# Multiclass Classification

Wei Xu

(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.

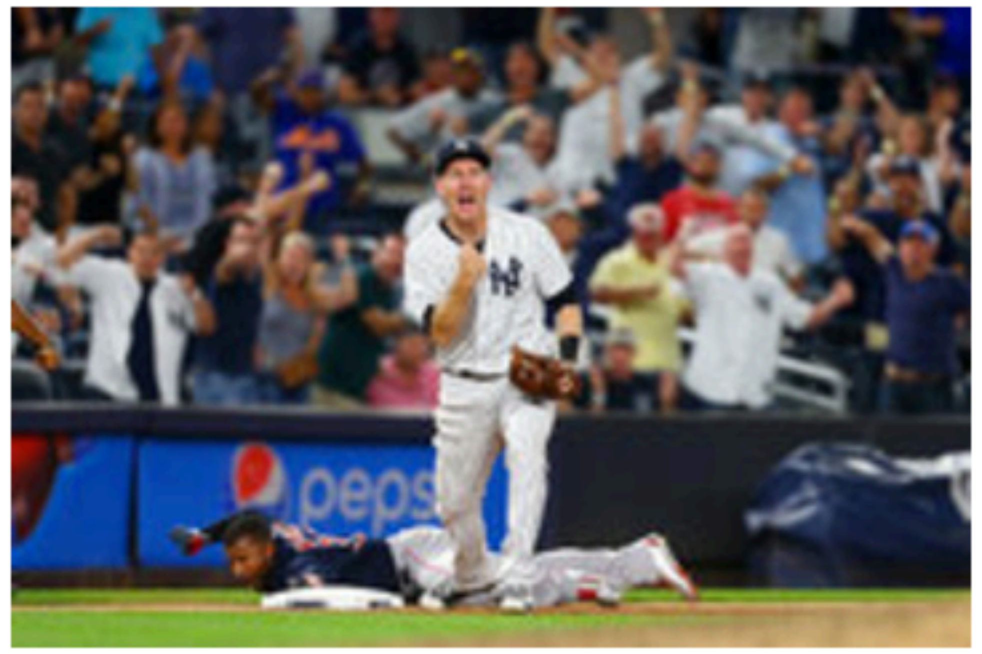
By GINA KOLATA

→ 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

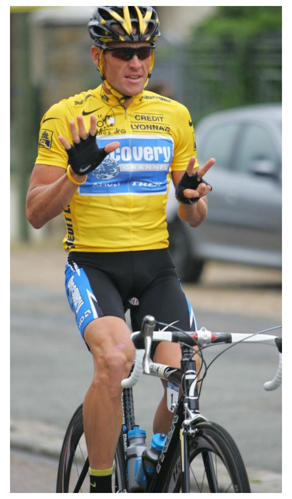
→ Dog

→ Car

- Thousands of classes (ImageNet)

# Entity Linking

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…

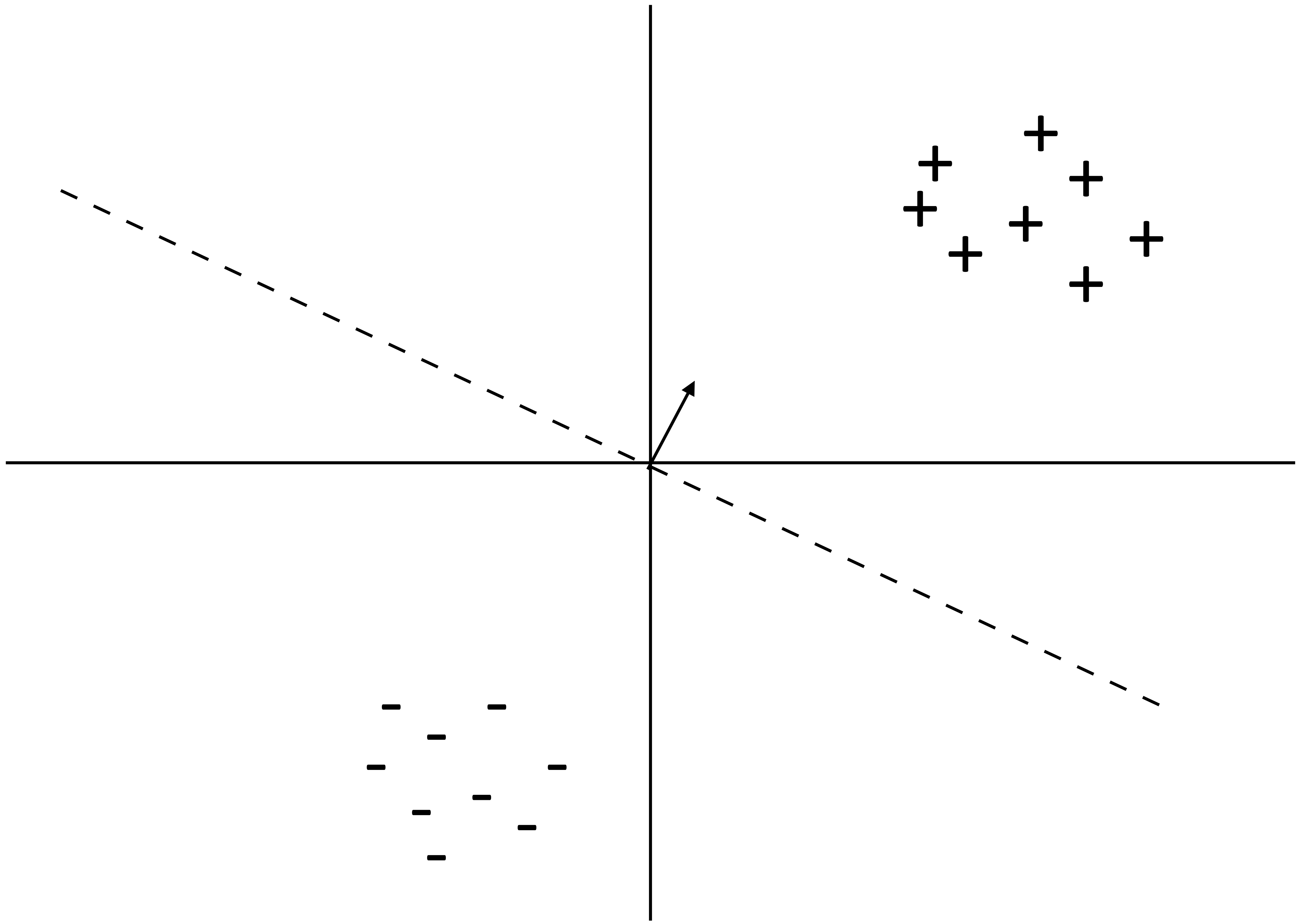
?

?

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

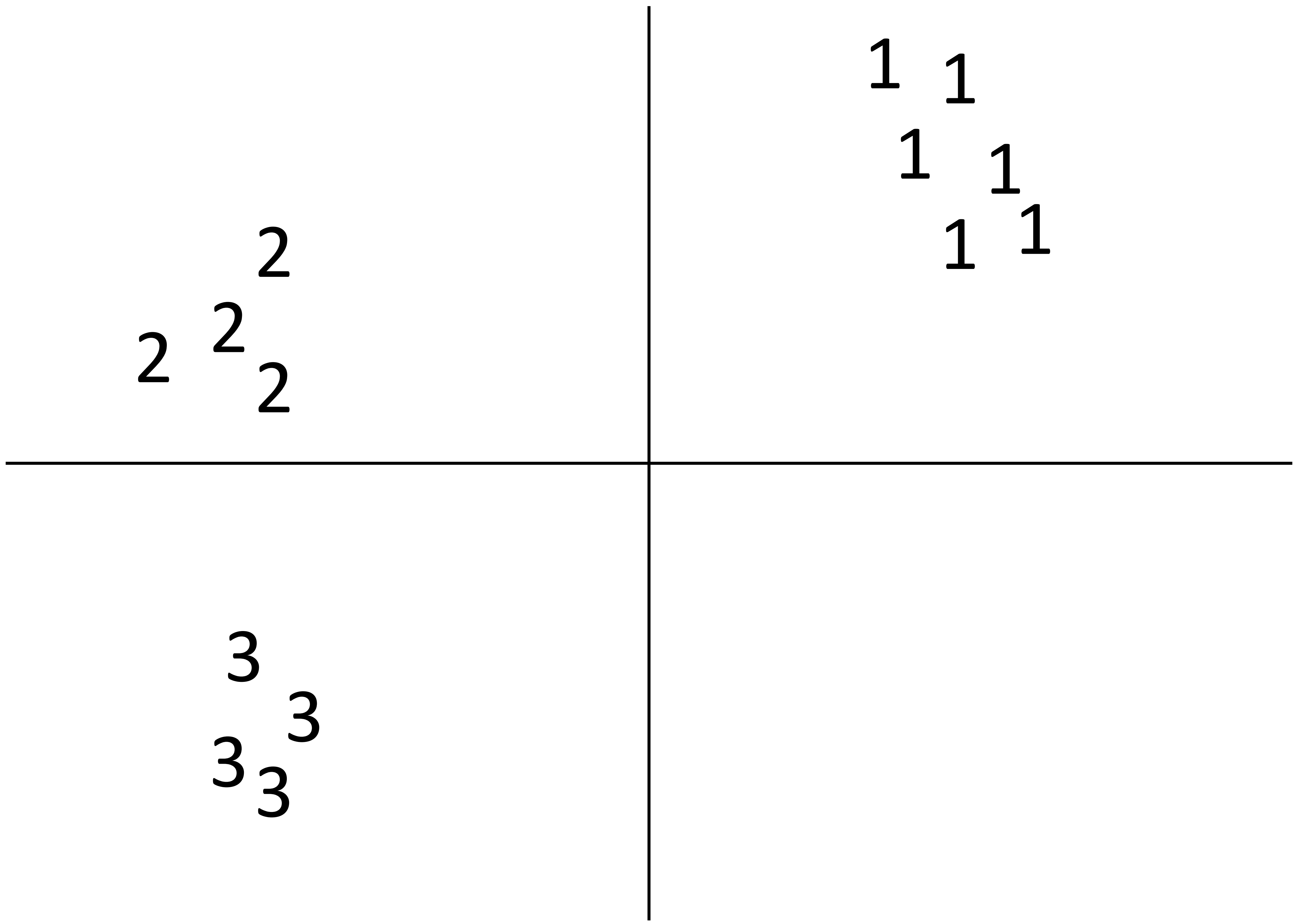
# Binary Classification

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



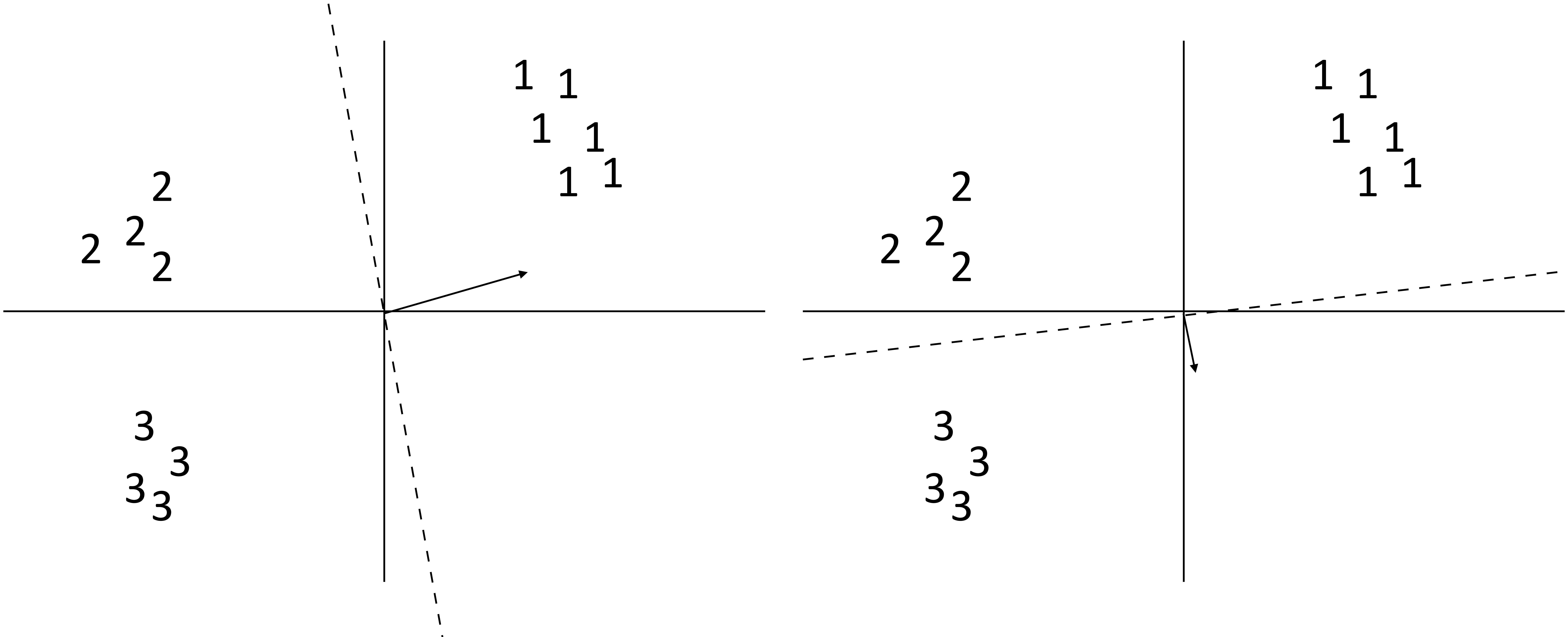
# Multiclass Classification

- Can we just use binary classifiers here?



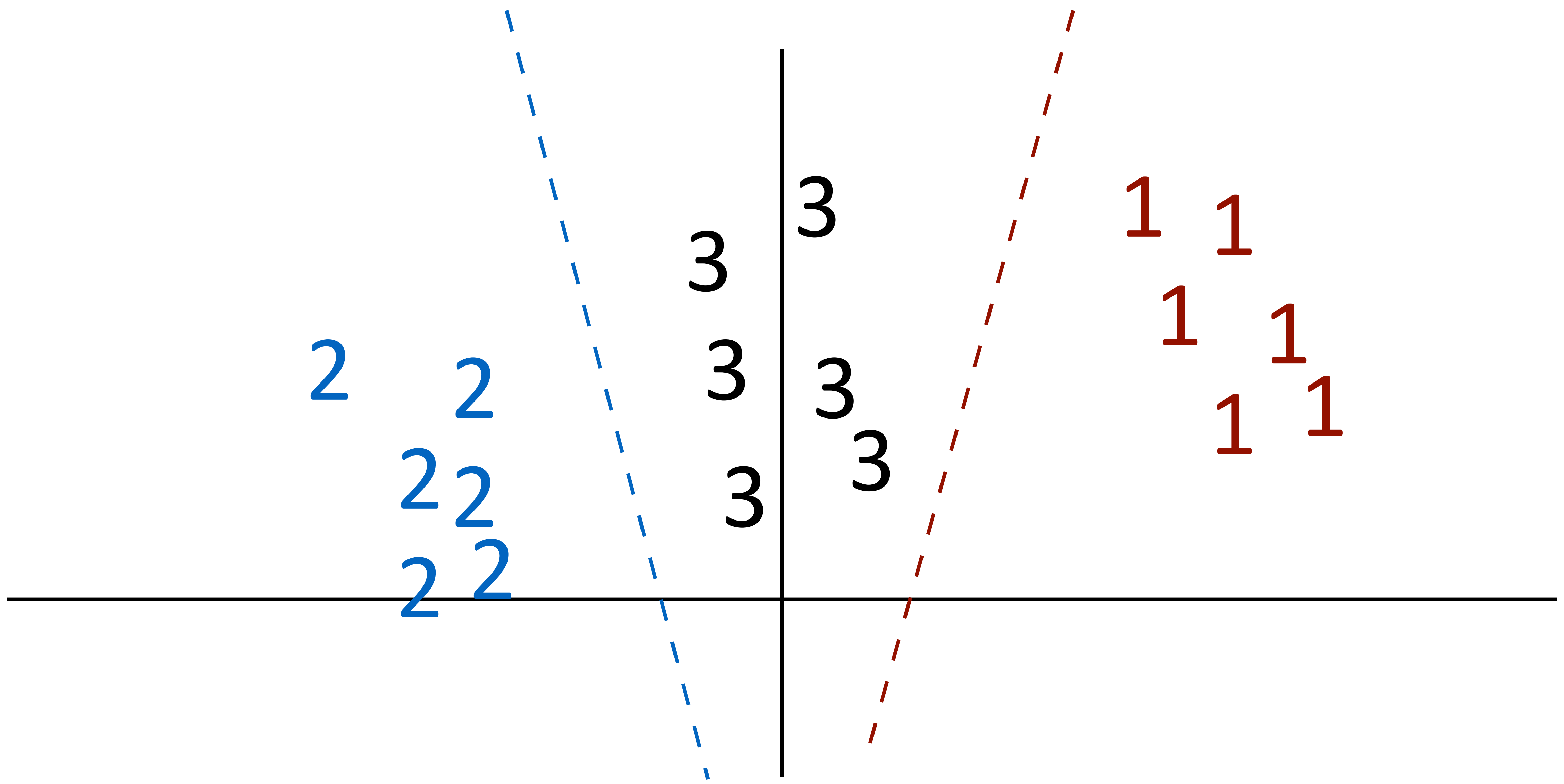
# Multiclass Classification

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



# Multiclass Classification

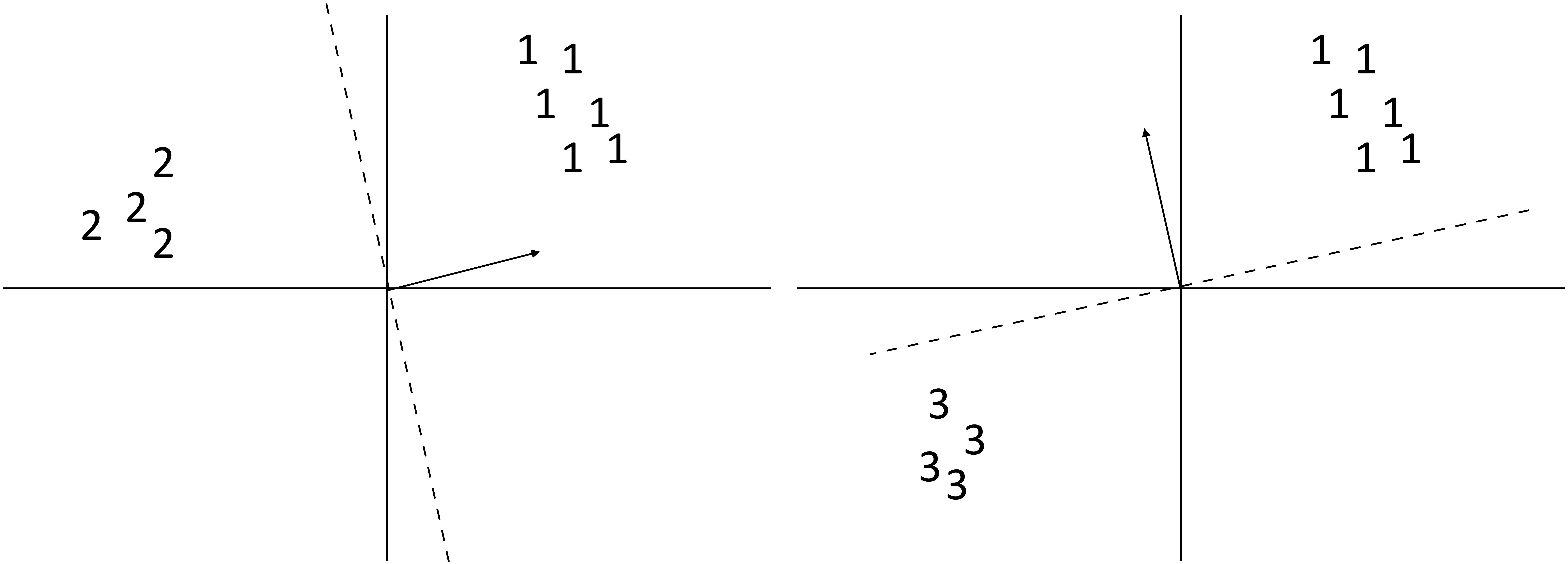
- Not all classes may even be separable using this approach




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

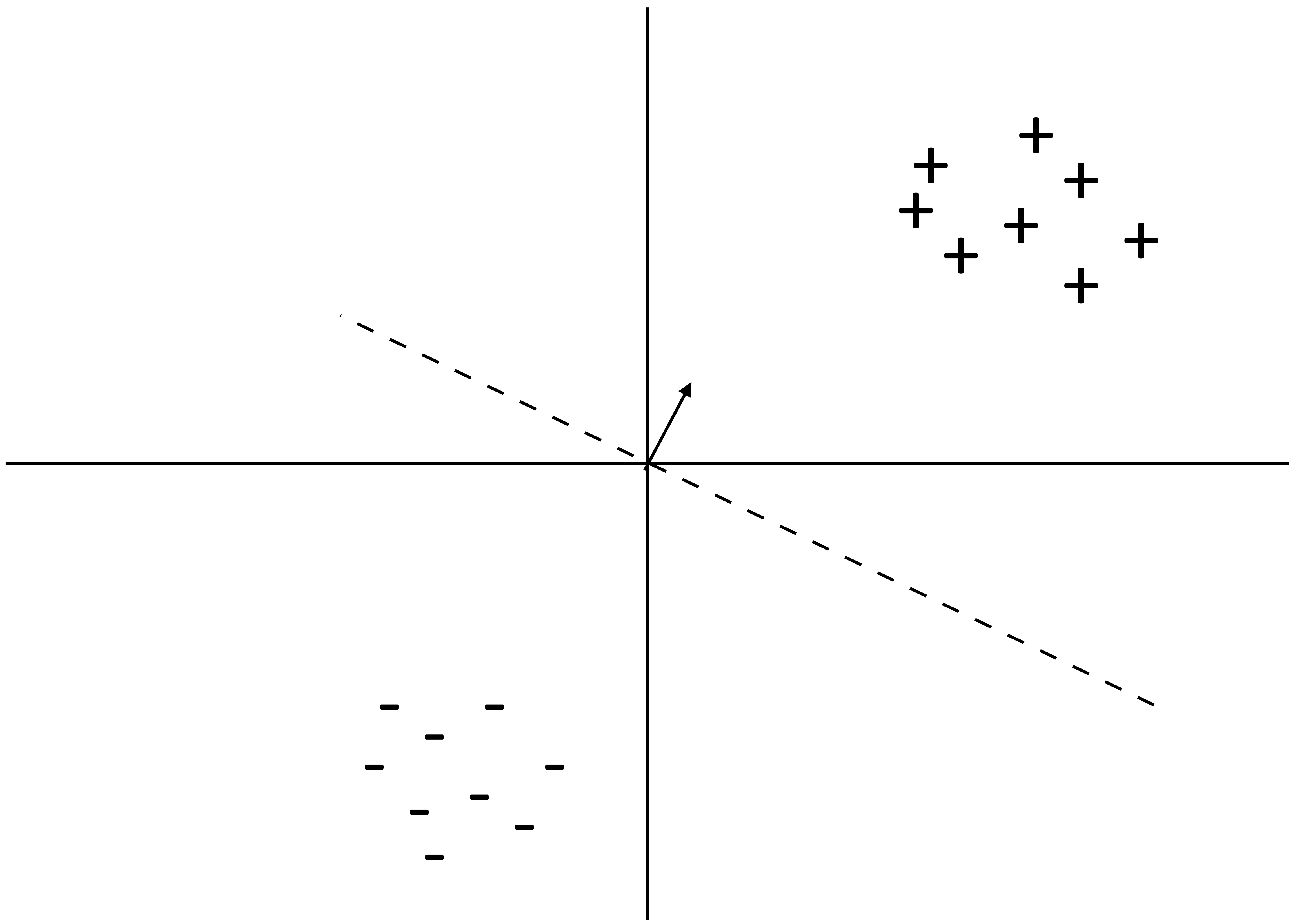
# Multiclass Classification

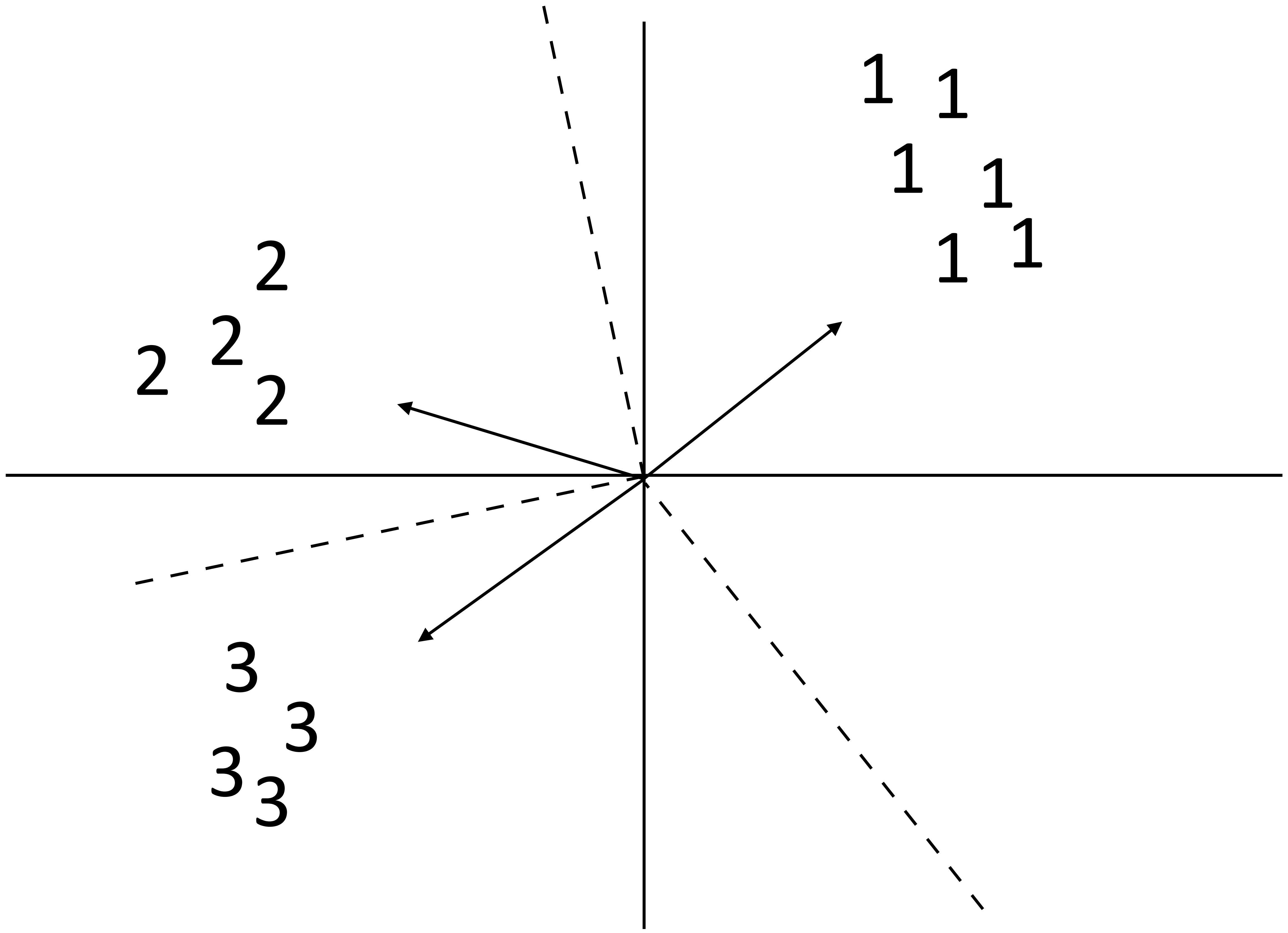
- All-vs-all: train n(n-1)/2 classifiers to differentiate each pair of classes
- Again, how to reconcile?



# Multiclass Classification

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



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

*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]

"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

- Classify *blocks* as one of 36 POS tags

*the router* *blocks* *the packets*

NNS
VBZ
NN
DT
...

- Example *x*: sentence with a word (in this case, *blocks*) highlighted

- Extract features with respect to this word:

$$f(x, y=\text{VBZ}) = \text{I[curr\_word=}\textit{blocks}\text{ \& tag = VBZ]},$$
$$\text{I[prev\_word=}\textit{router}\text{ \& tag = VBZ]}$$
$$\text{I[next\_word=}\textit{the}\text{ \& tag = VBZ]}$$
$$\text{I[curr\_suffix=}\textit{s}\text{ \& tag = VBZ]}$$

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

- Next two lectures: sequence labeling!

# Multiclass Logistic Regression

# 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

- 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

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

*too many drug trials, too few patients*

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

log(0.21) = - 1.56

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

# Multiclass Logistic Regression

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

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

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

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

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

model's expectation of feature value

gold 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]

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

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

- Learning: gradient ascent on the discriminative log-likelihood

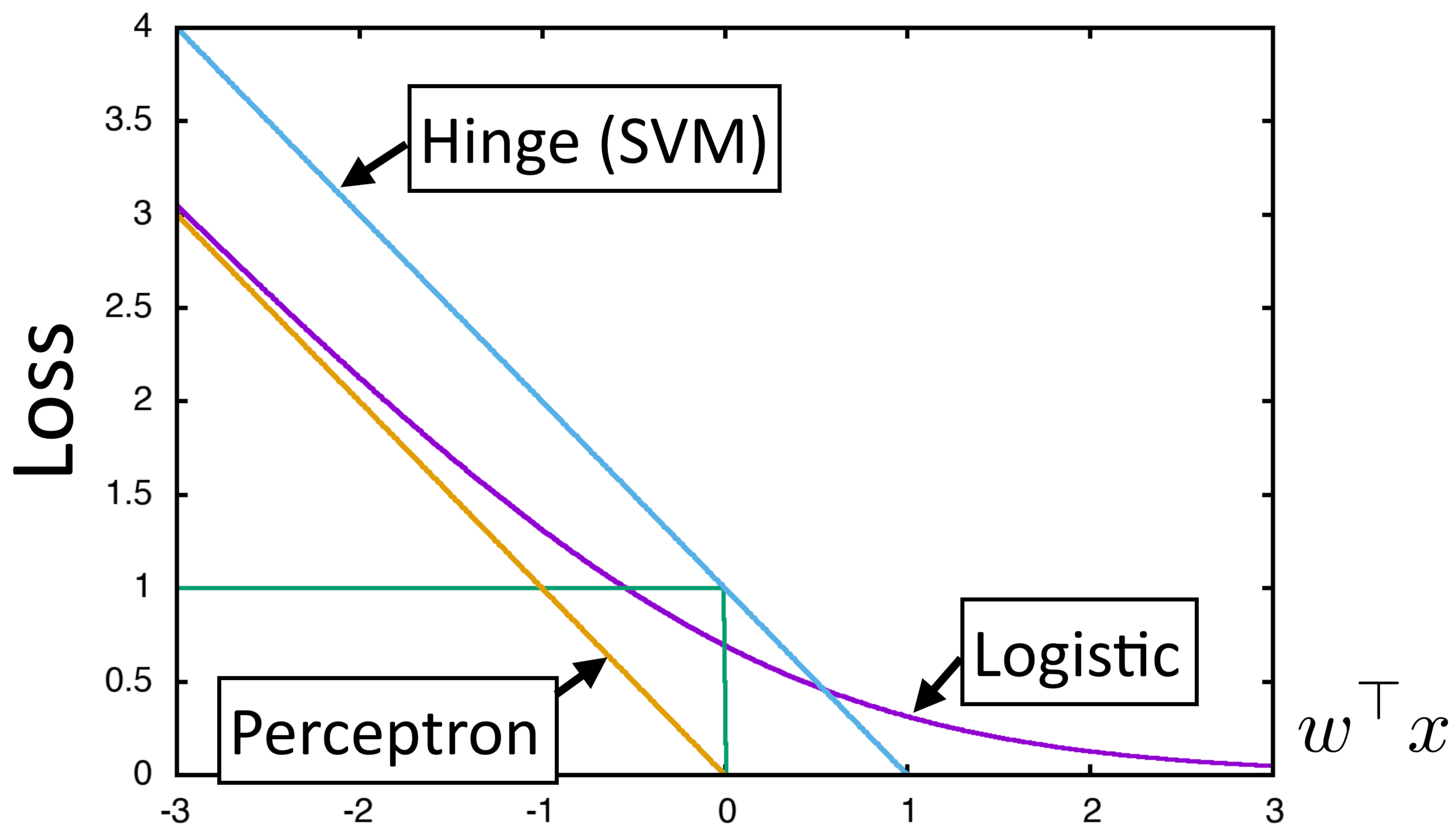
$$f(x,y^*) - \mathbb{E}_y[f(x,y)] = f(x,y^*) - \sum_y [P_w(y|x) f(x,y)]$$

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

# Optimization

# Recap

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



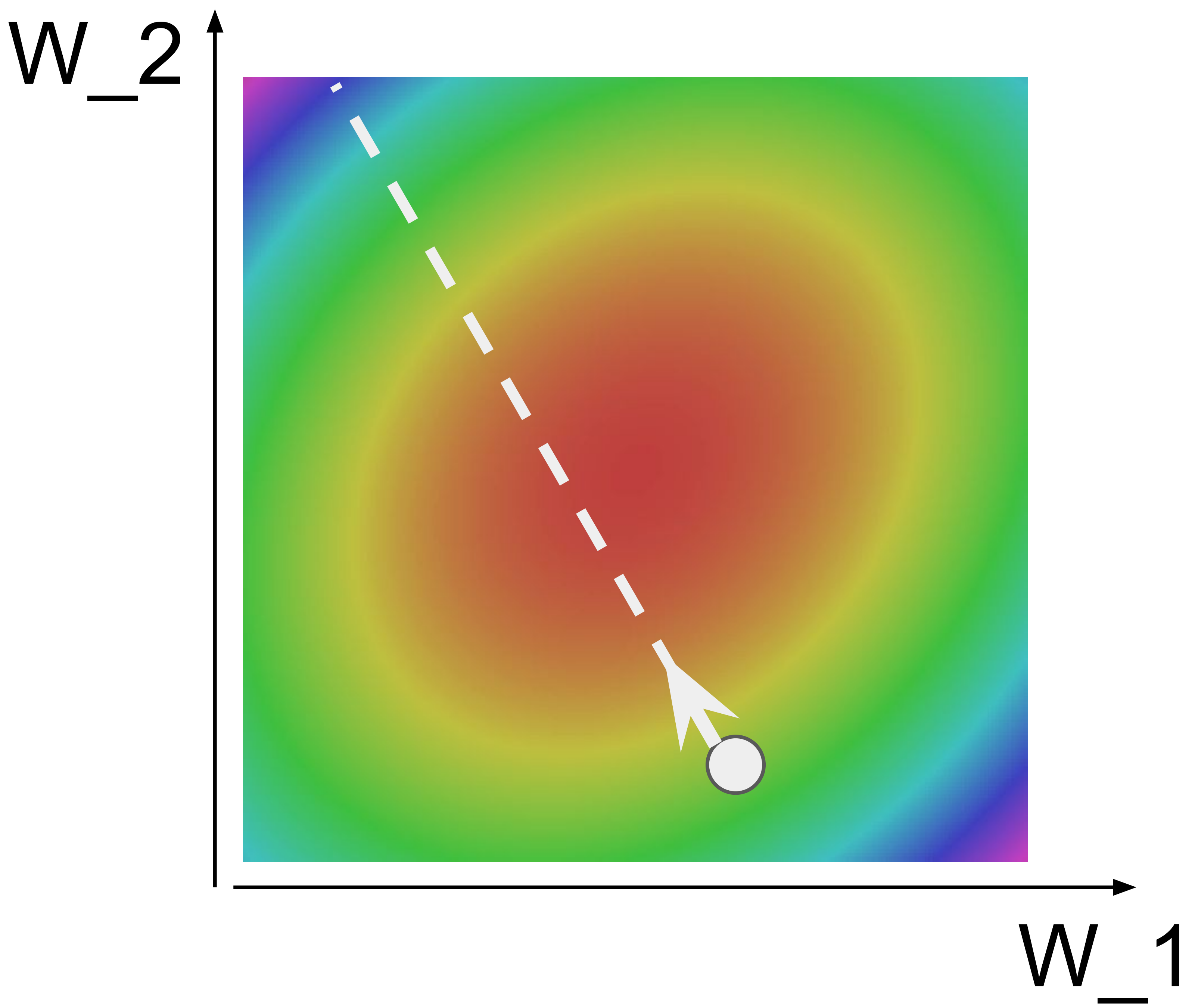

  - Inference: just maxes and simple expectations so far, but will get harder
  - Training: gradient descent?

# Optimization

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

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

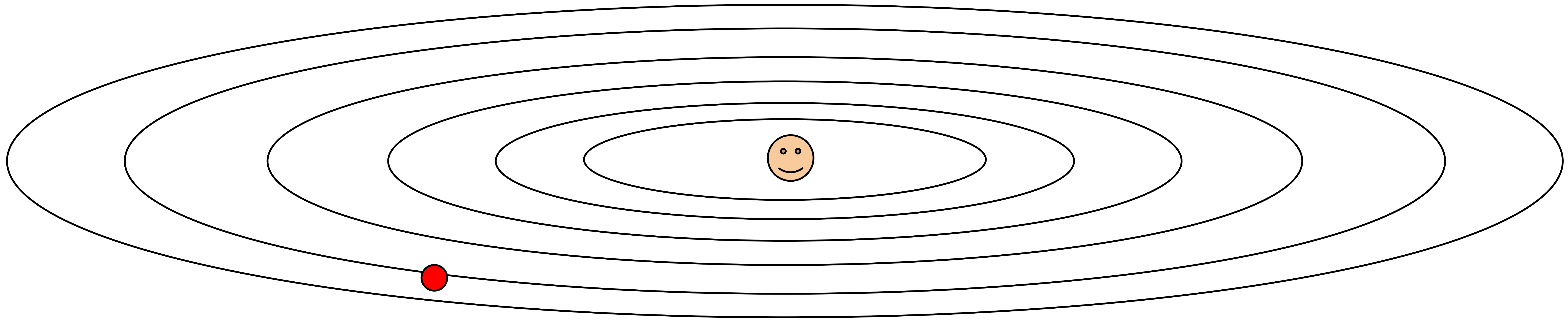
```
# Vanilla Gradient Descent

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?

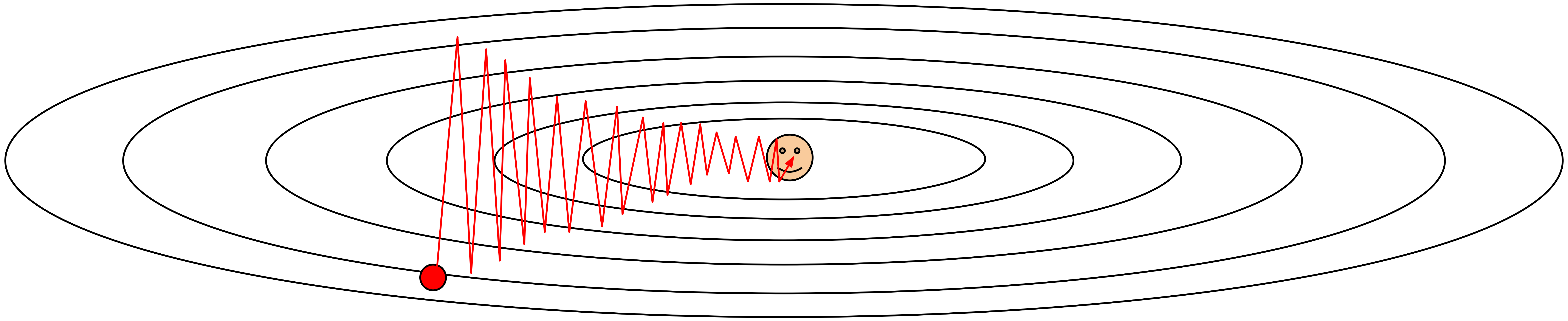
$$w \leftarrow w + \alpha g, \quad g = \frac{\partial}{\partial w}\mathcal{L}$$



# Optimization

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

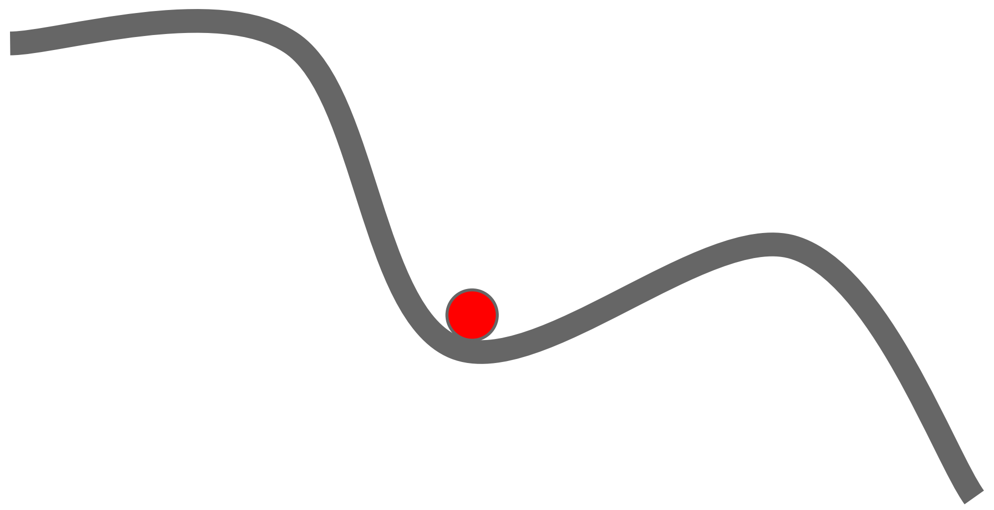
$$w \leftarrow w + \alpha g, \quad g = \frac{\partial}{\partial w}\mathcal{L}$$

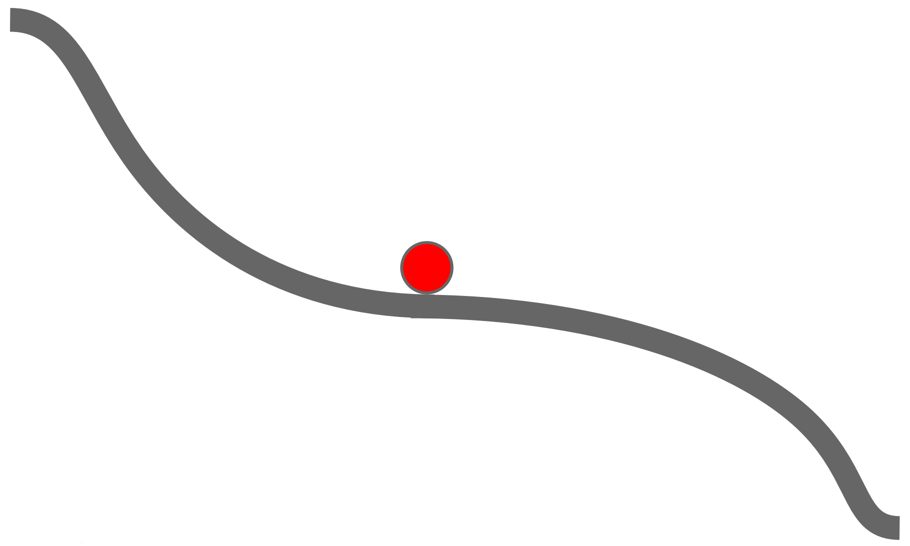


# Optimization

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

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



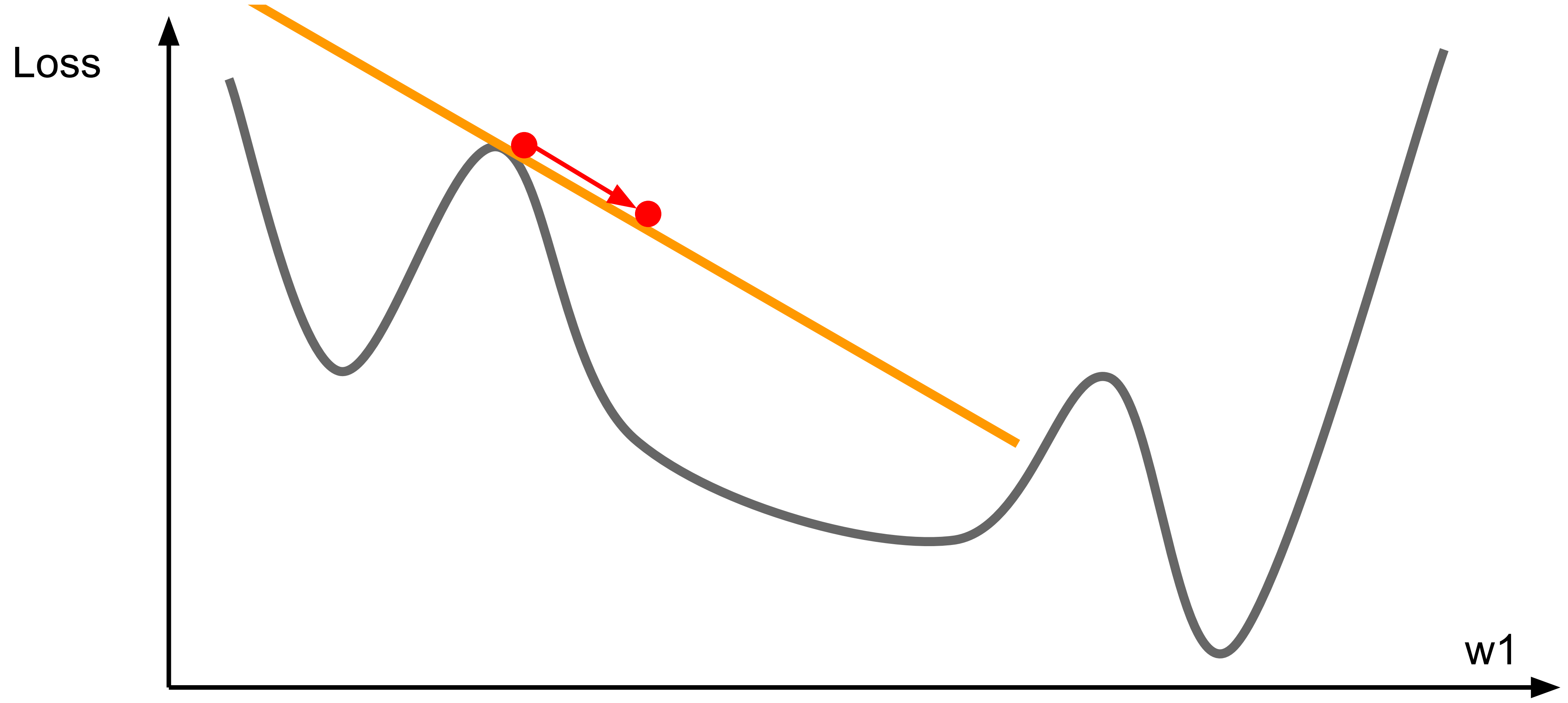


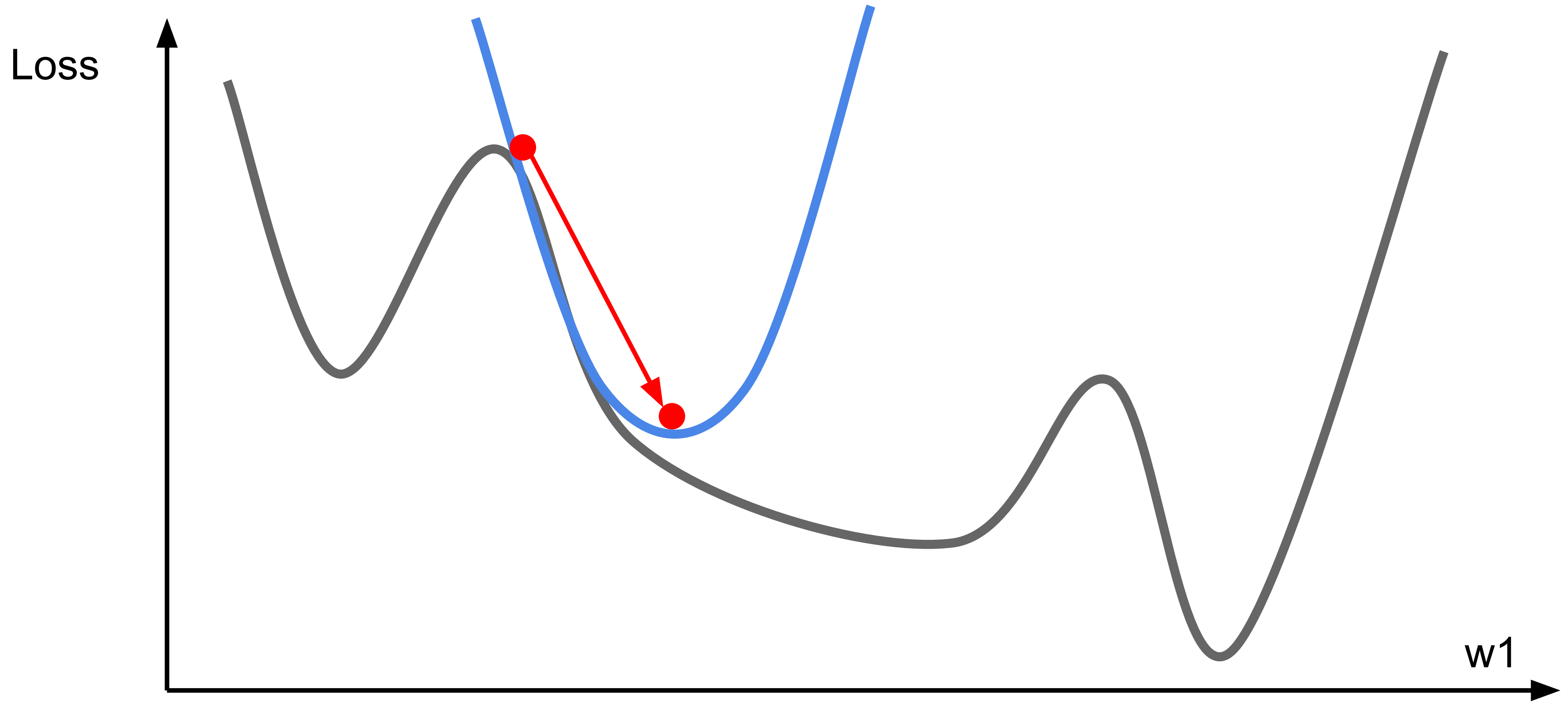
"Identifying and attacking the saddle point problem in high-dimensional non-convex optimization"
Dauphin et al. (2014)

# 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*
  - 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!

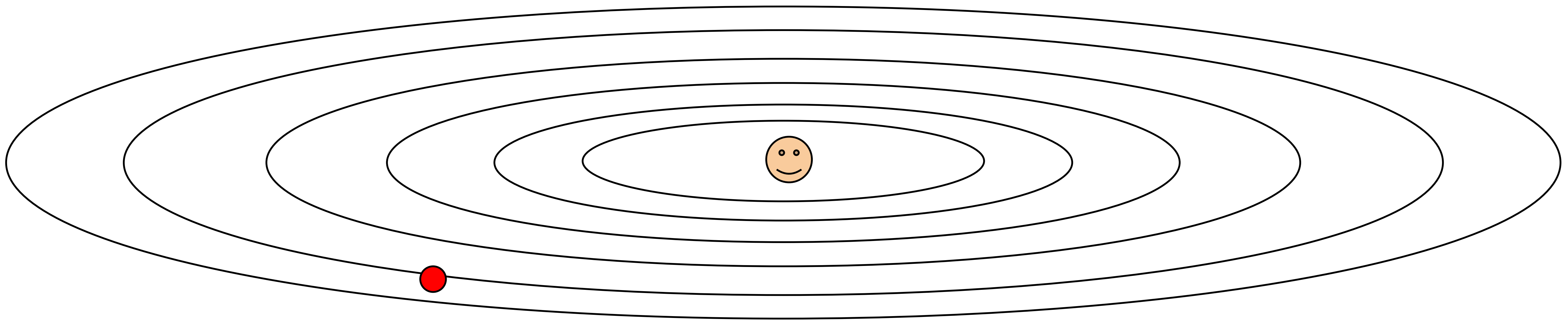
- Quasi-Newton methods: L-BFGS, etc. approximate inverse Hessian

# 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

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

← (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

- 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