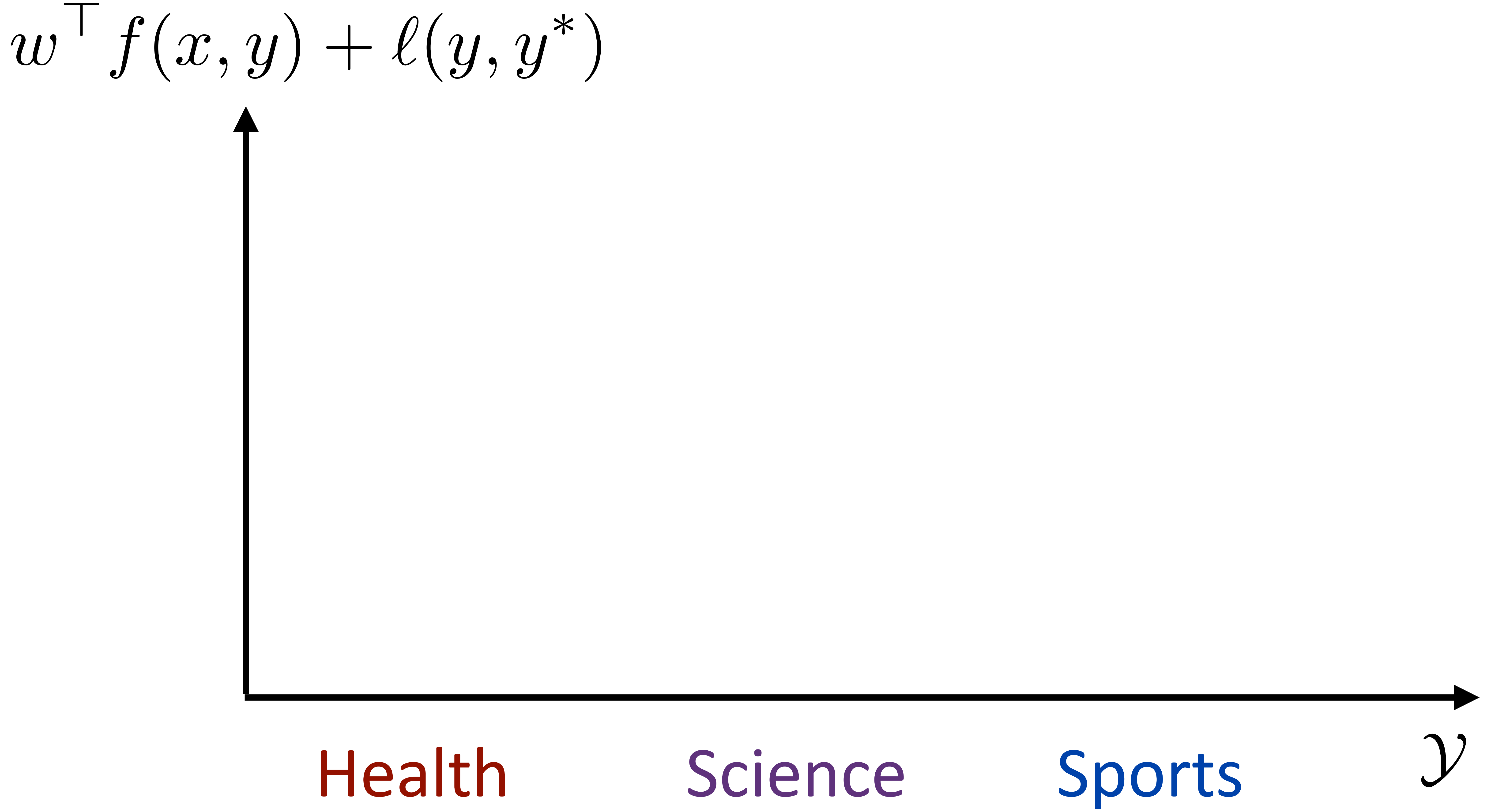
$$\ell(\text{Science}, \text{Health}) = 1$$

# Multiclass SVM

$$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$$

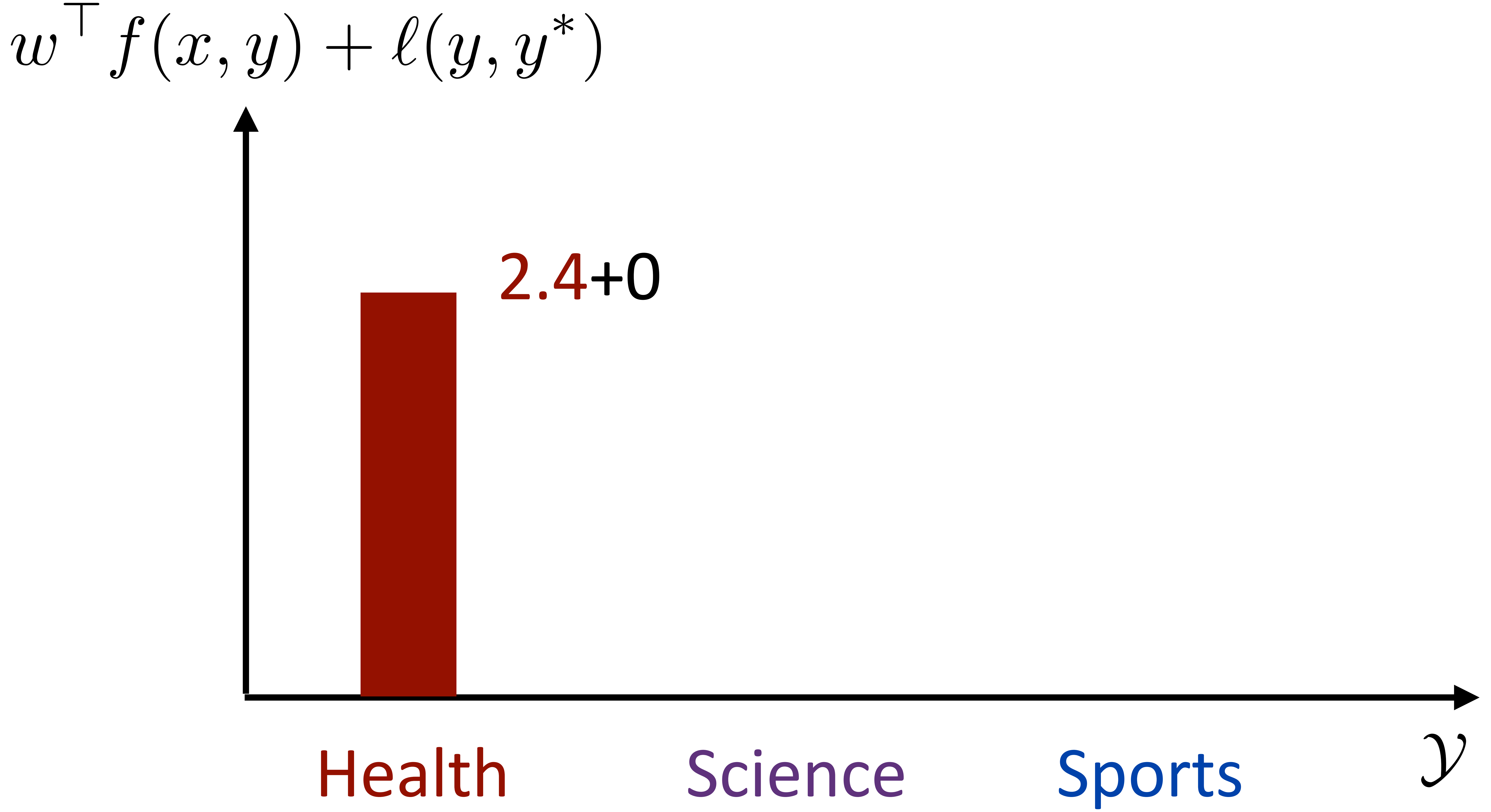
# Multiclass SVM

$$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$$



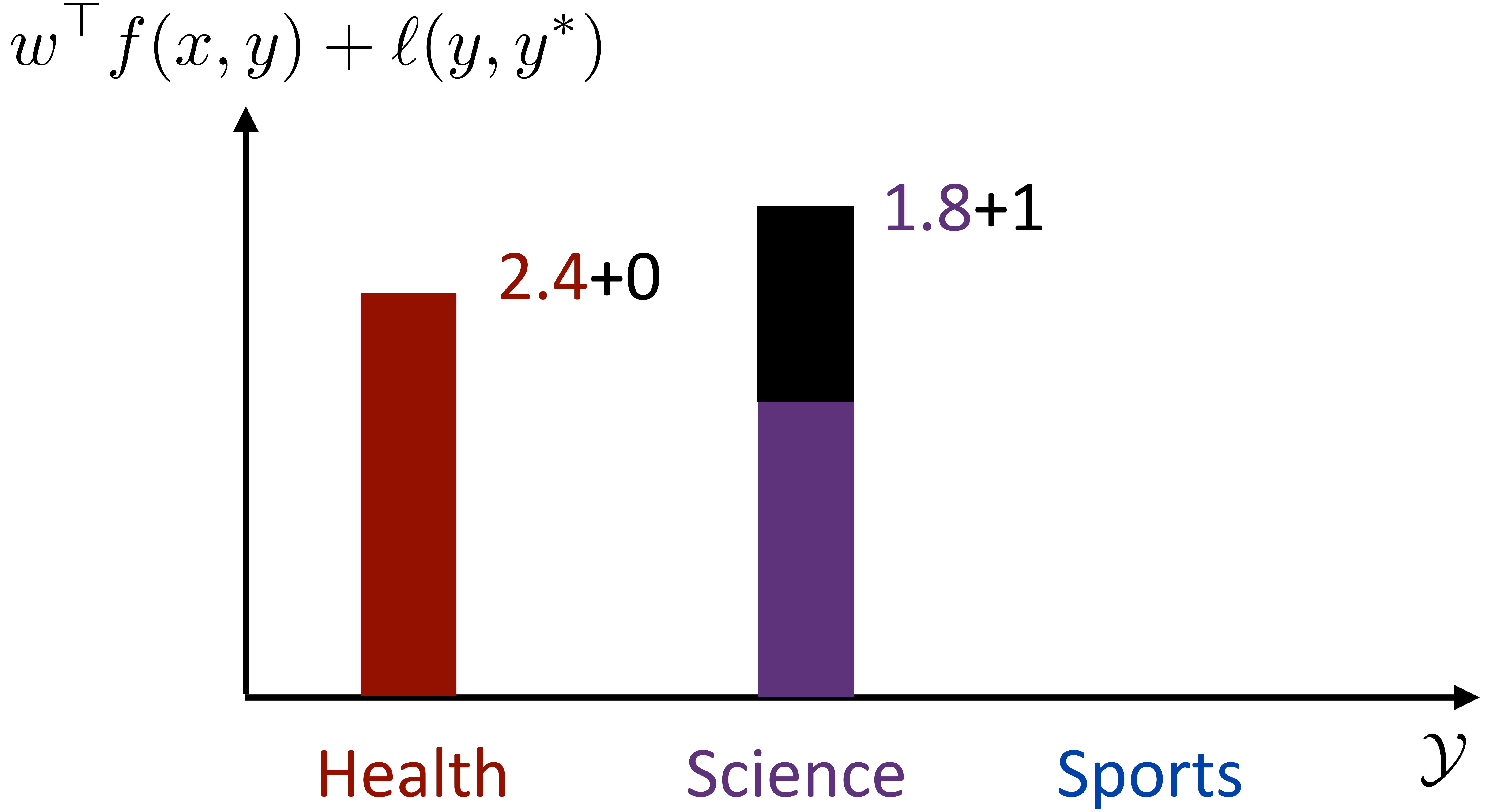
# Multiclass SVM

$$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$$



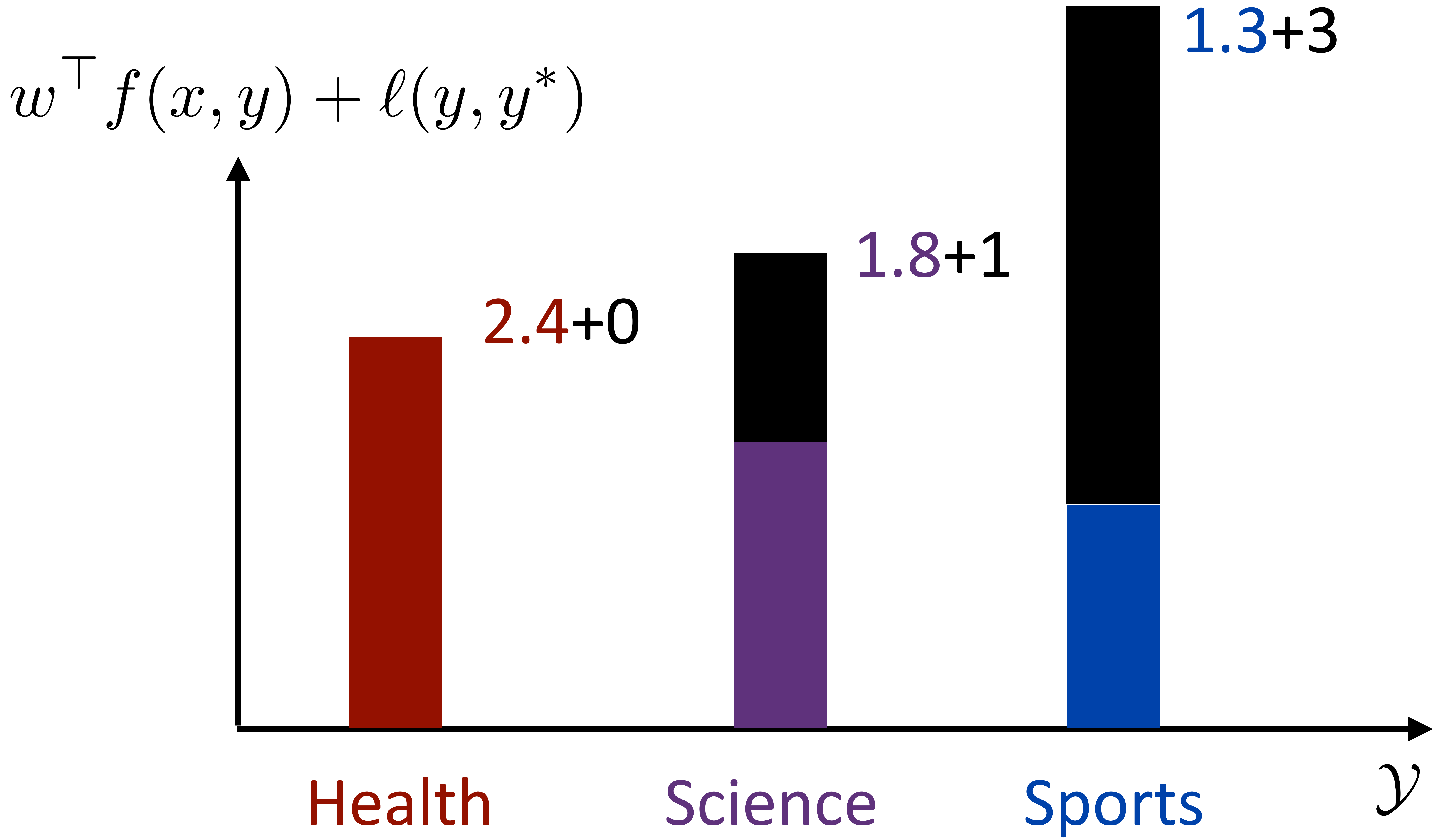
# Multiclass SVM

$$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$$



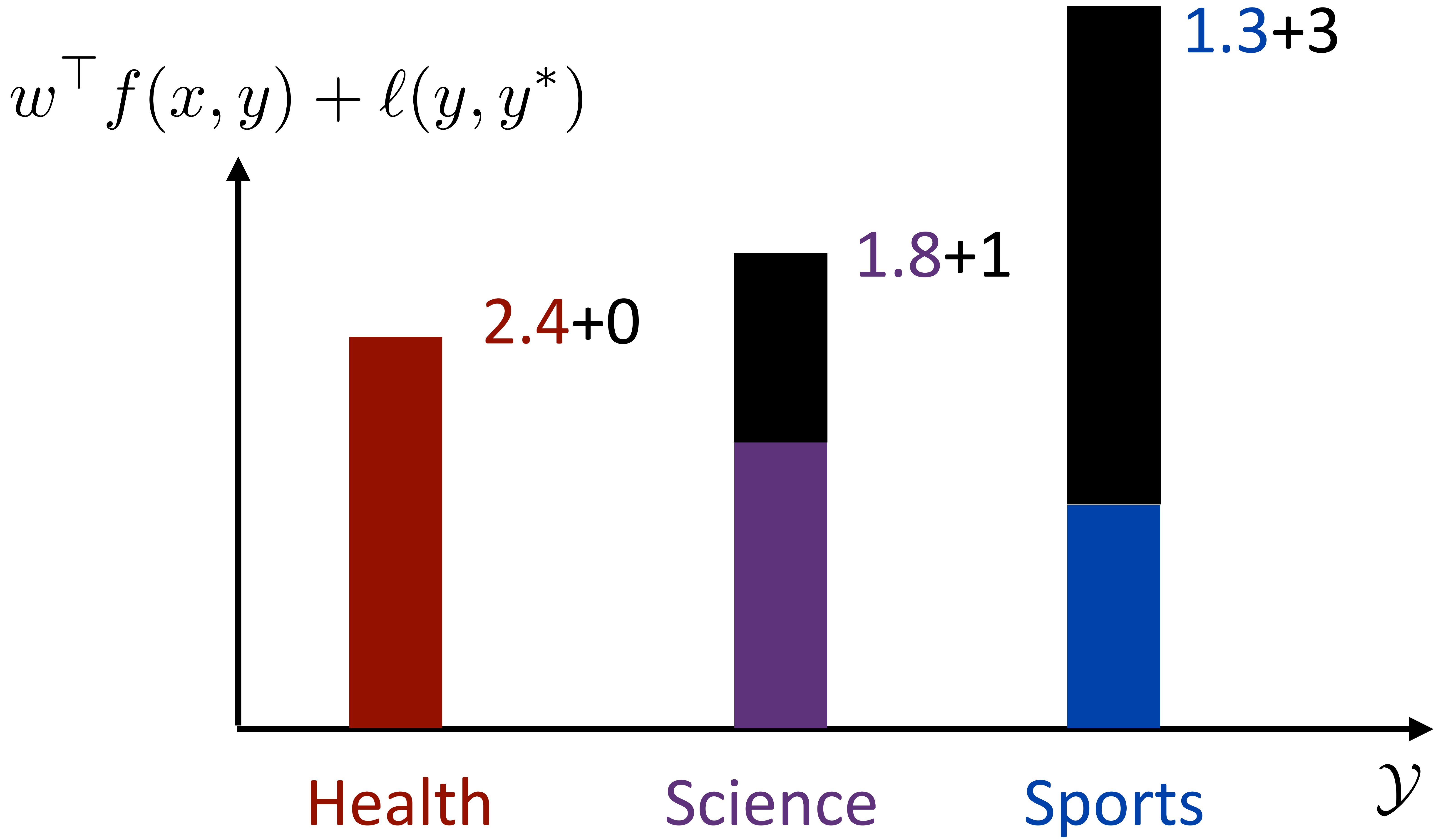
# Multiclass SVM

$$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$$



# Multiclass SVM

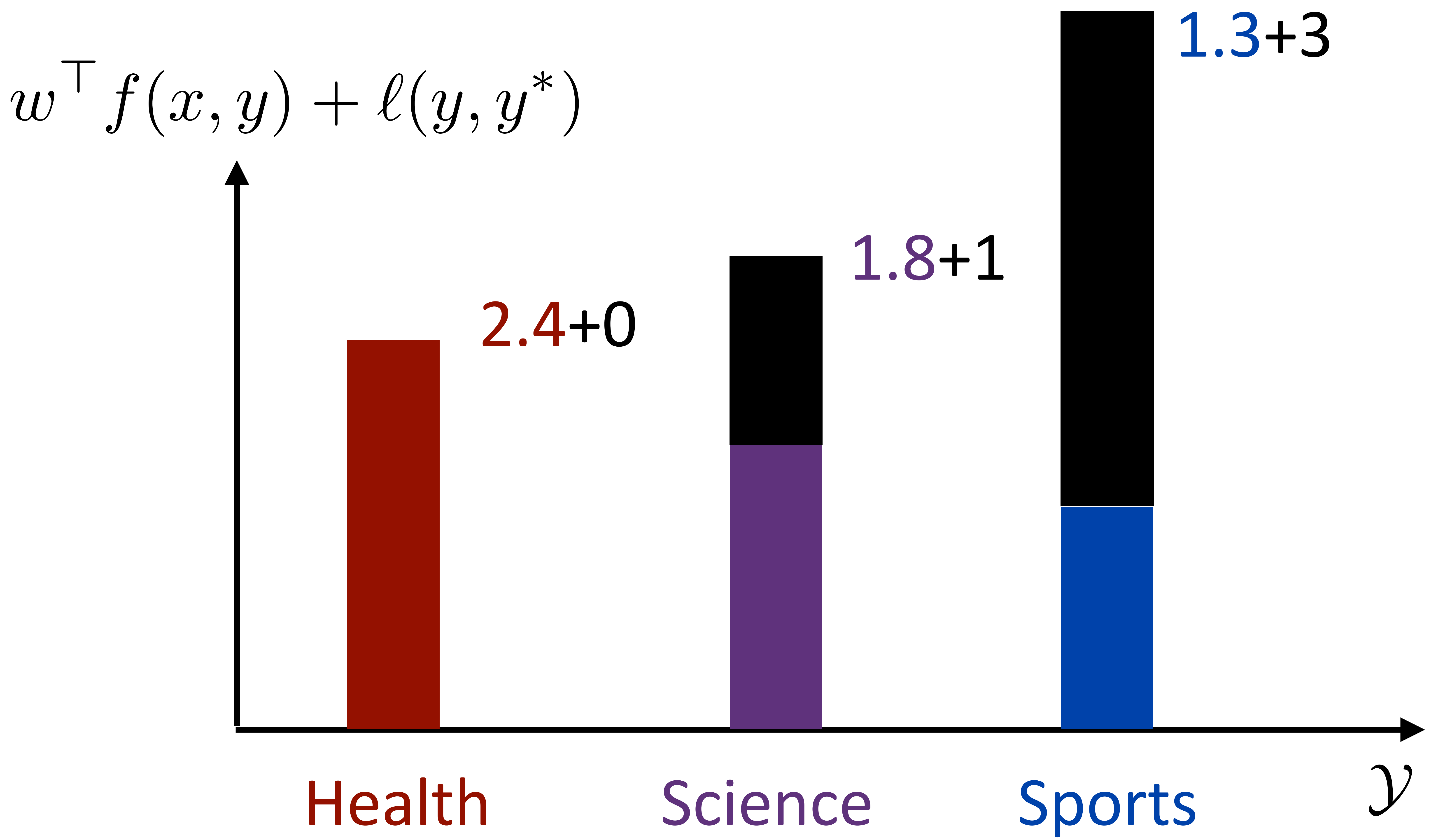
$$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$$




- Does gold beat every label + loss? No!

# Multiclass SVM

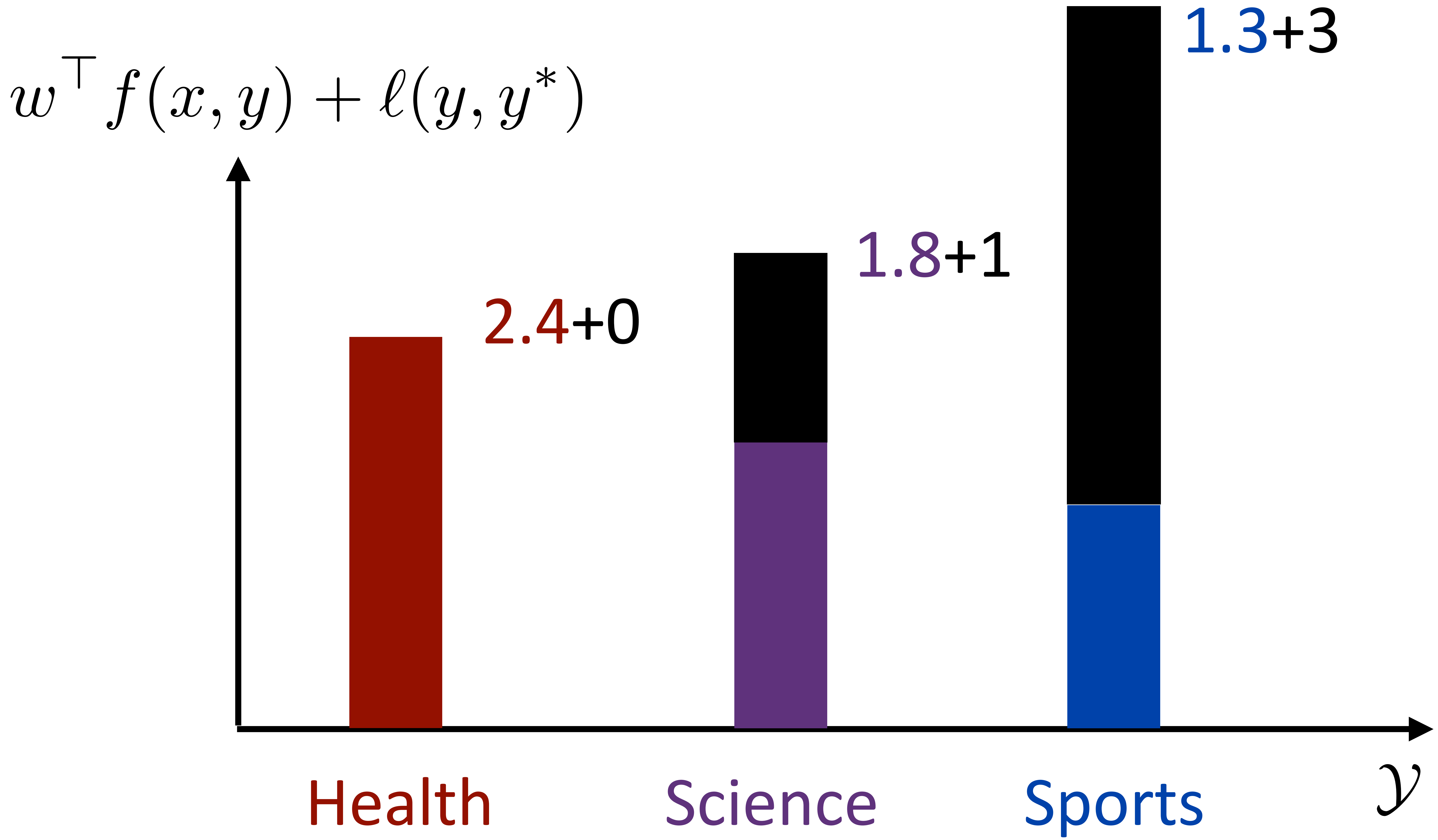
$$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$$




- Does gold beat every label + loss? No!
- Most violated constraint is Sports; what is $\xi_j$ ?

# Multiclass SVM

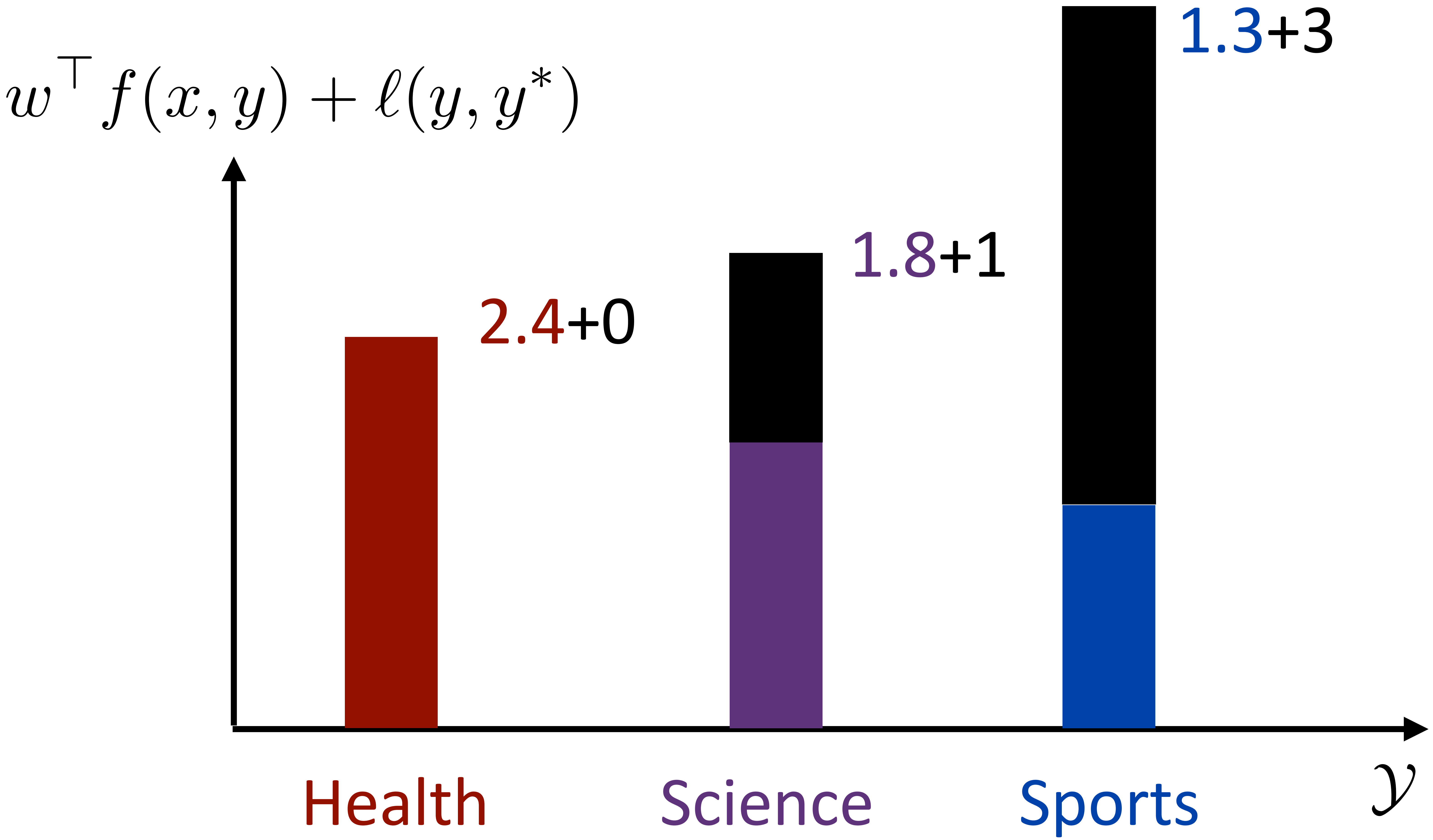
$$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$$




- Does gold beat every label + loss? No!
- Most violated constraint is Sports; what is $\xi_j$ ?
- $\xi_j$ = 4.3 - 2.4 = 1.9

# Multiclass SVM

$$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$$




- Does gold beat every label + loss? No!
- Most violated constraint is Sports; what is $\xi_j$ ?
- $\xi_j$ = 4.3 - 2.4 = 1.9
- Perceptron would make no update here

# Multiclass SVM

Minimize $\lambda\|w\|_2^2 + \sum_{j=1}^{m} \xi_j$

s.t. $\forall j \quad \xi_j \geq 0$

$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$

# Multiclass SVM

Minimize $\lambda\|w\|_2^2 + \sum_{j=1}^{m} \xi_j$

s.t. $\forall j \quad \xi_j \geq 0$

$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$

- One slack variable per example, so it's set to be whatever the *most violated constraint* is for that example

# Multiclass SVM

Minimize $\lambda\|w\|_2^2 + \sum_{j=1}^{m} \xi_j$

s.t. $\forall j \quad \xi_j \geq 0$

$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$

- One slack variable per example, so it's set to be whatever the *most violated constraint* is for that example

$$\xi_j = \max_{y \in \mathcal{Y}} w^\top f(x_j, y) + \ell(y, y_j^*) - w^\top f(x_j, y_j^*)$$

# Multiclass SVM

Minimize $\lambda \|w\|_2^2 + \sum_{j=1}^{m} \xi_j$

s.t. $\forall j \quad \xi_j \geq 0$

$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$

- One slack variable per example, so it's set to be whatever the *most violated constraint* is for that example

$$\xi_j = \max_{y \in \mathcal{Y}} \boxed{w^\top f(x_j, y) + \ell(y, y_j^*)} - w^\top f(x_j, y_j^*)$$

# Multiclass SVM

Minimize $\lambda \|w\|_2^2 + \sum_{j=1}^{m} \xi_j$

s.t. $\forall j \quad \xi_j \geq 0$

$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$

- One slack variable per example, so it's set to be whatever the *most violated constraint* is for that example

$$\xi_j = \max_{y \in \mathcal{Y}} w^\top f(x_j, y) + \ell(y, y_j^*) - w^\top f(x_j, y_j^*)$$

- Plug in the gold *y* and you get 0, so slack is always nonnegative!

# Computing the Subgradient

Minimize $\lambda \|w\|_2^2 + \sum_{j=1}^{m} \xi_j$

s.t. $\forall j \quad \xi_j \geq 0$

$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$

# Computing the Subgradient

Minimize $\lambda\|w\|_2^2 + \sum_{j=1}^{m} \xi_j$

s.t. $\forall j \quad \xi_j \geq 0$

$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$

- If $\xi_j = 0$, the example is not a support vector, gradient is zero

# Computing the Subgradient

Minimize $\lambda\|w\|_2^2 + \sum_{j=1}^{m} \xi_j$

s.t. $\forall j \quad \xi_j \geq 0$

$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$

- If $\xi_j = 0$, the example is not a support vector, gradient is zero
- Otherwise, $\xi_j = \max_{y \in \mathcal{Y}} w^\top f(x_j, y) + \ell(y, y_j^*) - w^\top f(x_j, y_j^*)$

# Computing the Subgradient

Minimize $\lambda\|w\|_2^2 + \sum_{j=1}^{m} \xi_j$

s.t. $\forall j \quad \xi_j \geq 0$

$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$

- If $\xi_j = 0$, the example is not a support vector, gradient is zero
- Otherwise, $\xi_j = \max_{y \in \mathcal{Y}} w^\top f(x_j, y) + \ell(y, y_j^*) - w^\top f(x_j, y_j^*)$

$$\frac{\partial}{\partial w_i} \xi_j = f_i(x_j, y_{\max}) - f_i(x_j, y_j^*)$$

# Computing the Subgradient

Minimize $\lambda\|w\|_2^2 + \sum_{j=1}^{m} \xi_j$

s.t. $\forall j \quad \xi_j \geq 0$

$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$

- If $\xi_j = 0$, the example is not a support vector, gradient is zero
- Otherwise, $\xi_j = \max_{y \in \mathcal{Y}} w^\top f(x_j, y) + \ell(y, y_j^*) - w^\top f(x_j, y_j^*)$

$$\frac{\partial}{\partial w_i} \xi_j = f_i(x_j, y_{\max}) - f_i(x_j, y_j^*)$$

← (update looks backwards — we're minimizing here!)

# Computing the Subgradient

Minimize $\lambda\|w\|_2^2 + \sum_{j=1}^{m} \xi_j$

s.t. $\forall j \quad \xi_j \geq 0$

$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$

- If $\xi_j = 0$, the example is not a support vector, gradient is zero
- Otherwise, $\xi_j = \max_{y \in \mathcal{Y}} w^\top f(x_j, y) + \ell(y, y_j^*) - w^\top f(x_j, y_j^*)$

$$\frac{\partial}{\partial w_i} \xi_j = f_i(x_j, y_{\max}) - f_i(x_j, y_j^*)$$

← (update looks backwards — we're minimizing here!)

- Perceptron-like, but we update away from *loss-augmented* prediction

# Putting it Together

Minimize $\lambda \|w\|_2^2 + \sum_{j=1}^{m} \xi_j$

s.t. $\forall j \quad \xi_j \geq 0$

$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$

# Putting it Together

Minimize $\lambda \|w\|_2^2 + \sum_{j=1}^{m} \xi_j$

s.t. $\forall j \quad \xi_j \geq 0$

$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$

- (Unregularized) gradients:

# Putting it Together

Minimize $\lambda\|w\|_2^2 + \sum_{j=1}^{m} \xi_j$

s.t. $\forall j \quad \xi_j \geq 0$

$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$

- (Unregularized) gradients:
  - SVM: $f(x, y^*) - f(x, y_{\max})$ (loss-augmented max)

# Putting it Together

Minimize $\lambda\|w\|_2^2 + \sum_{j=1}^{m} \xi_j$

s.t. $\forall j \quad \xi_j \geq 0$

$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$

- (Unregularized) gradients:
  - SVM: $f(x, y^*) - f(x, y_{\max})$ (loss-augmented max)
  - Log reg: $f(x, y^*) - \mathbb{E}_y[f(x, y)] = f(x, y^*) - \sum_y [P_w(y|x) f(x, y)]$

# Putting it Together

Minimize $\lambda\|w\|_2^2 + \sum_{j=1}^{m} \xi_j$

s.t. $\forall j \quad \xi_j \geq 0$

$\forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j$

- (Unregularized) gradients:
  - SVM: $f(x, y^*) - f(x, y_{\max})$ (loss-augmented max)
  - Log reg: $f(x, y^*) - \mathbb{E}_y[f(x, y)] = f(x, y^*) - \sum_y [P_w(y|x) f(x, y)]$
- SVM: max over $y$s to compute gradient. LR: need to sum over $y$s

# Optimization

# Recap

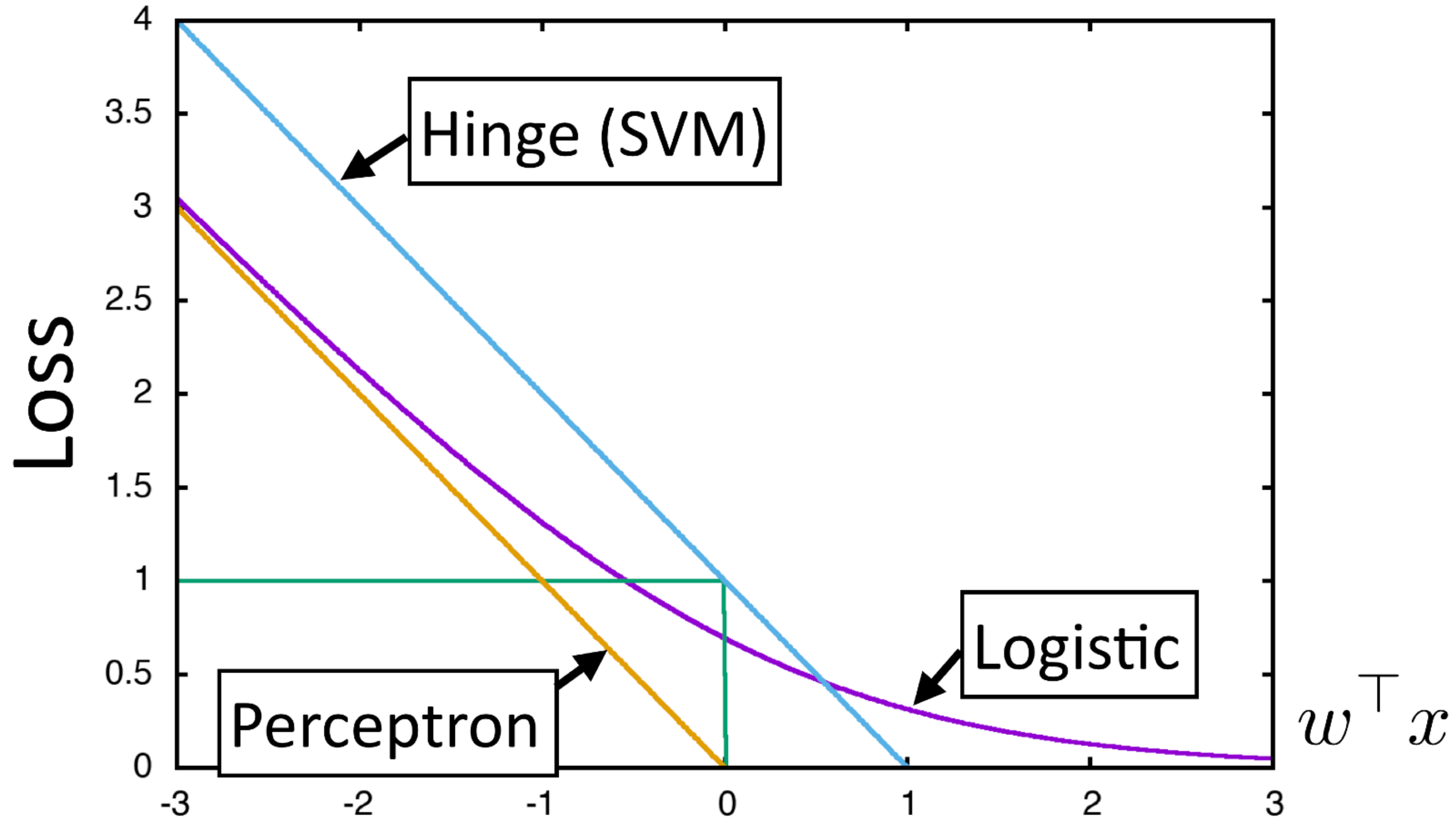
- Four elements of a machine learning method:

# Recap

- Four elements of a machine learning method:
  - Model: probabilistic, max-margin, deep neural network

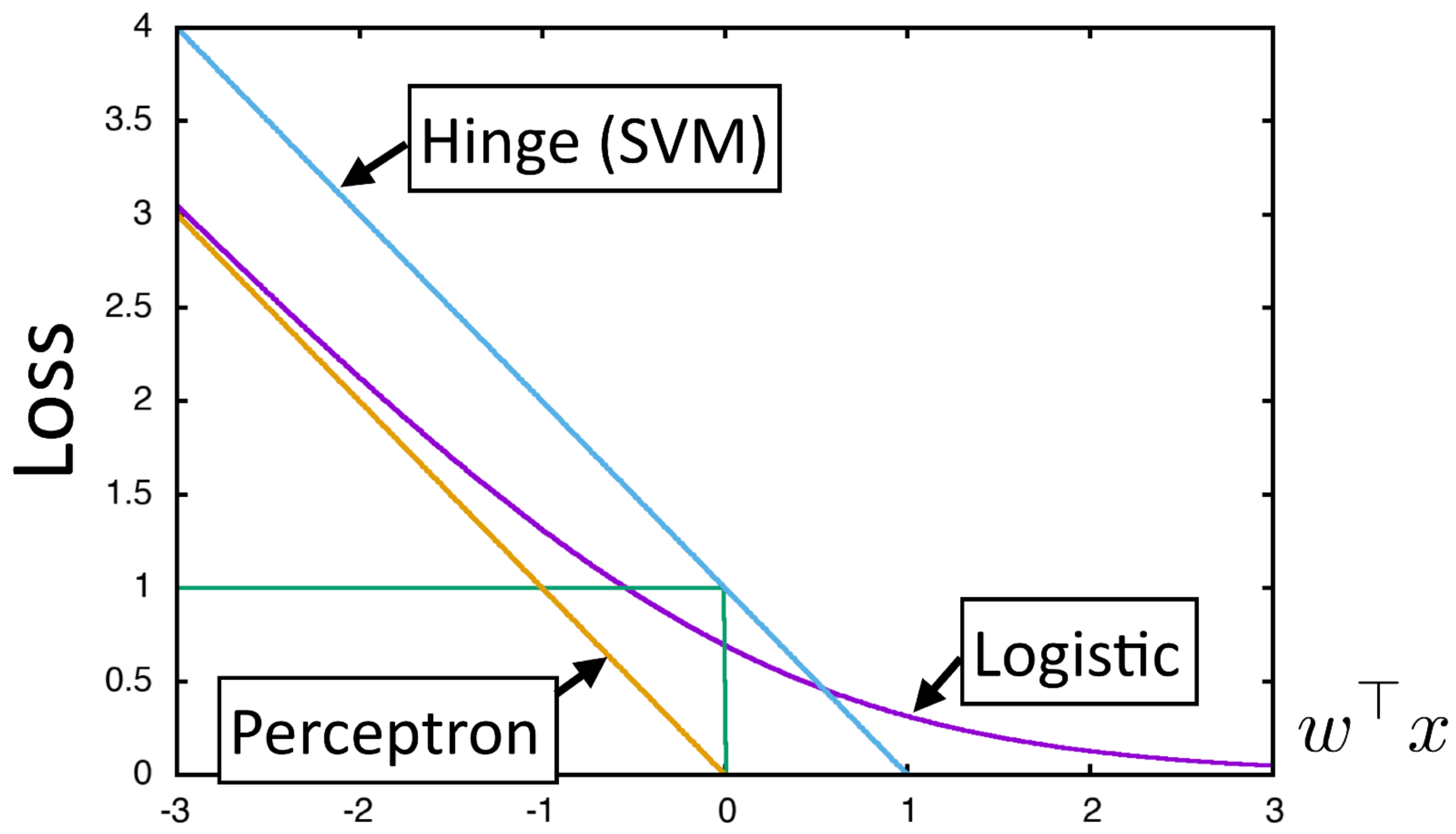
# Recap

- Four elements of a machine learning method:
  - Model: probabilistic, max-margin, deep neural network
  - Objective:



# Recap

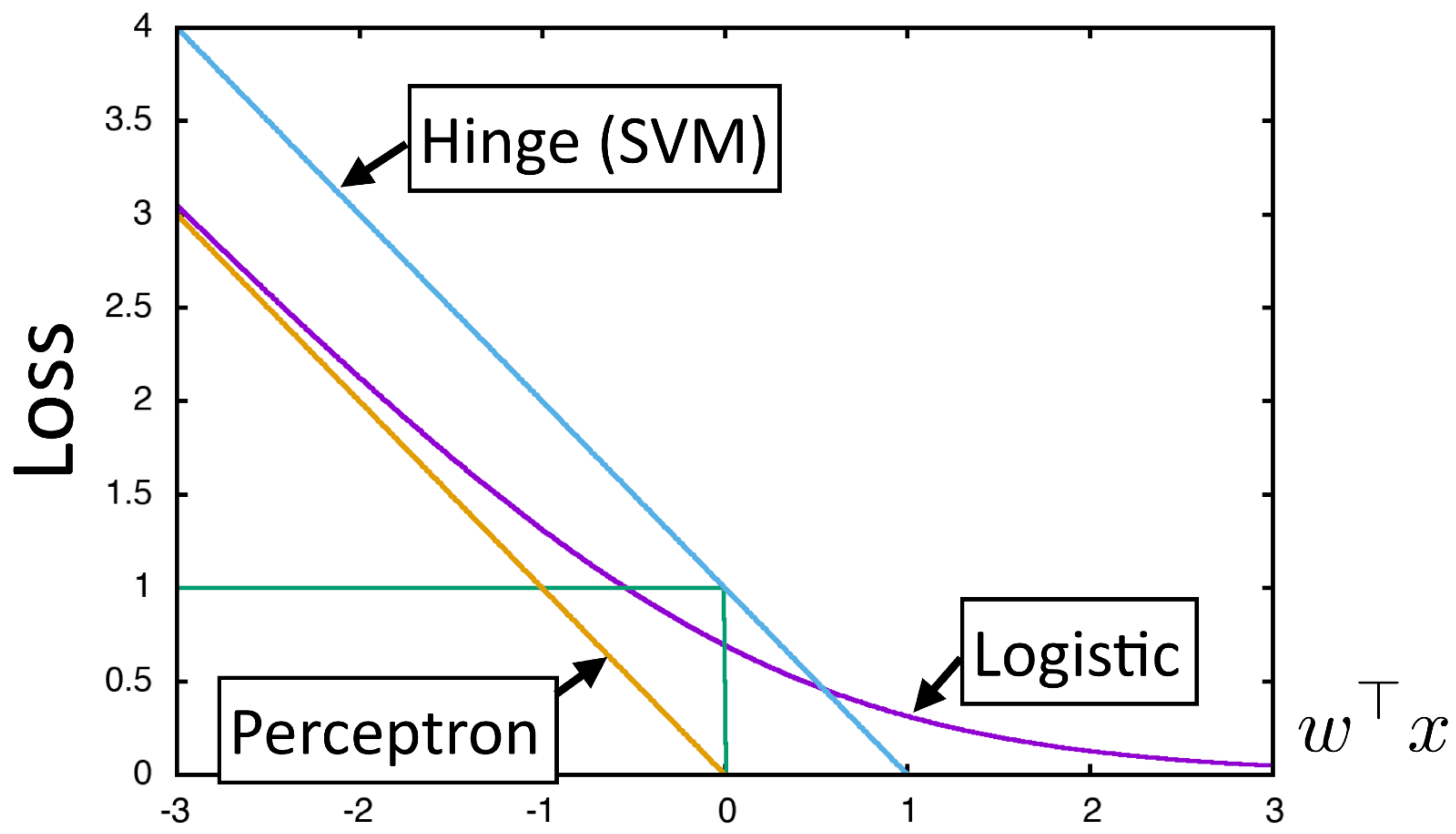
- Four elements of a machine learning method:
  - Model: probabilistic, max-margin, deep neural network
  - Objective:




  - Inference: just maxes and simple expectations so far, but will get harder

# Recap

- Four elements of a machine learning method:
  - Model: probabilistic, max-margin, deep neural network
  - Objective:




  - Inference: just maxes and simple expectations so far, but will get harder
  - Training: gradient descent?

# Optimization

# Optimization

- Stochastic gradient *ascent*

# Optimization

- Stochastic gradient *ascent*

$$w \leftarrow w + \alpha g, \quad g = \frac{\partial}{\partial w}\mathcal{L}$$

# Optimization

- Stochastic gradient *ascent*
  - Very simple to code up

$$w \leftarrow w + \alpha g, \quad g = \frac{\partial}{\partial w}\mathcal{L}$$

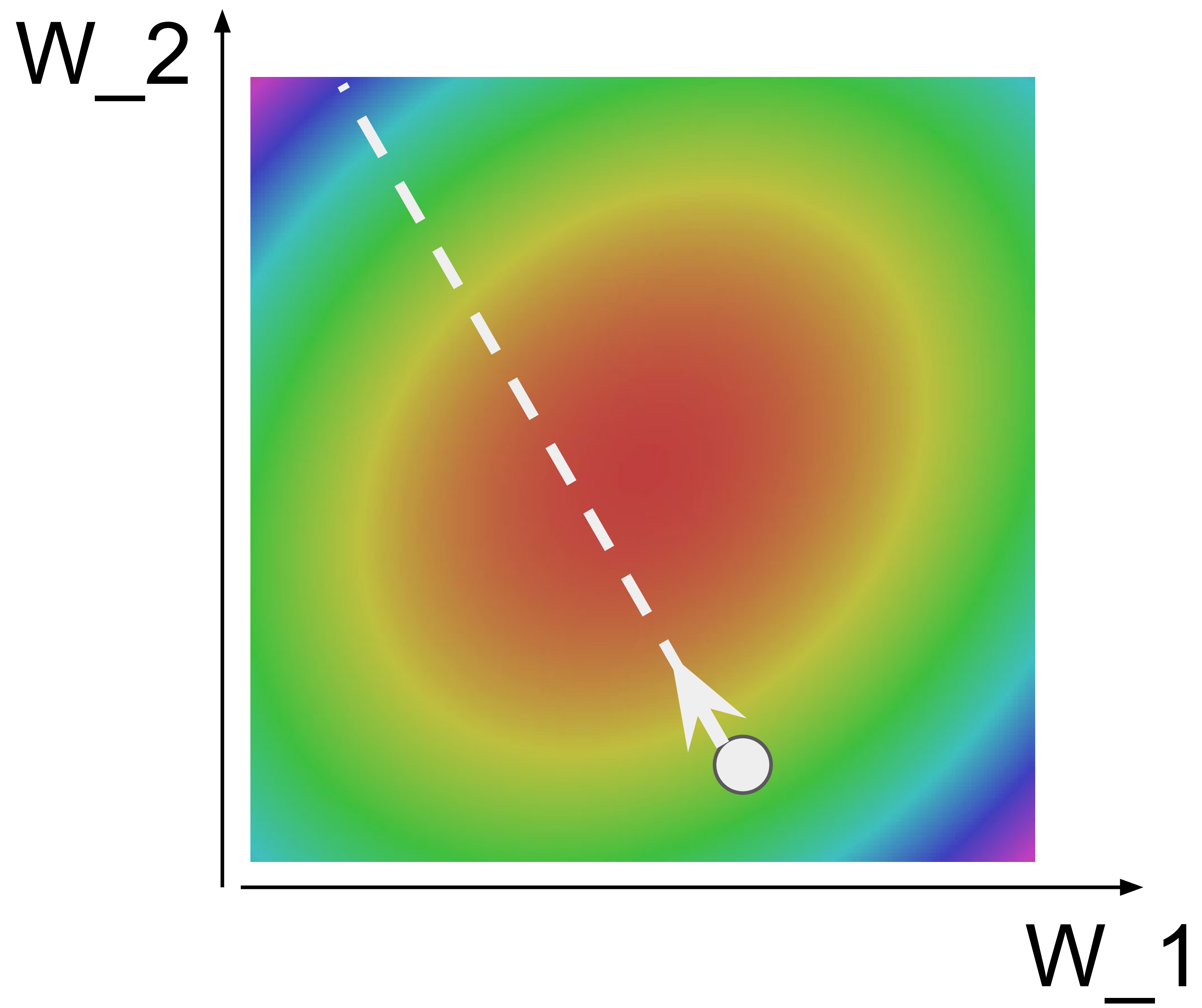
# Optimization

- Stochastic gradient *ascent*
  - Very simple to code up

$$w \leftarrow w + \alpha g, \quad g = \frac{\partial}{\partial w}\mathcal{L}$$

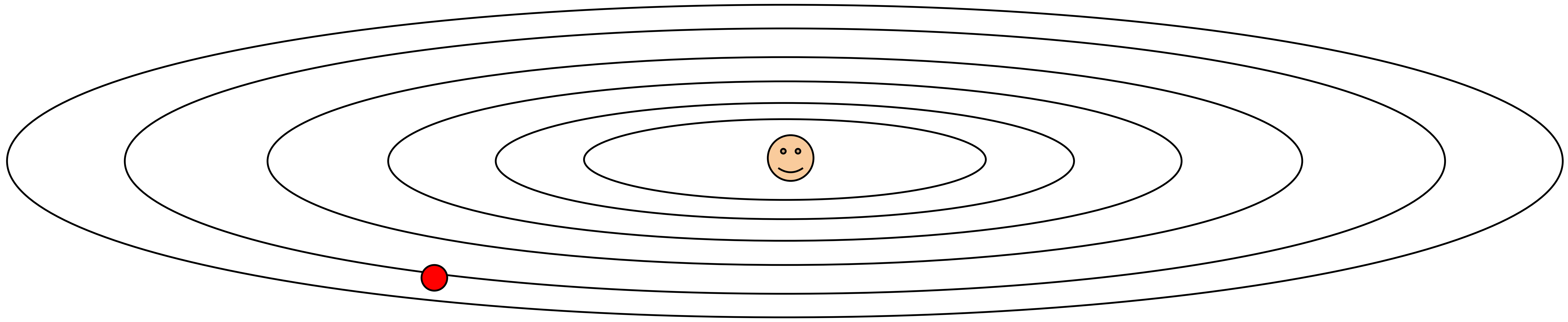
```
# Vanilla Gradient Descent

while True:
  weights_grad = evaluate_gradient(loss_fun, data, weights)
  weights += - step_size * weights_grad # perform parameter update
```



# Optimization

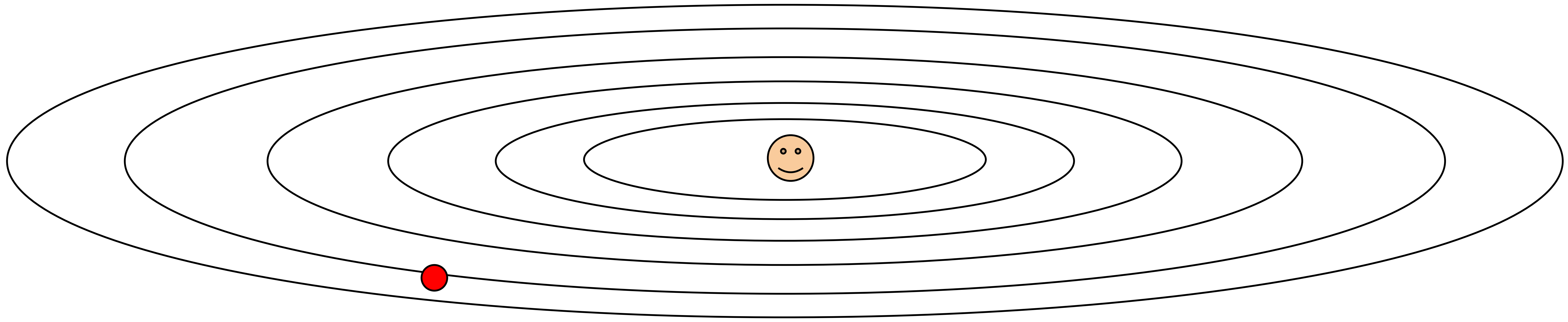
- Stochastic gradient *ascent*
  - Very simple to code up
  - What if loss changes quickly in one direction and slowly in another direction?



# Optimization

- Stochastic gradient *ascent*
  - Very simple to code up
  - What if loss changes quickly in one direction and slowly in another direction?

$$w \leftarrow w + \alpha g, \quad g = \frac{\partial}{\partial w}\mathcal{L}$$

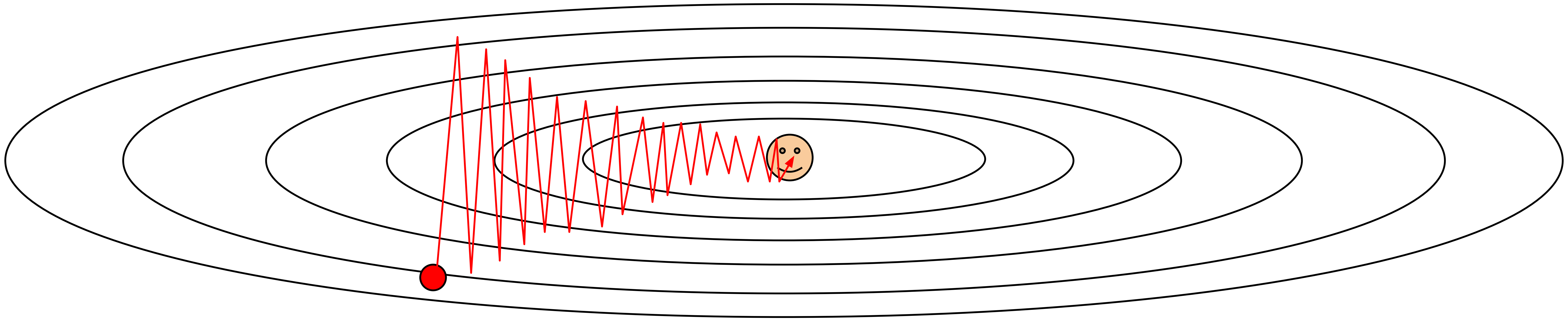


# Optimization

- Stochastic gradient *ascent*
  - Very simple to code up

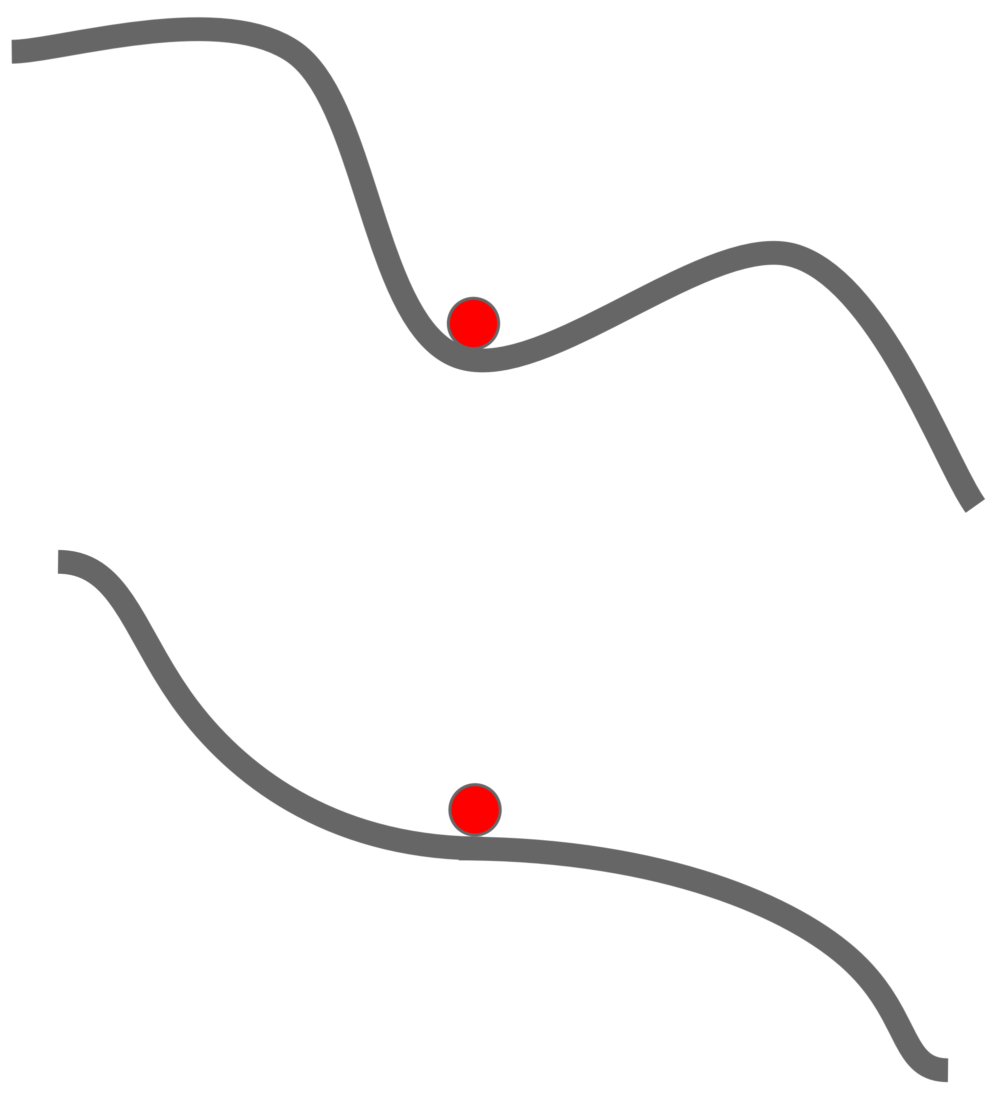
$$w \leftarrow w + \alpha g, \quad g = \frac{\partial}{\partial w}\mathcal{L}$$

  - What if loss changes quickly in one direction and slowly in another direction?



# Optimization

- Stochastic gradient *ascent*
  - Very simple to code up
  - What if the loss function has a local minima or saddle point?

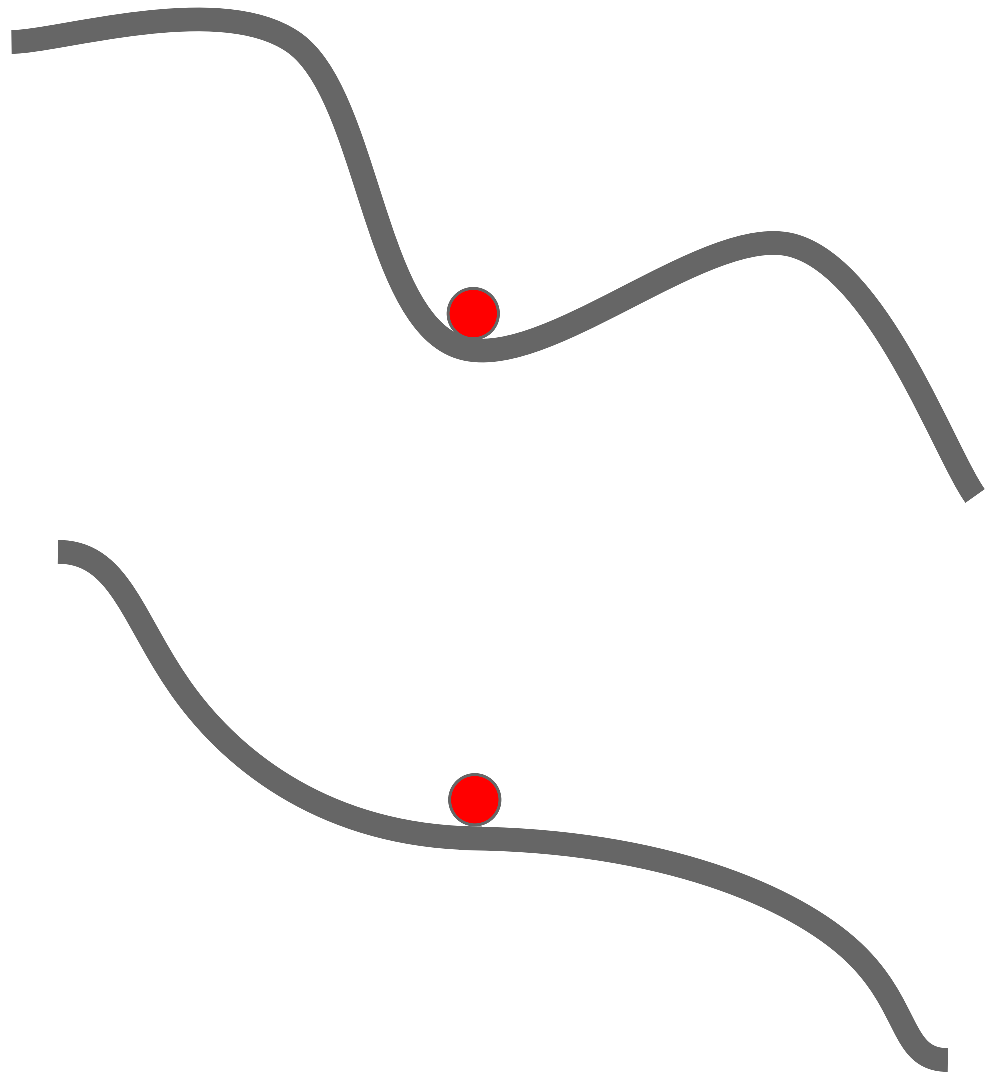


# Optimization

- Stochastic gradient *ascent*

$$w \leftarrow w + \alpha g, \quad g = \frac{\partial}{\partial w}\mathcal{L}$$

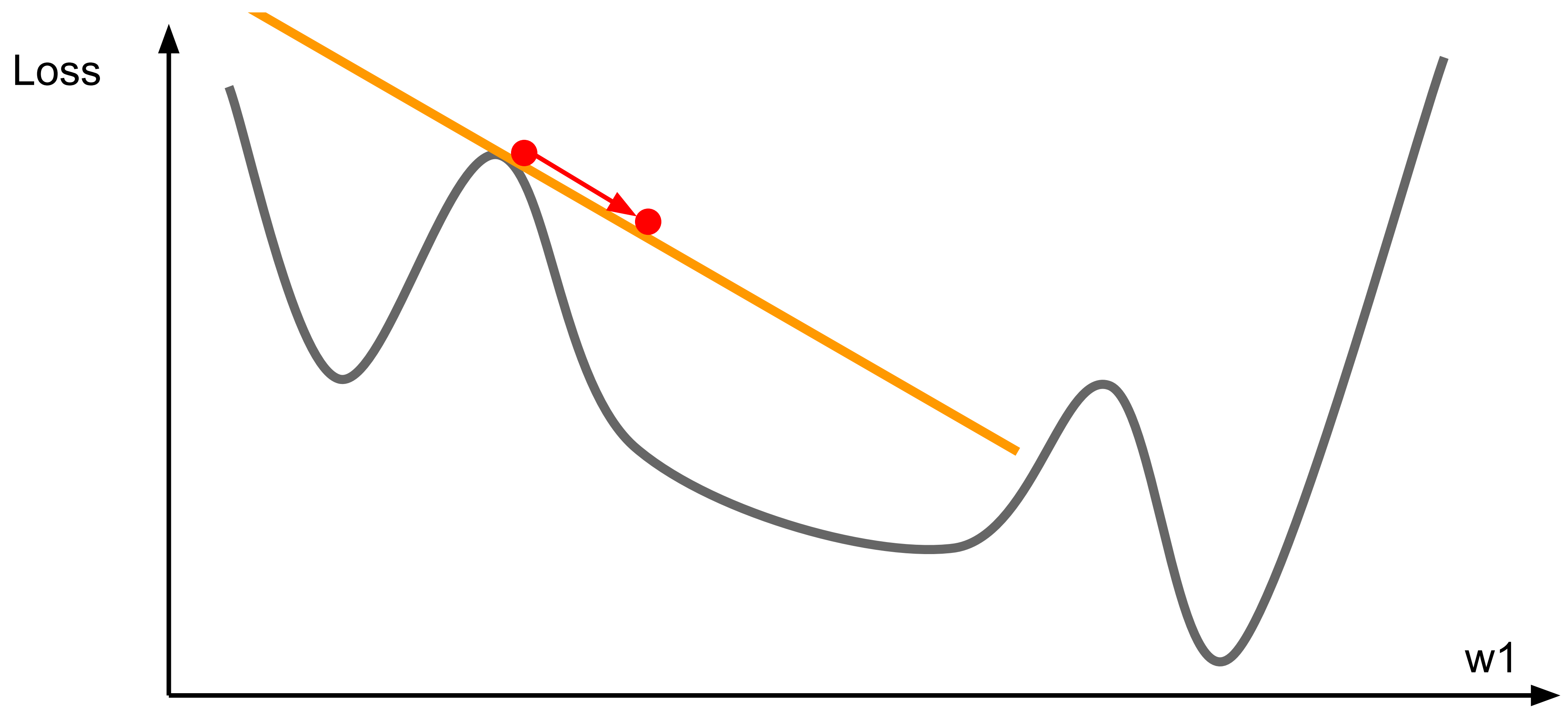
  - Very simple to code up
  - What if the loss function has a local minima or saddle point?

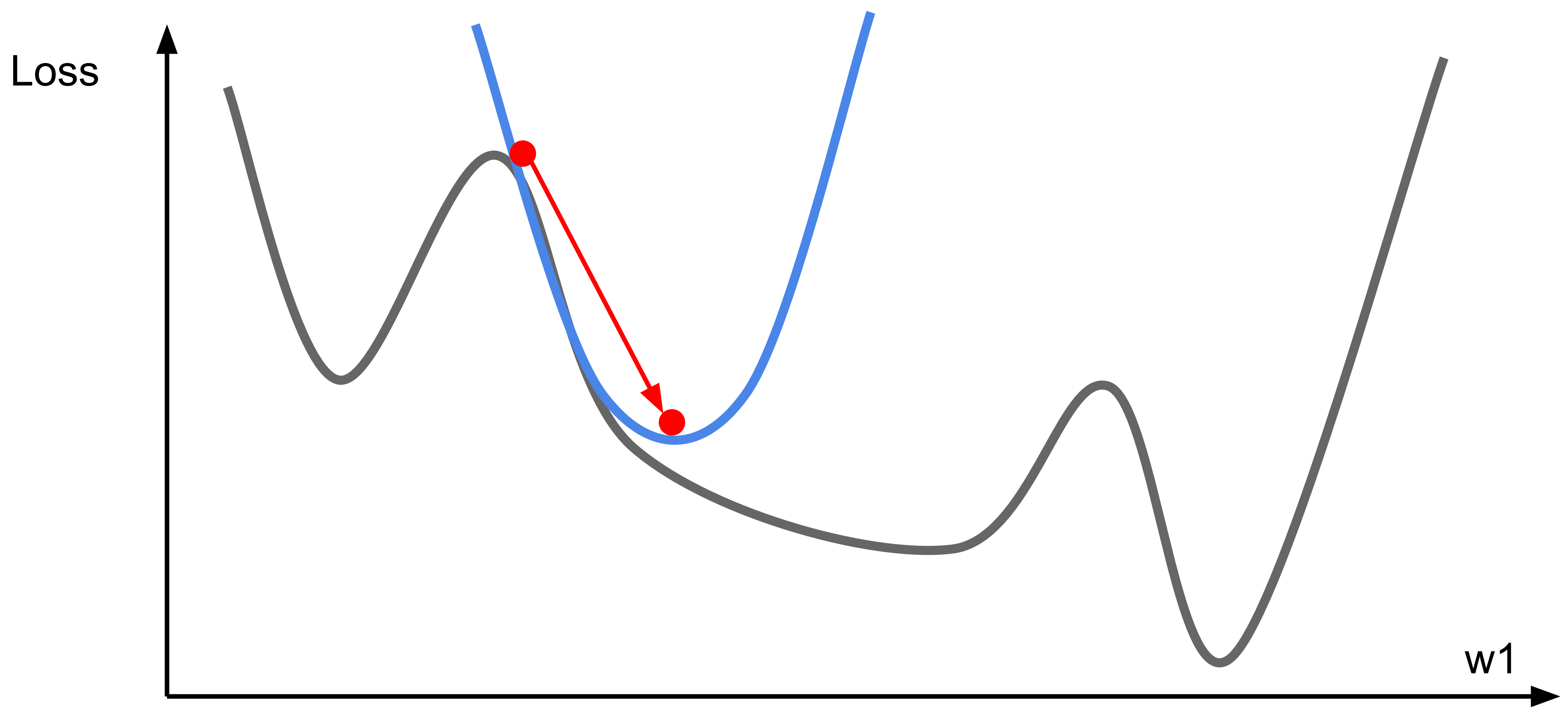


# Optimization

- Stochastic gradient *ascent*
  - Very simple to code up
  - "First-order" technique: only relies on having gradient

$$w \leftarrow w + \alpha g, \quad g = \frac{\partial}{\partial w}\mathcal{L}$$



# Optimization

- Stochastic gradient *ascent*

$$w \leftarrow w + \alpha g, \quad g = \frac{\partial}{\partial w}\mathcal{L}$$

  - Very simple to code up
  - "First-order" technique: only relies on having gradient
  - Setting step size is hard (decrease when held-out performance worsens?)

# Optimization

- Stochastic gradient *ascent*

$$w \leftarrow w + \alpha g, \quad g = \frac{\partial}{\partial w}\mathcal{L}$$

  - Very simple to code up
  - “First-order” technique: only relies on having gradient
  - Setting step size is hard (decrease when held-out performance worsens?)
- Newton’s method

# Optimization

- Stochastic gradient *ascent*

$$w \leftarrow w + \alpha g, \quad g = \frac{\partial}{\partial w}\mathcal{L}$$

  - Very simple to code up
  - "First-order" technique: only relies on having gradient
  - Setting step size is hard (decrease when held-out performance worsens?)
- Newton's method

$$w \leftarrow w + \left(\frac{\partial^2}{\partial w^2}\mathcal{L}\right)^{-1} g$$

# Optimization

- Stochastic gradient *ascent*
  - Very simple to code up
  - "First-order" technique: only relies on having gradient
  - Setting step size is hard (decrease when held-out performance worsens?)

$$w \leftarrow w + \alpha g, \quad g = \frac{\partial}{\partial w}\mathcal{L}$$

- Newton's method
  - Second-order technique

$$w \leftarrow w + \left(\frac{\partial^2}{\partial w^2}\mathcal{L}\right)^{-1} g$$

# Optimization

- Stochastic gradient *ascent*
  - Very simple to code up
  - "First-order" technique: only relies on having gradient
  - Setting step size is hard (decrease when held-out performance worsens?)

$$w \leftarrow w + \alpha g, \quad g = \frac{\partial}{\partial w}\mathcal{L}$$

- Newton's method
  - Second-order technique

$$w \leftarrow w + \left(\frac{\partial^2}{\partial w^2}\mathcal{L}\right)^{-1} g$$

Inverse Hessian: *n* x *n* mat, expensive!

# Optimization

- Stochastic gradient *ascent*
  - Very simple to code up
  - "First-order" technique: only relies on having gradient
  - Setting step size is hard (decrease when held-out performance worsens?)

$$w \leftarrow w + \alpha g, \quad g = \frac{\partial}{\partial w}\mathcal{L}$$

- Newton's method
  - Second-order technique
  - Optimizes quadratic instantly

$$w \leftarrow w + \left(\frac{\partial^2}{\partial w^2}\mathcal{L}\right)^{-1} g$$

Inverse Hessian: *n* x *n* mat, expensive!

# Optimization

- Stochastic gradient *ascent*

$$w \leftarrow w + \alpha g, \quad g = \frac{\partial}{\partial w}\mathcal{L}$$

  - Very simple to code up
  - "First-order" technique: only relies on having gradient
  - Setting step size is hard (decrease when held-out performance worsens?)
- Newton's method

$$w \leftarrow w + \left(\frac{\partial^2}{\partial w^2}\mathcal{L}\right)^{-1} g$$

  - Second-order technique
  - Optimizes quadratic instantly

Inverse Hessian: *n* x *n* mat, expensive!

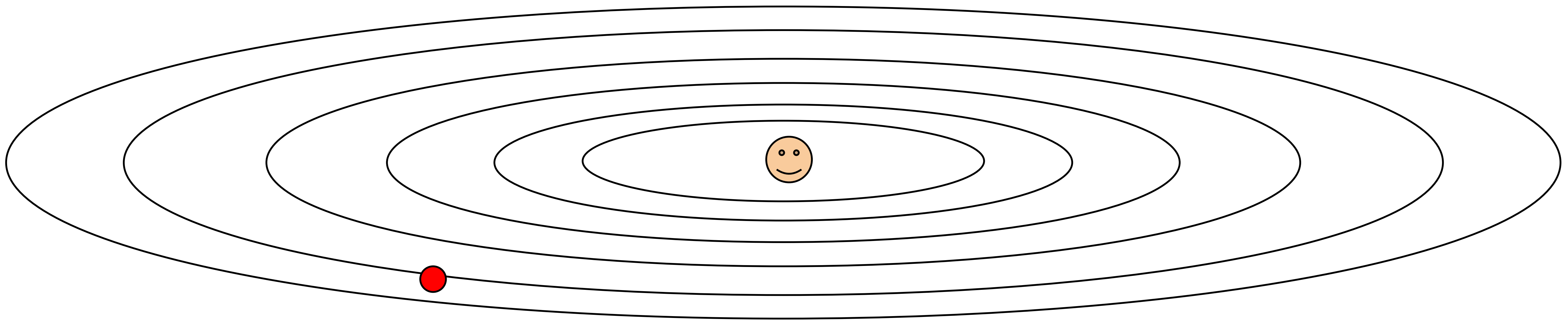
- Quasi-Newton methods: L-BFGS, etc. approximate inverse Hessian

# AdaGrad

- Optimized for problems with sparse features

- Per-parameter learning rate: smaller updates are made to parameters that get updated frequently

```
grad_squared = 0
while True:
  dx = compute_gradient(x)
  grad_squared += dx * dx
  x -= learning_rate * dx / (np.sqrt(grad_squared) + 1e-7)
```



Duchi et al. (2011)

# AdaGrad

- Optimized for problems with sparse features
- Per-parameter learning rate: smaller updates are made to parameters that get updated frequently

Duchi et al. (2011)

# AdaGrad

- Optimized for problems with sparse features

- Per-parameter learning rate: smaller updates are made to parameters that get updated frequently

$$w_i \leftarrow w_i + \alpha \frac{1}{\sqrt{\epsilon + \sum_{\tau=1}^{t} g_{\tau,i}^2}} g_{t_i}$$

← (smoothed) sum of squared gradients from all updates

Duchi et al. (2011)

# AdaGrad

- Optimized for problems with sparse features
- Per-parameter learning rate: smaller updates are made to parameters that get updated frequently

$$w_i \leftarrow w_i + \alpha \frac{1}{\sqrt{\epsilon + \sum_{\tau=1}^{t} g_{\tau,i}^2}} g_{t_i}$$

(smoothed) sum of squared gradients from all updates

- Generally more robust than SGD, requires less tuning of learning rate

Duchi et al. (2011)

# AdaGrad

- Optimized for problems with sparse features

- Per-parameter learning rate: smaller updates are made to parameters that get updated frequently

$$w_i \leftarrow w_i + \alpha \frac{1}{\sqrt{\epsilon + \sum_{\tau=1}^{t} g_{\tau,i}^2}} g_{t_i}$$

(smoothed) sum of squared gradients from all updates

- Generally more robust than SGD, requires less tuning of learning rate

- Other techniques for optimizing deep models — more later!

Duchi et al. (2011)

# Summary

# Summary

- Design tradeoffs need to reflect interactions:

# Summary

- Design tradeoffs need to reflect interactions:
  - Model and objective are coupled: probabilistic model <-> maximize likelihood

# Summary

- Design tradeoffs need to reflect interactions:
  - Model and objective are coupled: probabilistic model <-> maximize likelihood
  - …but not always: a linear model or neural network can be trained to minimize any differentiable loss function

# Summary

- Design tradeoffs need to reflect interactions:
  - Model and objective are coupled: probabilistic model <-> maximize likelihood
  - …but not always: a linear model or neural network can be trained to minimize any differentiable loss function
  - Inference governs what learning: need to be able to compute expectations to use logistic regression