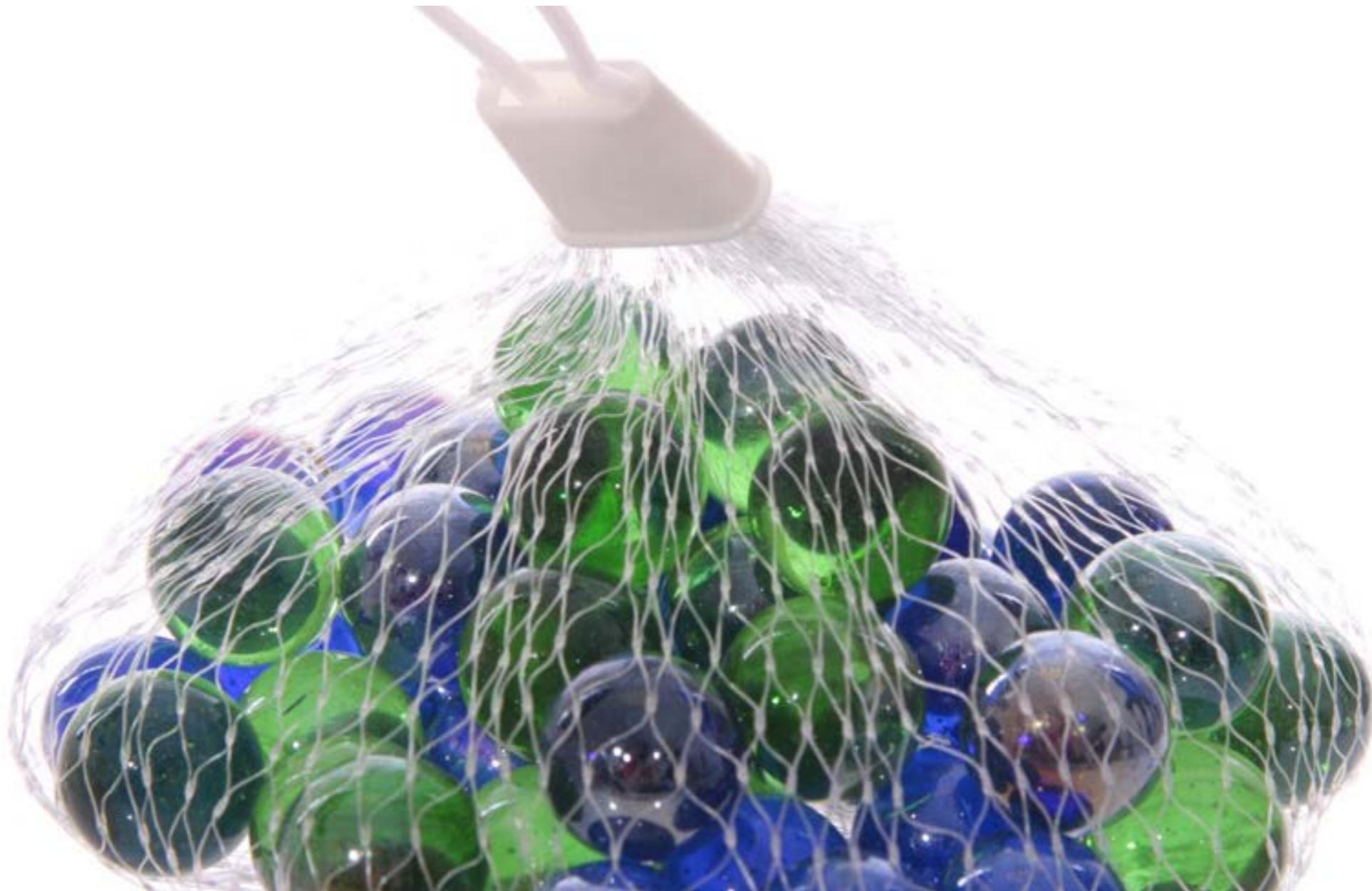


Image classification



- Slide credits Kris Kitani, Noah Snavely, Fei-Fei Li

Overview of today's lecture

- Bag-of-words.
- K-means clustering.
- Classification.
- K nearest neighbors.
- Naïve Bayes.
- Support vector machine.

Image Classification



(assume given set of discrete labels)
{dog, cat, truck, plane, ...}



cat

Image Classification: Problem



08	02	22	97	38	15	00	40	00	75	04	05	07	78	52	12	50	77	81	02
49	49	99	40	17	81	18	57	60	87	17	40	98	43	69	46	04	56	62	00
81	49	31	73	55	79	14	29	93	71	40	67	55	88	30	03	49	13	36	65
02	70	95	23	04	60	11	42	62	87	55	56	01	32	86	71	97	02	36	91
22	31	16	71	51	67	45	89	41	92	36	54	22	40	40	28	66	33	13	80
24	47	29	60	99	03	45	02	44	75	33	53	78	36	84	20	35	17	12	50
32	95	81	28	64	23	67	10	26	38	40	67	89	54	70	66	18	38	64	70
67	26	20	68	02	62	12	20	93	63	94	39	63	08	40	91	66	49	94	21
24	95	58	03	66	73	99	26	97	17	78	78	96	03	14	88	34	89	63	72
21	36	23	09	75	00	76	44	20	45	35	14	00	61	33	97	34	31	33	95
78	17	53	28	22	75	31	67	15	94	03	80	04	62	16	14	09	53	56	92
16	39	05	42	96	85	31	47	55	58	88	24	00	17	64	24	36	29	85	57
86	56	00	48	35	71	89	07	08	44	44	37	44	60	21	38	51	54	17	88
19	80	81	68	05	94	47	69	28	73	92	13	86	52	17	77	04	89	55	40
04	52	08	83	97	35	99	16	07	97	57	32	16	26	26	79	33	27	98	66
45	44	68	87	57	62	20	72	03	46	33	67	46	55	12	32	63	93	53	69
04	42	16	73	38	95	39	11	24	94	72	18	08	16	29	32	40	62	76	36
20	69	36	41	72	30	23	88	31	68	89	82	67	59	85	74	04	36	16	
20	73	35	29	78	31	90	01	74	31	49	71	95	00	31	16	23	57	08	54
01	70	54	71	83	51	54	69	16	92	33	48	61	43	52	01	89	11	42	48

What the computer sees

image classification

82% cat
15% dog
2% hat
1% mug

Data-driven approach

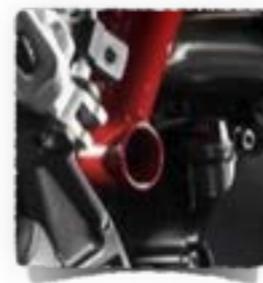
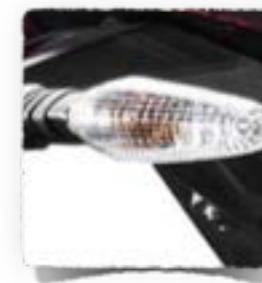
- Collect a database of images with labels
- Use ML to train an image classifier
- Evaluate the classifier on test images

Example training set

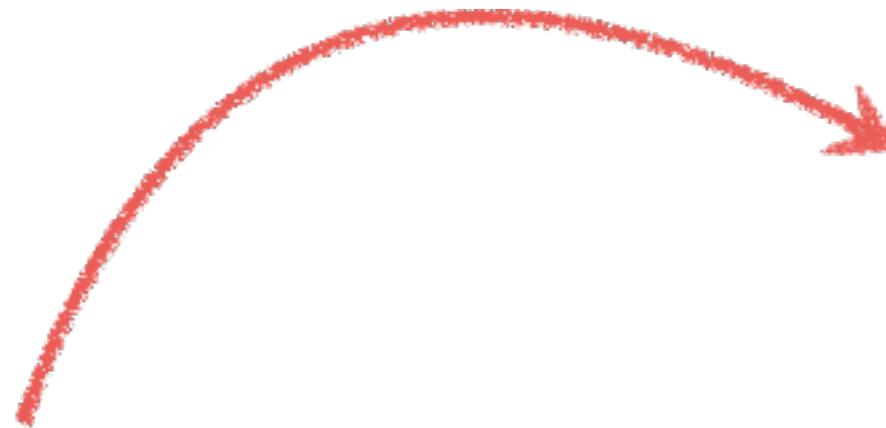


Bag of words

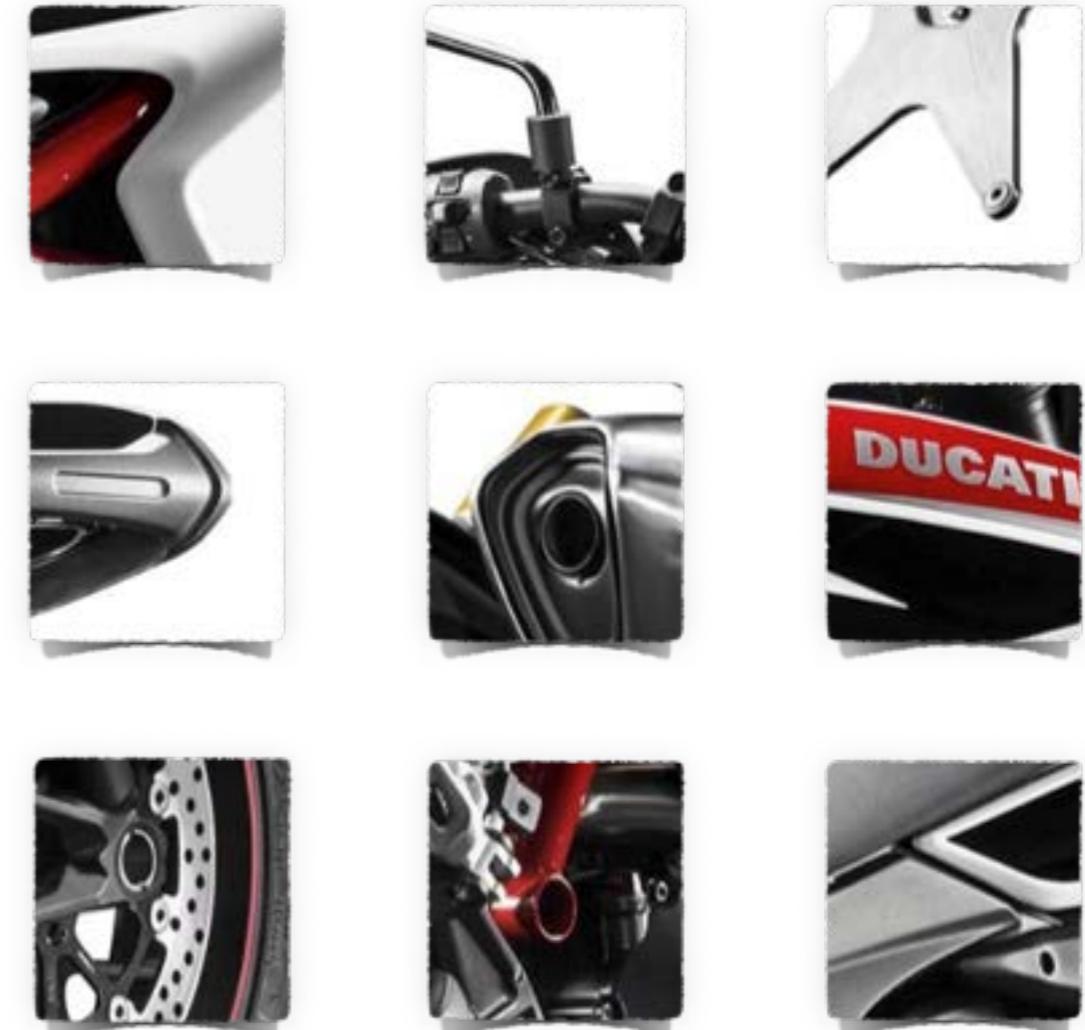
What object do these parts belong to?



Some local feature are very informative



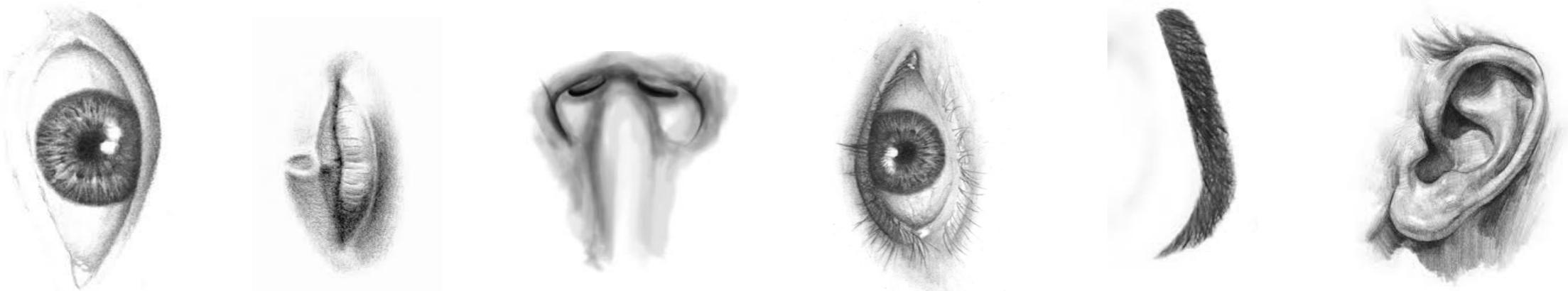
An object as



a collection of local features
(bag-of-features)

- deals well with occlusion
- scale invariant
- rotation invariant

(not so) crazy assumption



spatial information of local features
can be ignored for object recognition (i.e., verification)

CalTech6 dataset



class	bag of features	bag of features	Parts-and-shape model
	Zhang et al. (2005)	Willamowski et al. (2004)	Fergus et al. (2003)
airplanes	98.8	97.1	90.2
cars (rear)	98.3	98.6	90.3
cars (side)	95.0	87.3	88.5
faces	100	99.3	96.4
motorbikes	98.5	98.0	92.5
spotted cats	97.0	—	90.0

Works pretty well for image-level classification

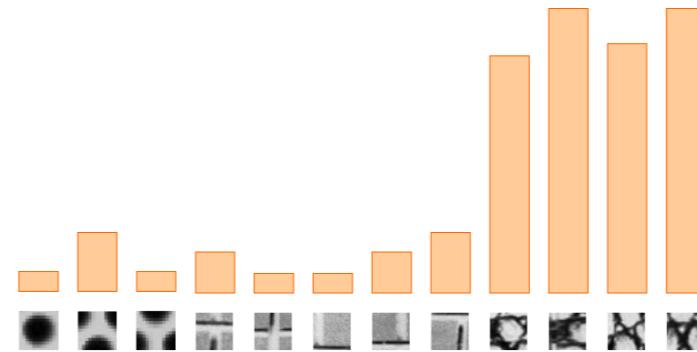
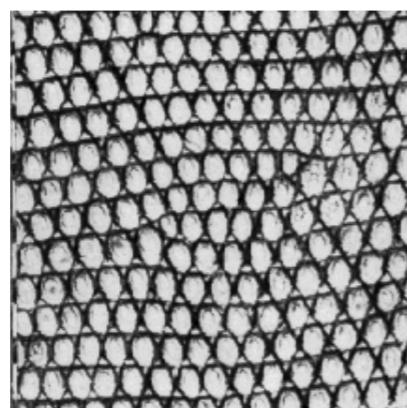
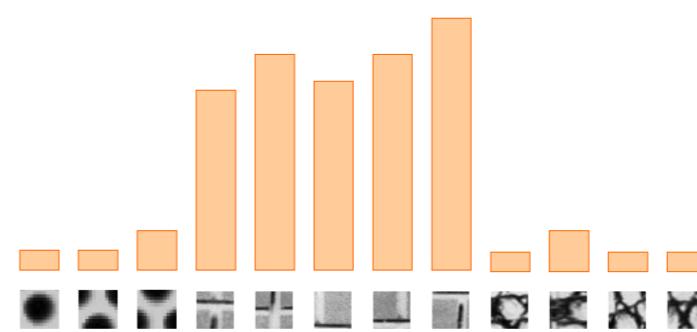
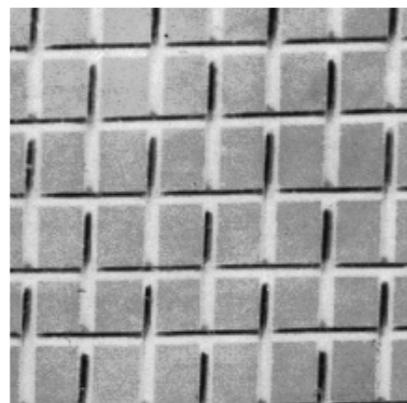
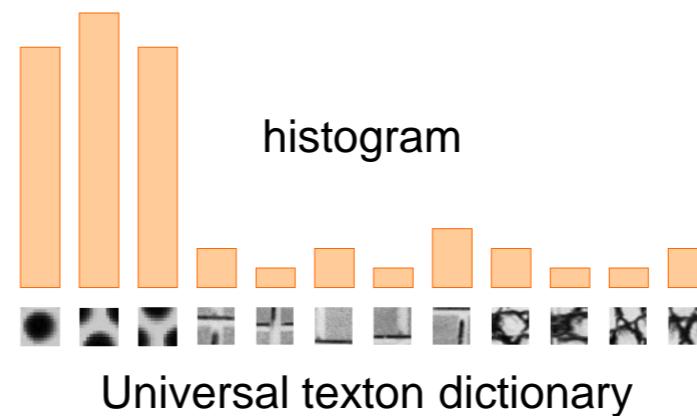
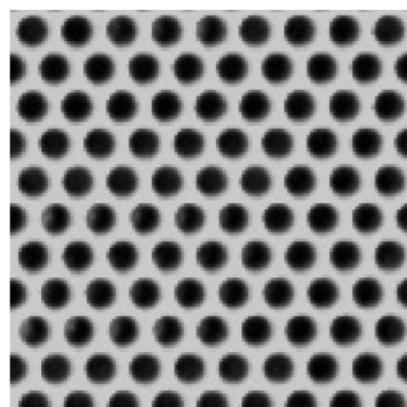
Bag-of-features

represent a data item (document, texture, image)
as a histogram over features

an old idea

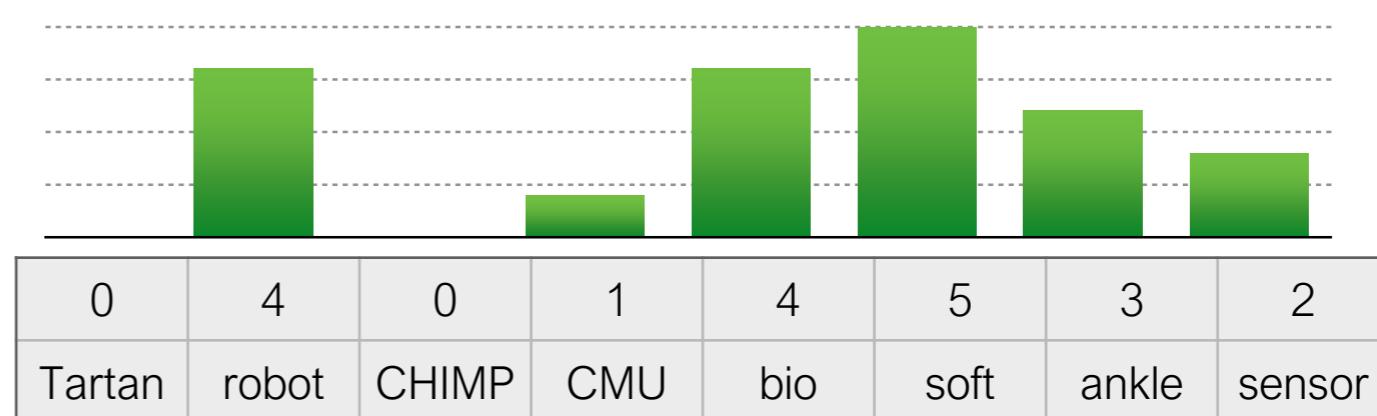
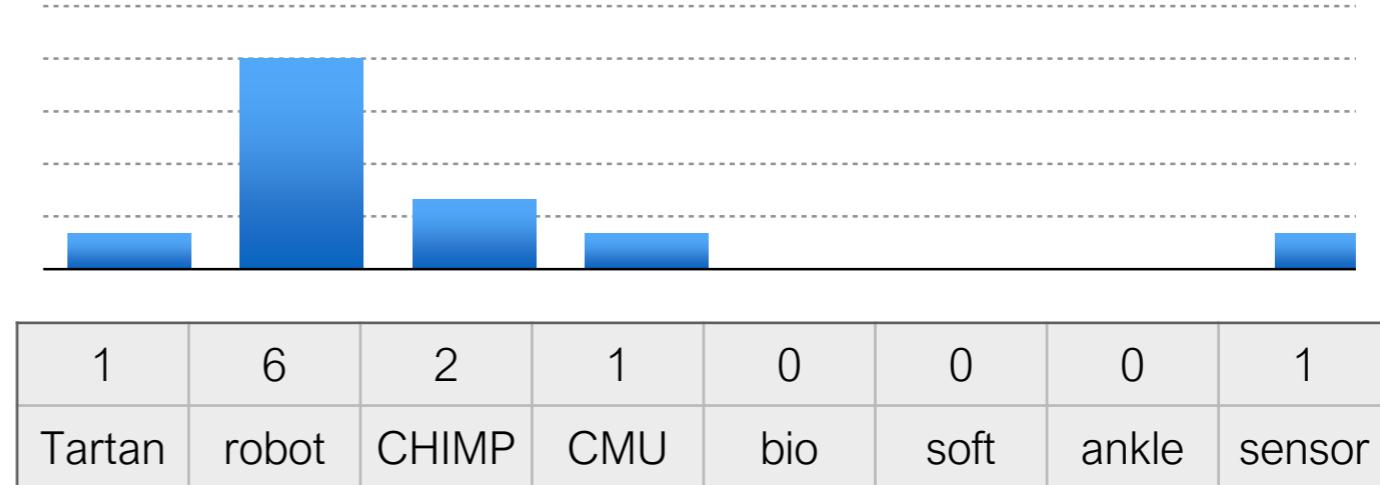
(e.g., texture recognition and information retrieval)

Texture recognition



Vector Space Model

G. Salton. 'Mathematics and Information Retrieval' Journal of Documentation, 1979



A document (datapoint) is a vector of counts over each word (feature)

$$\mathbf{v}_d = [n(w_{1,d}) \ n(w_{2,d}) \ \dots \ n(w_{T,d})]$$

$n(\cdot)$ counts the number of occurrences

just a histogram over words

What is the similarity between two documents?



A document (datapoint) is a vector of counts over each word (feature)

$$\mathbf{v}_d = [n(w_{1,d}) \ n(w_{2,d}) \ \dots \ n(w_{T,d})]$$

$n(\cdot)$ counts the number of occurrences

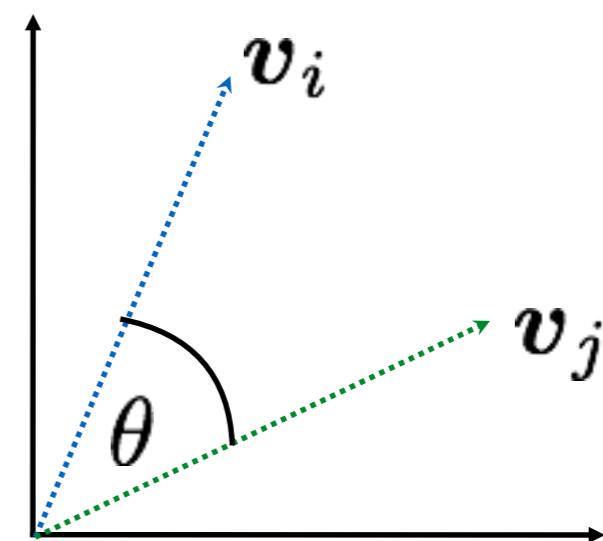
just a histogram over words

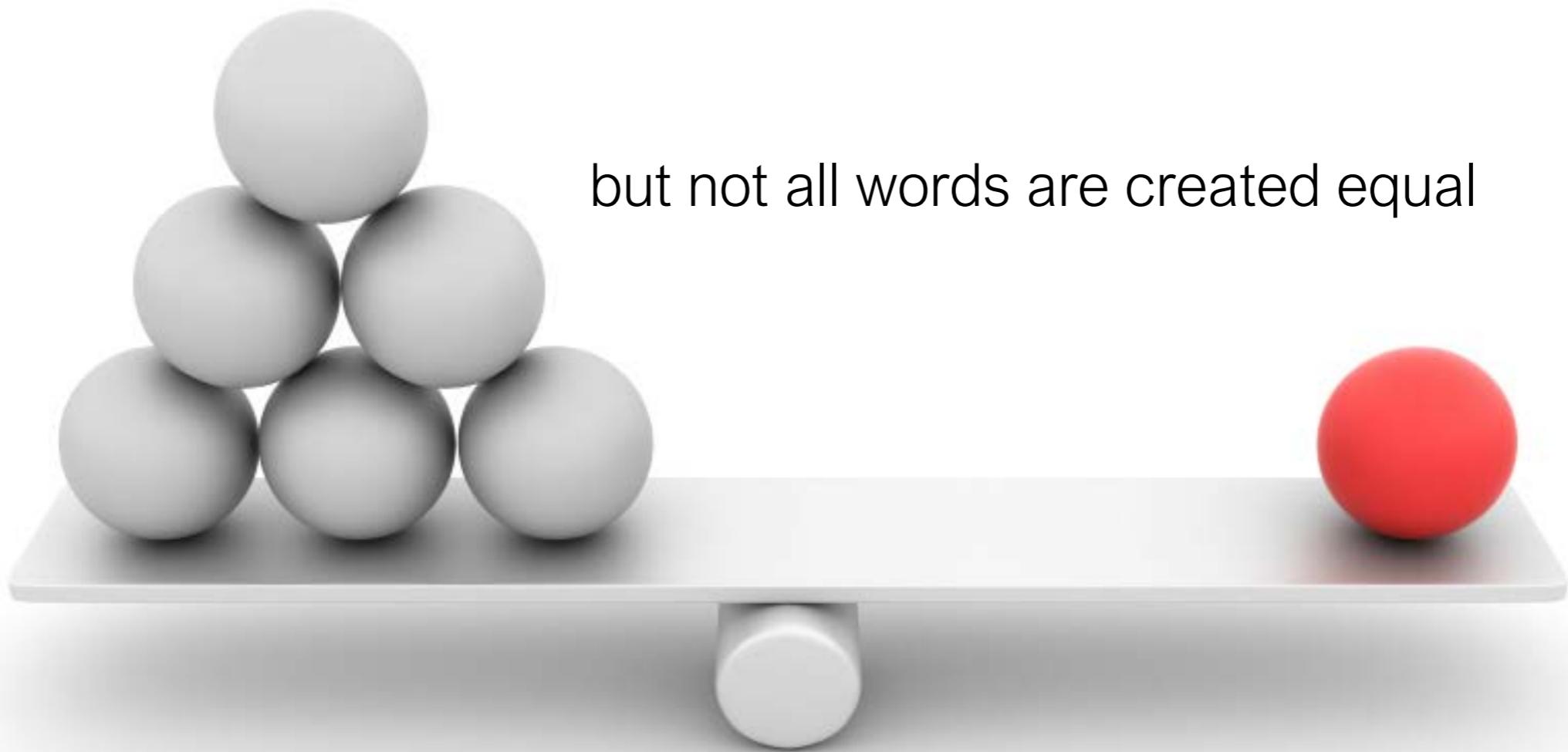
What is the similarity between two documents?



Use any distance you want but the cosine distance is fast.

$$\begin{aligned} d(\mathbf{v}_i, \mathbf{v}_j) &= \cos \theta \\ &= \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|} \end{aligned}$$





but not all words are created equal

TF-IDF

Term Frequency Inverse Document Frequency

$$\mathbf{v}_d = [n(w_{1,d}) \ n(w_{2,d}) \ \cdots \ n(w_{T,d})]$$

weigh each word by a heuristic

$$\mathbf{v}_d = [n(w_{1,d})\alpha_1 \ n(w_{2,d})\alpha_2 \ \cdots \ n(w_{T,d})\alpha_T]$$

$$n(w_{i,d})\alpha_i = n(w_{i,d}) \log \left\{ \frac{D}{\sum_{d'} \mathbf{1}[w_i \in d']} \right\}$$

term frequency inverse document frequency

(down-weights **common** terms)

Standard BOW pipeline

(for image classification)

Dictionary Learning:

Learn Visual Words using clustering

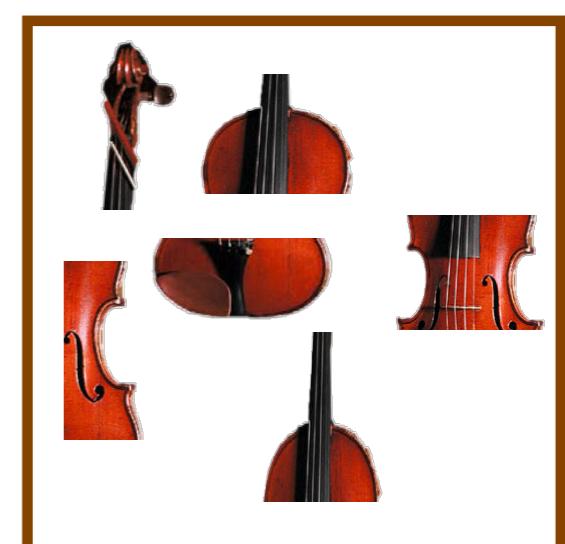
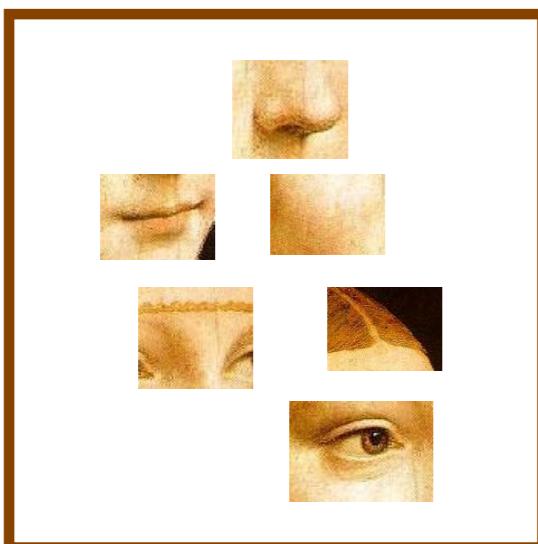
Encode:
build Bags-of-Words (BOW) vectors
for each image

Classify:
Train and test data using BOWs

Dictionary Learning:

Learn Visual Words using clustering

1. extract features (e.g., SIFT) from images



Dictionary Learning:

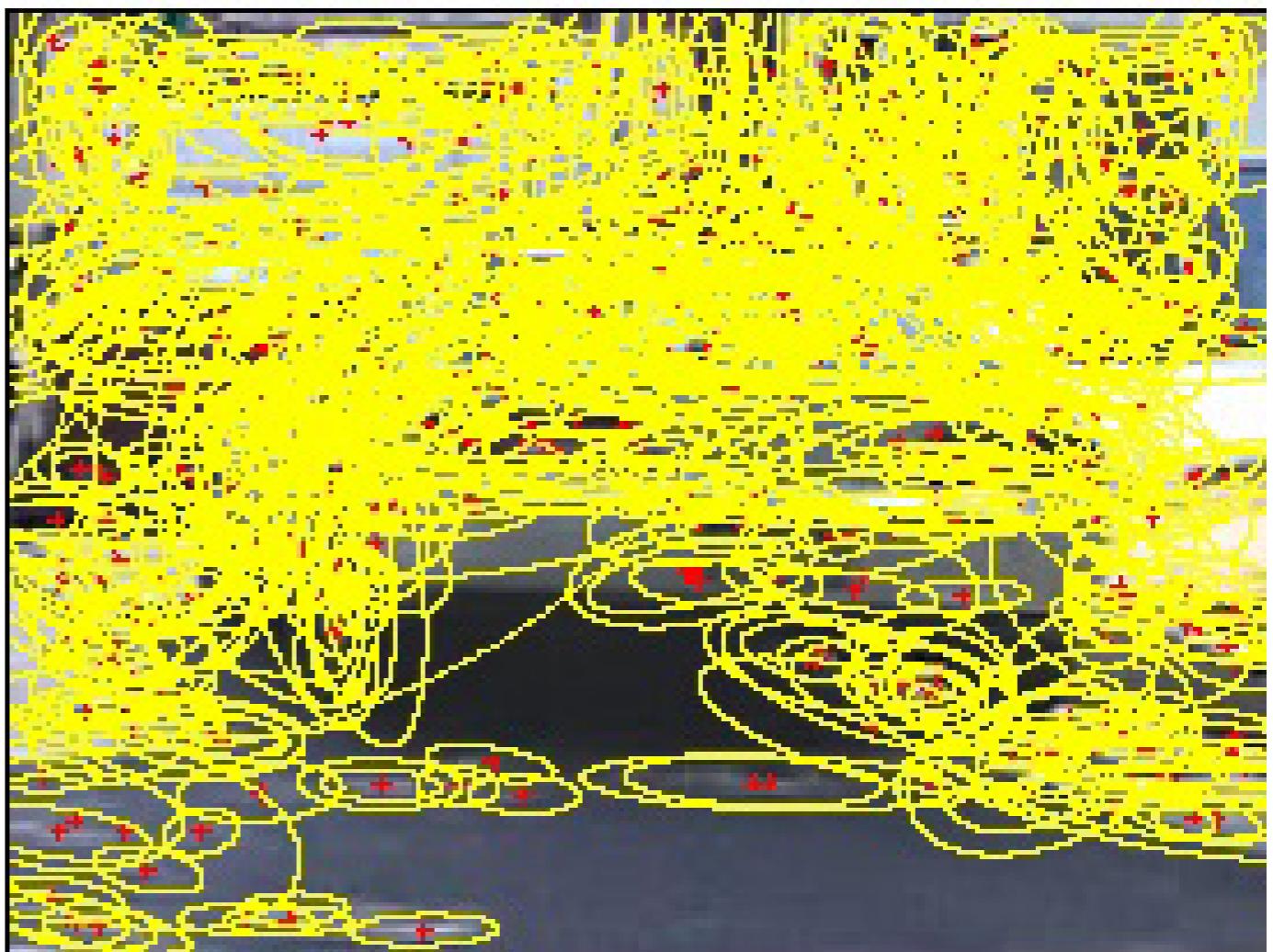
Learn Visual Words using clustering

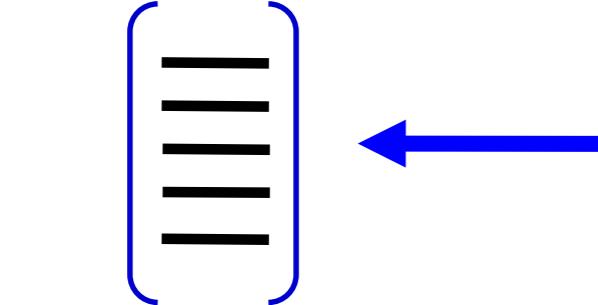
2. Learn visual dictionary (e.g., K-means clustering)



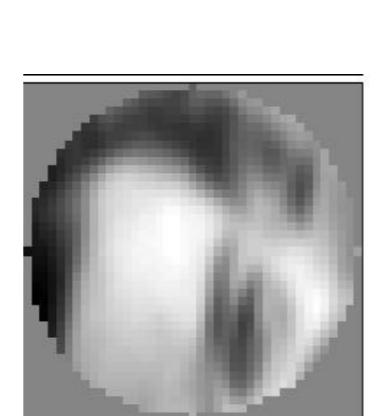
What kinds of features can we extract?

- Regular grid
 - Vogel & Schiele, 2003
 - Fei-Fei & Perona, 2005
- Interest point detector
 - Csurka et al. 2004
 - Fei-Fei & Perona, 2005
 - Sivic et al. 2005
- Other methods
 - Random sampling (Vidal-Naquet & Ullman, 2002)
 - Segmentation-based patches (Barnard et al. 2003)





**Compute SIFT
descriptor**
[Lowe'99]



Normalize patch

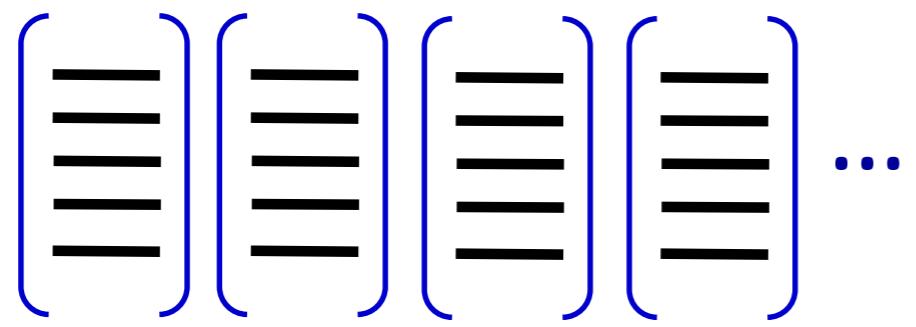


Detect patches

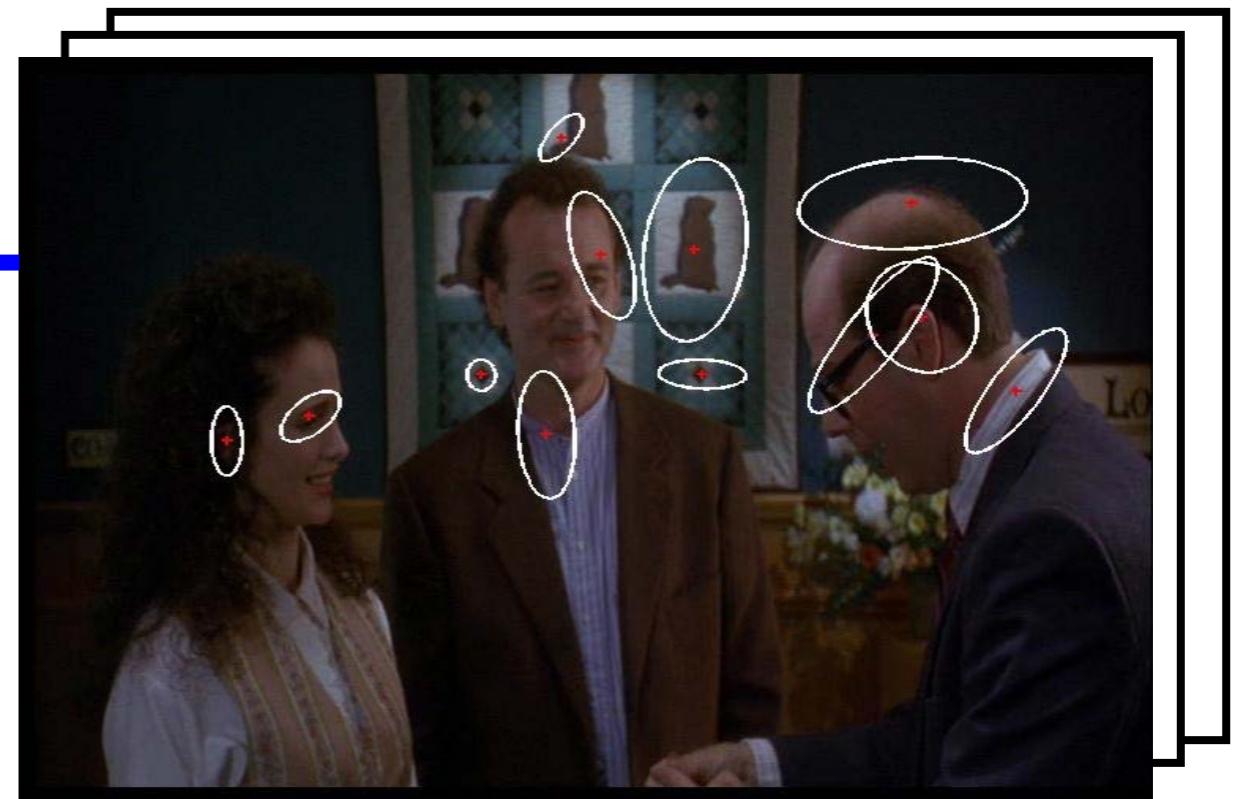
[Mikojaczyk and Schmid '02]

[Mata, Chum, Urban & Pajdla, '02]

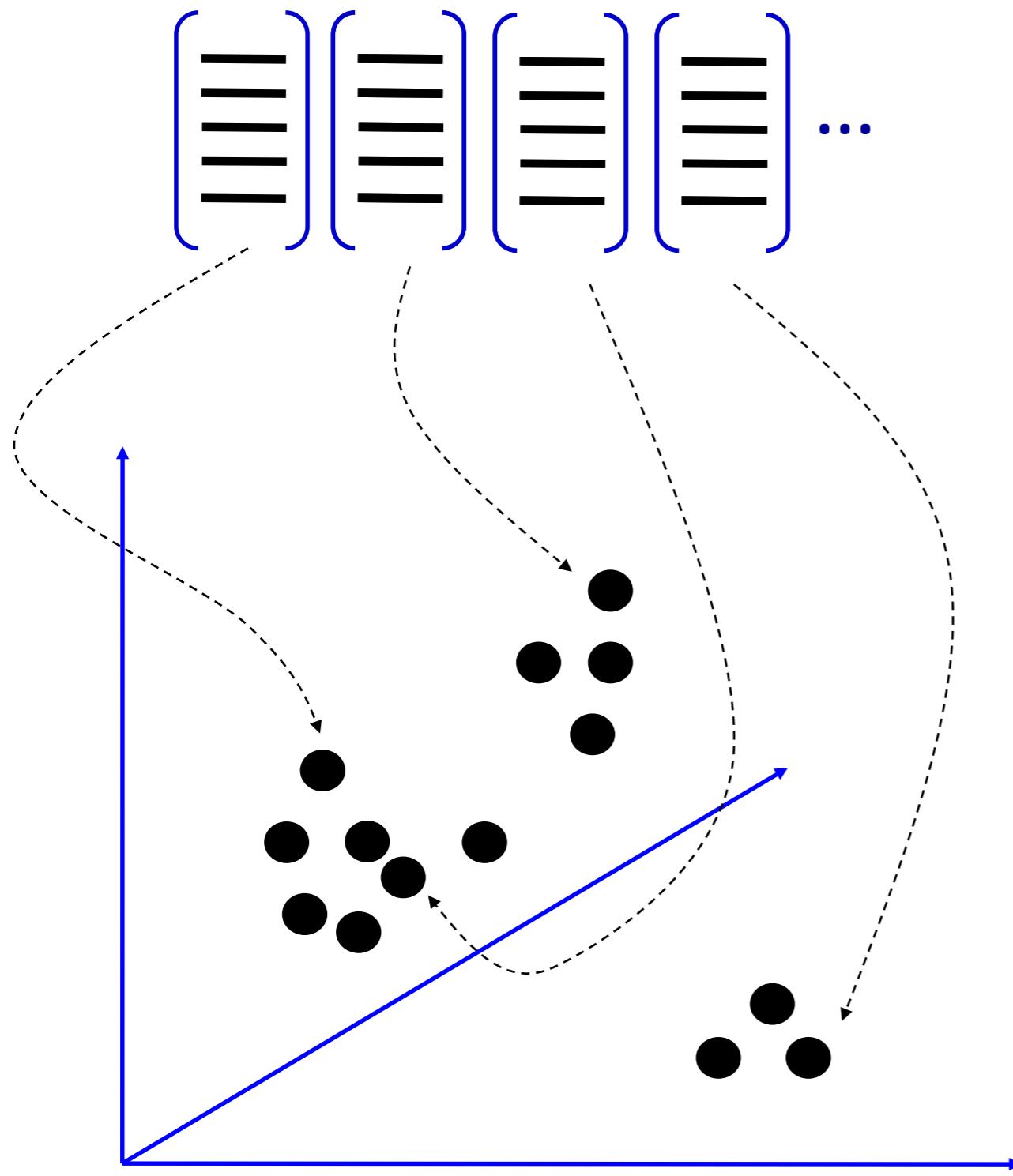
[Sivic & Zisserman, '03]

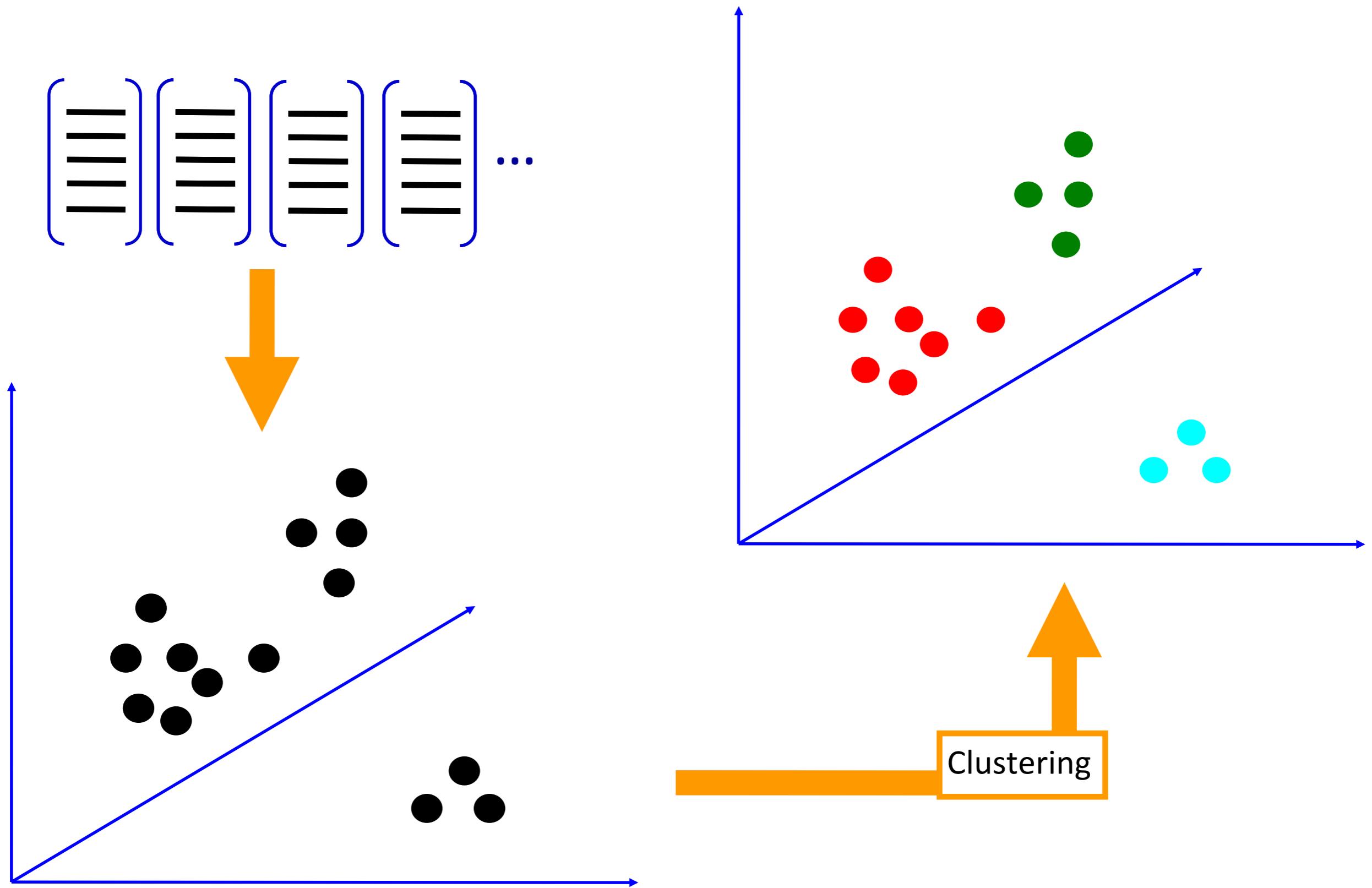


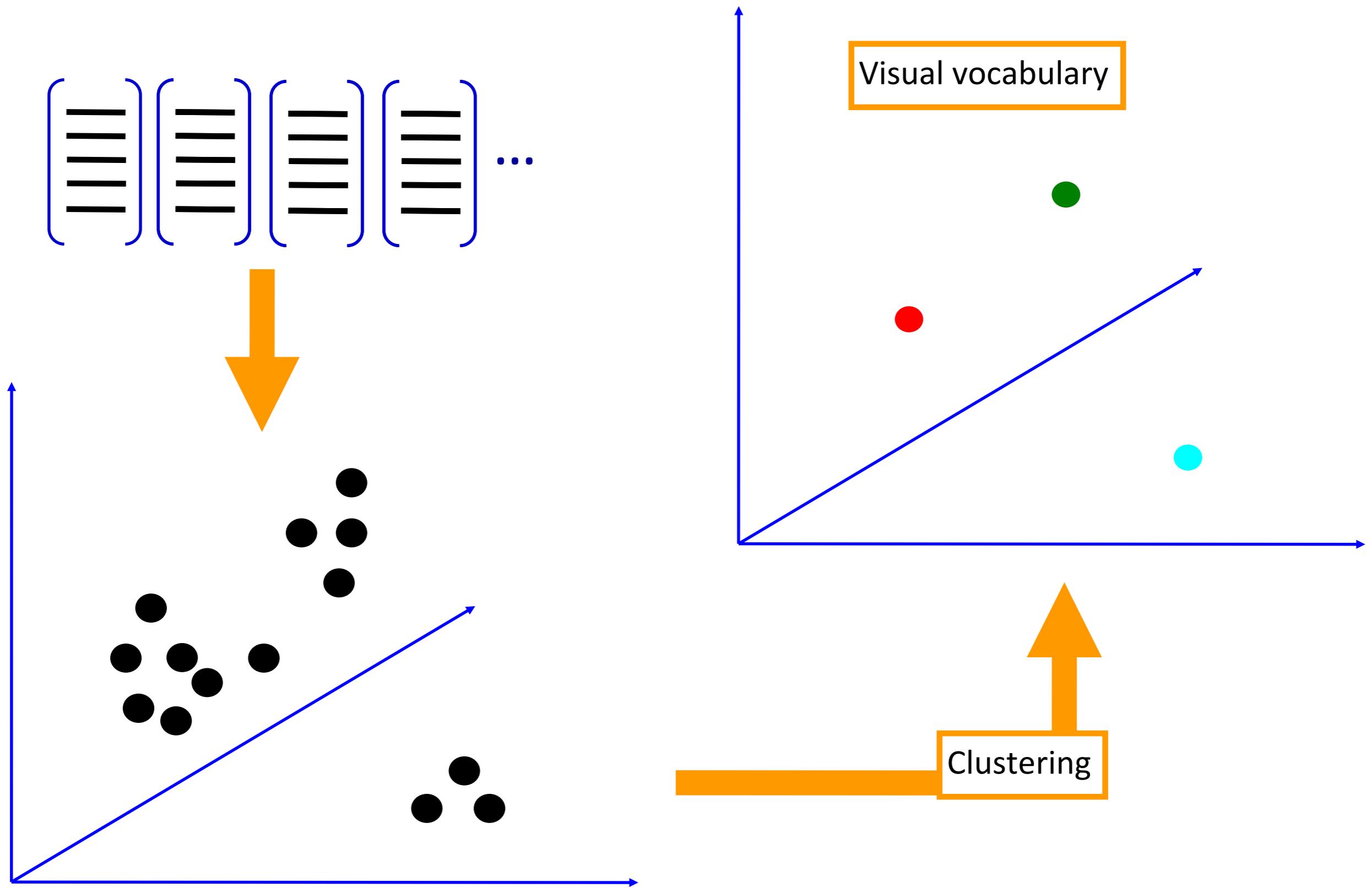
...



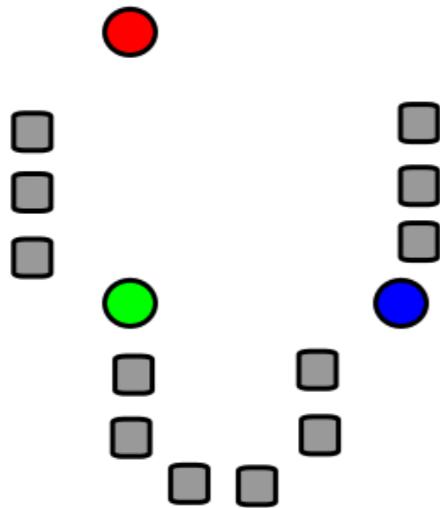
How do we learn the dictionary?



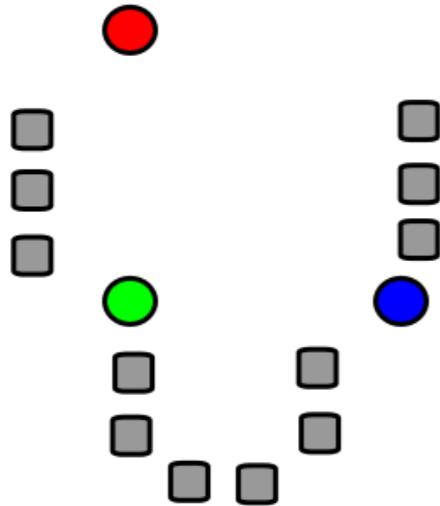




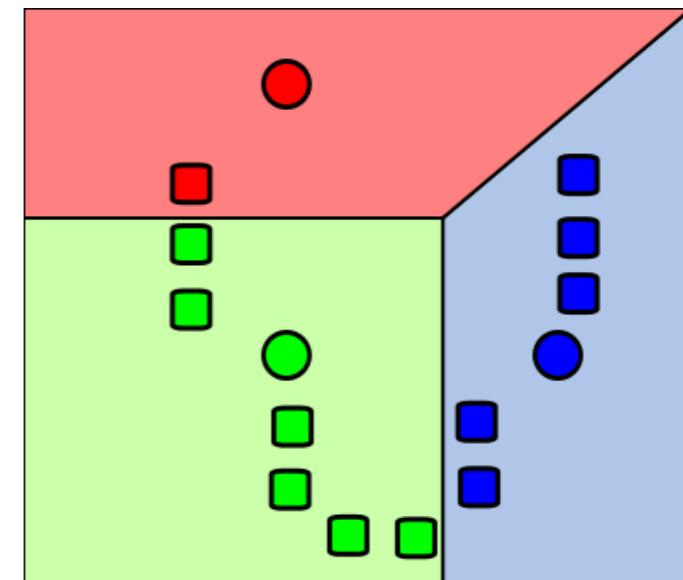
K-means clustering



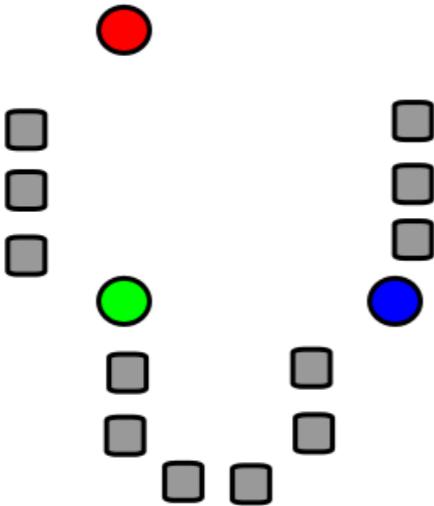
1. Select initial
centroids at random



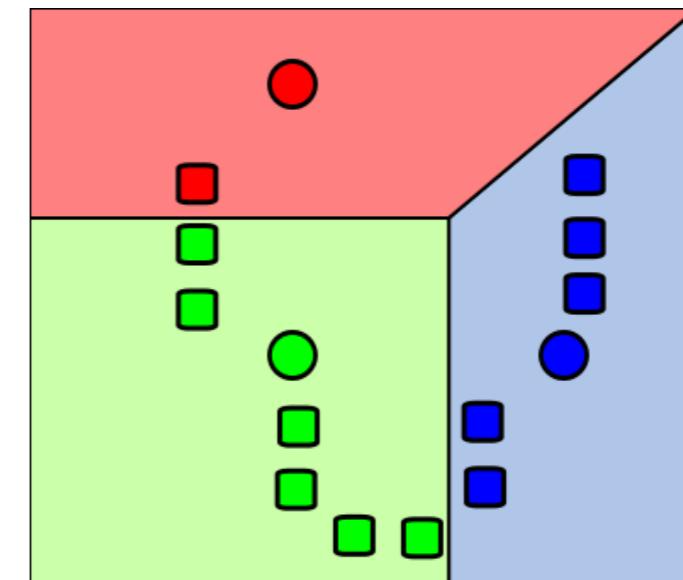
1. Select initial centroids at random



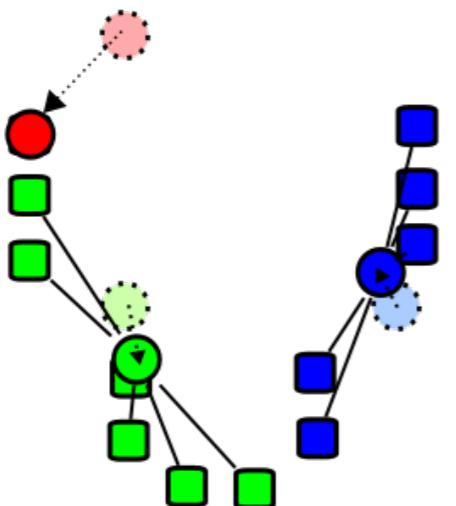
2. Assign each object to the cluster with the nearest centroid.



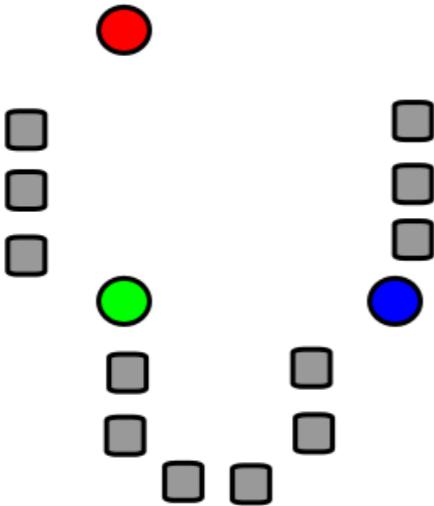
1. Select initial centroids at random



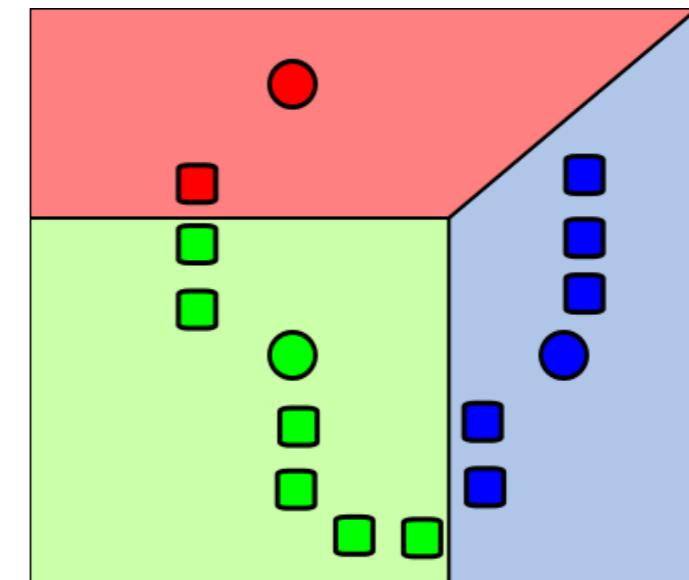
2. Assign each object to the cluster with the nearest centroid.



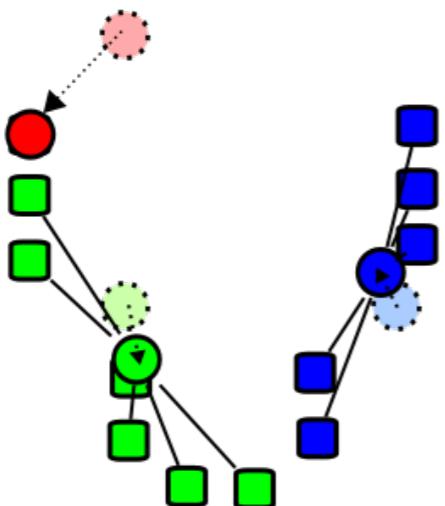
3. Compute each centroid as the mean of the objects assigned to it (go to 2)



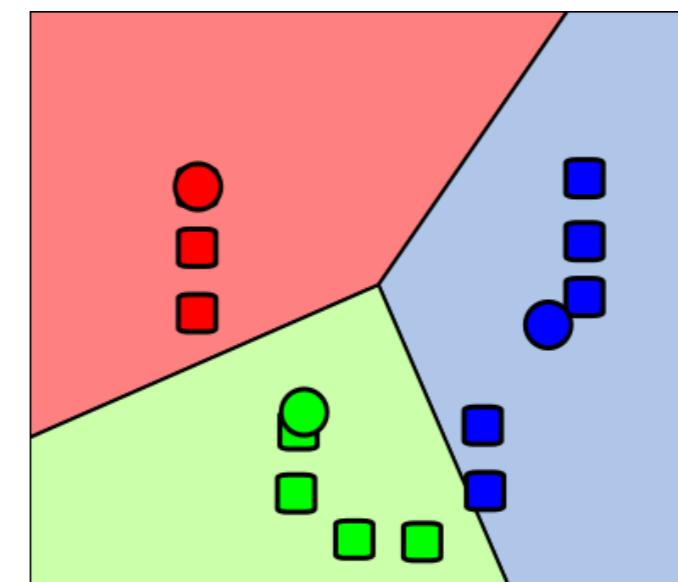
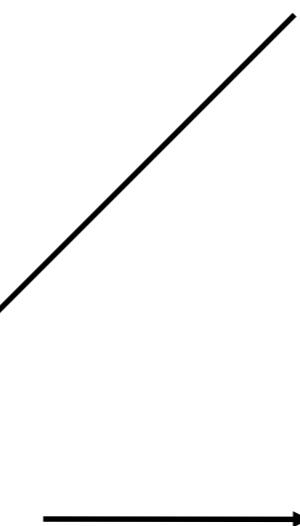
1. Select initial centroids at random



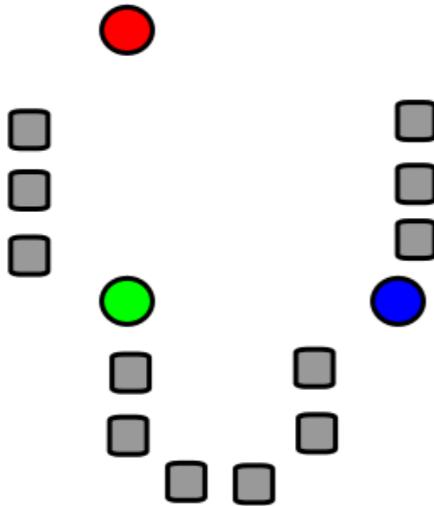
2. Assign each object to the cluster with the nearest centroid.



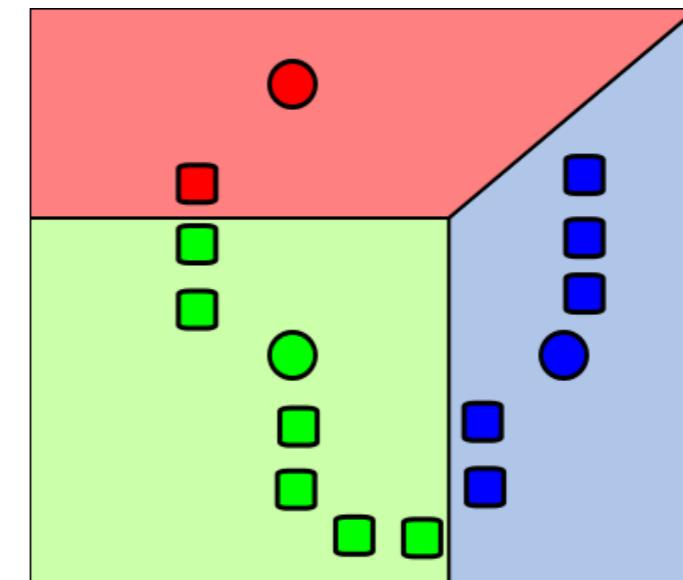
3. Compute each centroid as the mean of the objects assigned to it (go to 2)



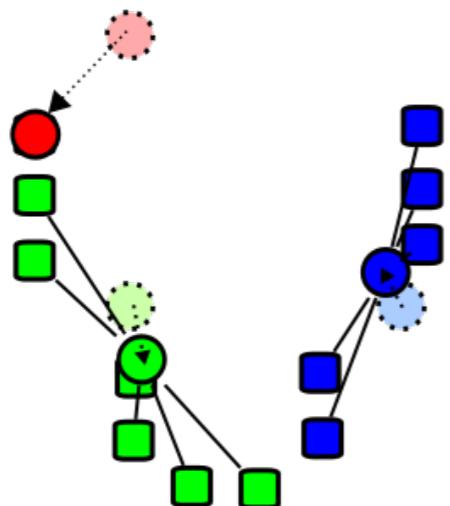
2. Assign each object to the cluster with the nearest centroid.



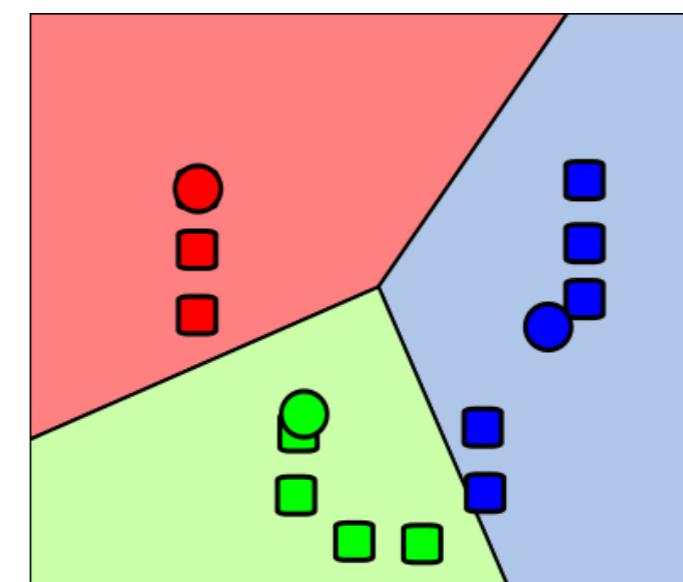
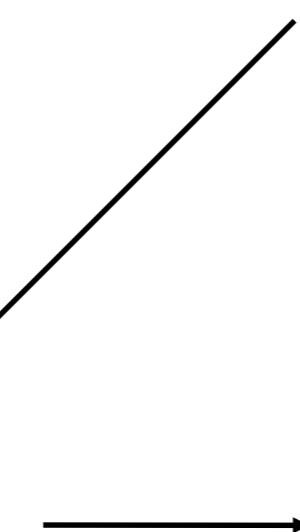
1. Select initial centroids at random



2. Assign each object to the cluster with the nearest centroid.



3. Compute each centroid as the mean of the objects assigned to it (go to 2)



2. Assign each object to the cluster with the nearest centroid.

Repeat previous 2 steps until no change

K-means Clustering

Given k:

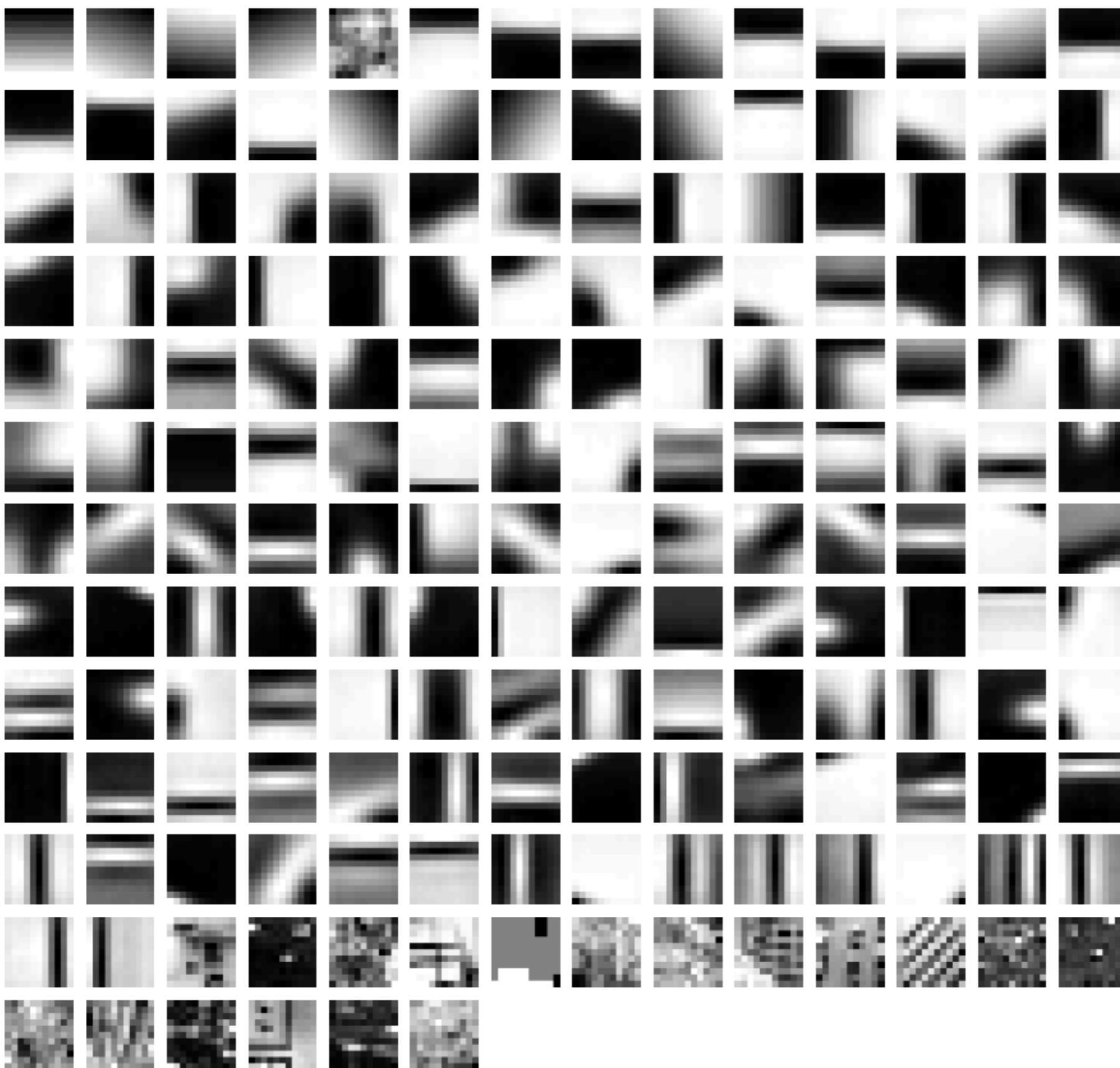
1. Select initial centroids at random.
2. Assign each object to the cluster with the nearest centroid.
3. Compute each centroid as the mean of the objects assigned to it.
4. Repeat previous 2 steps until no change.

*From what **data** should I learn the dictionary?*

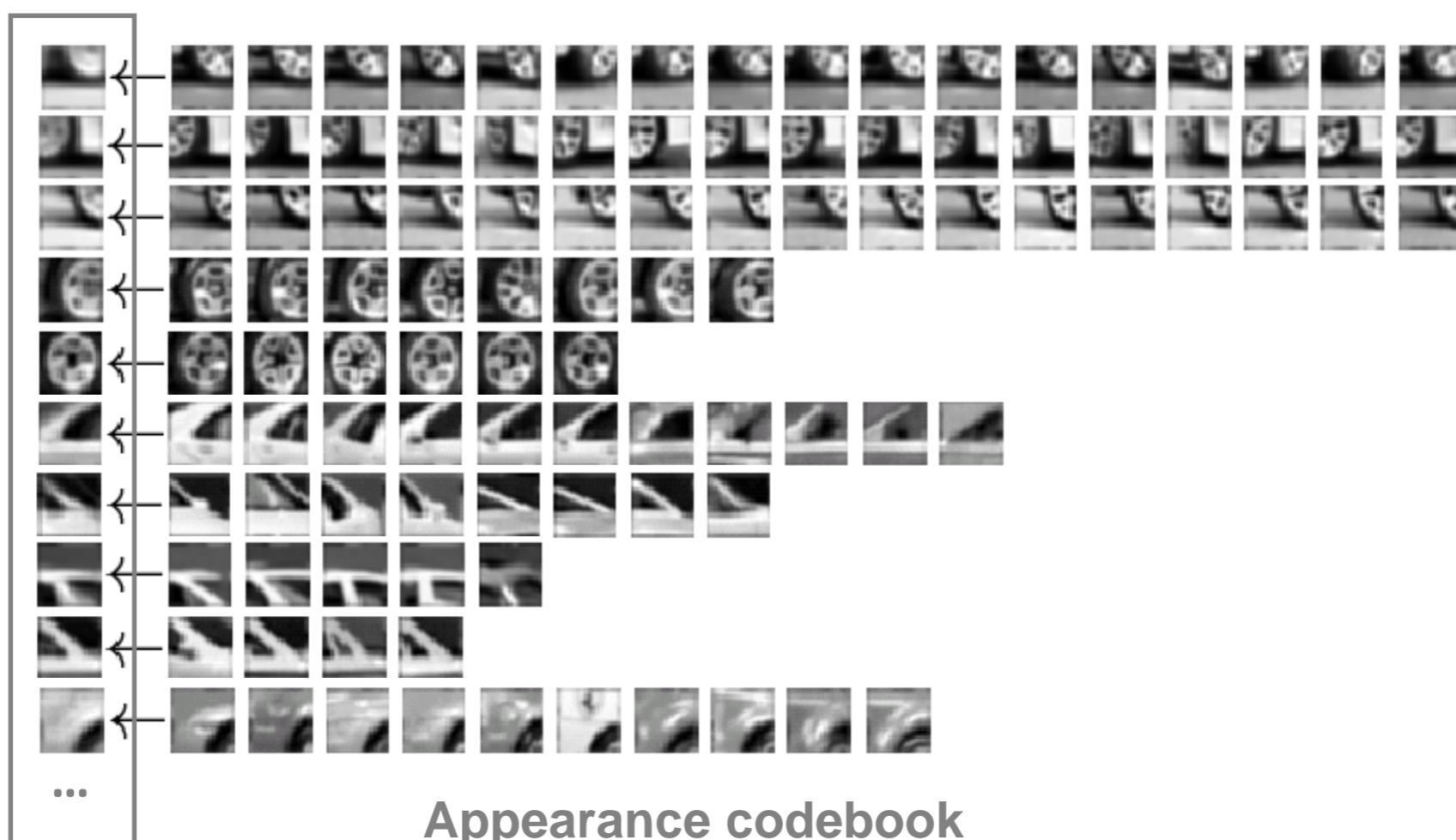
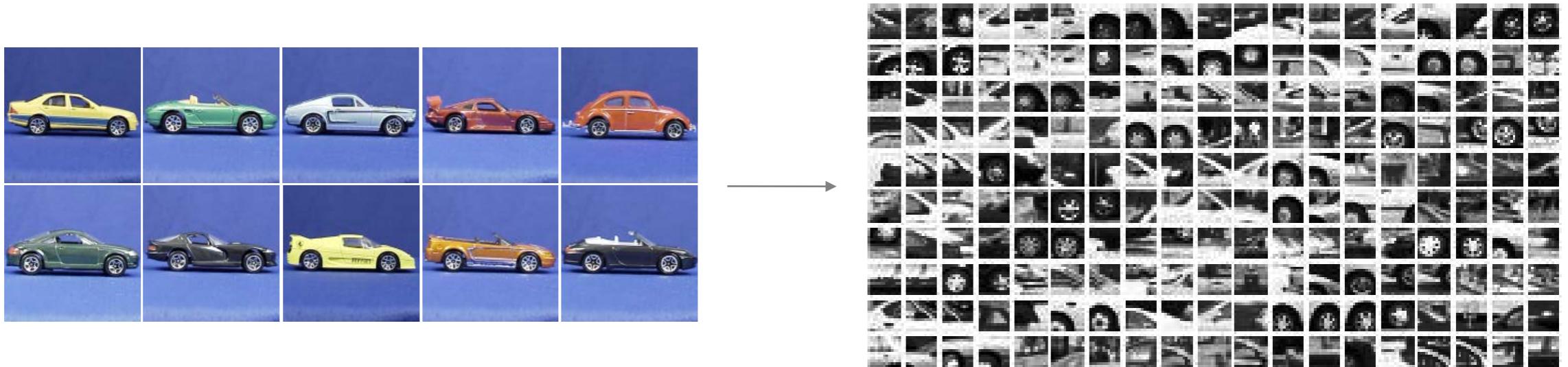
*From what **data** should I learn the dictionary?*

- Dictionary can be learned on separate training set
- Provided the training set is sufficiently representative, the dictionary will be “universal”

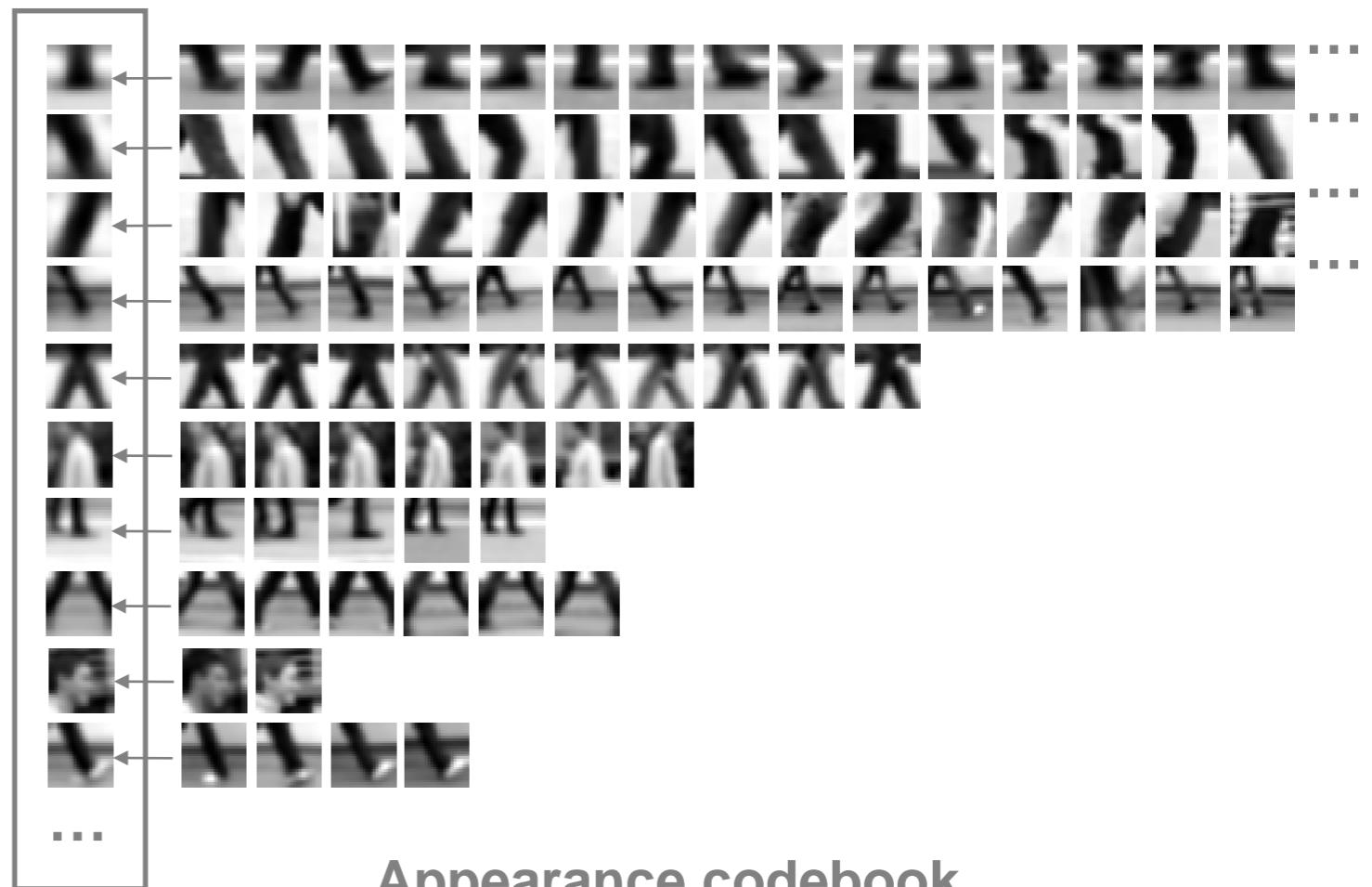
Example visual dictionary



Example dictionary



Another dictionary



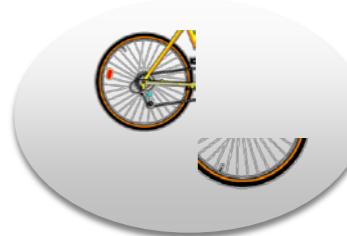
Appearance codebook

Dictionary Learning:

Learn Visual Words using clustering

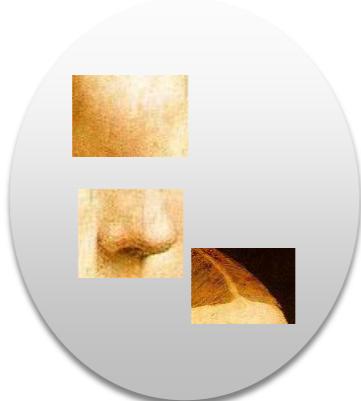
Encode:
build Bags-of-Words (BOW) vectors
for each image

Classify:
Train and test data using BOWs



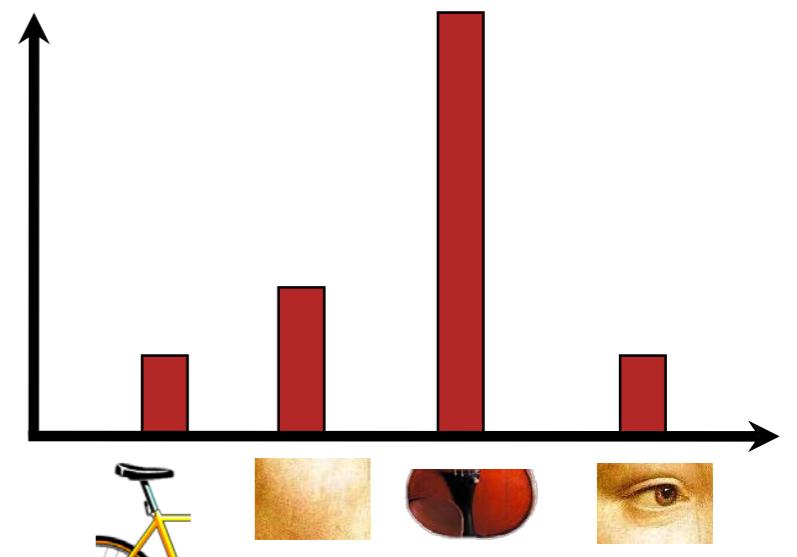
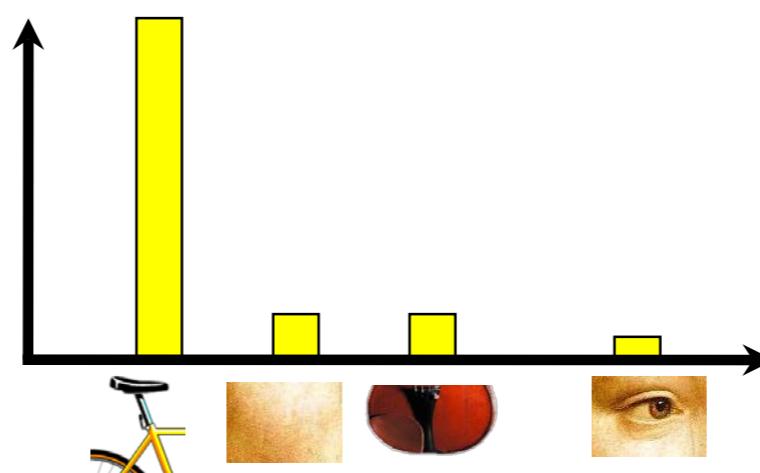
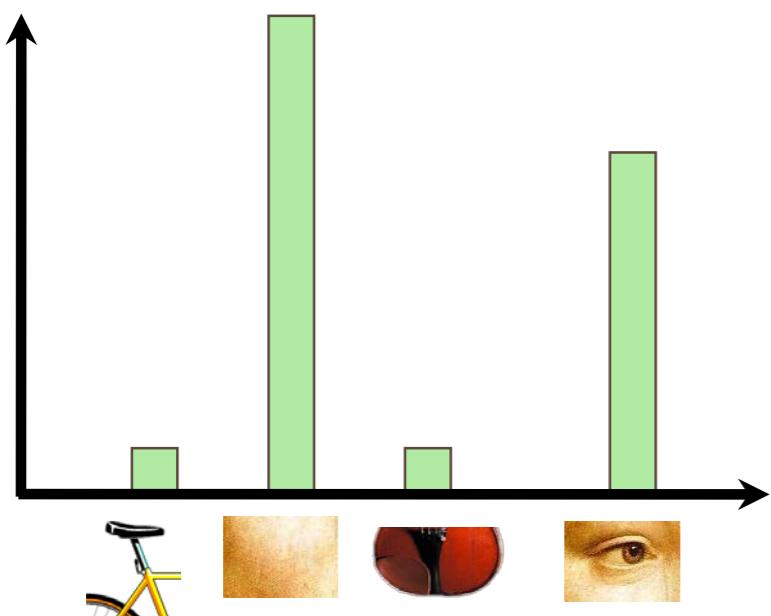
1. Quantization: image features gets associated to a visual word (nearest cluster center)

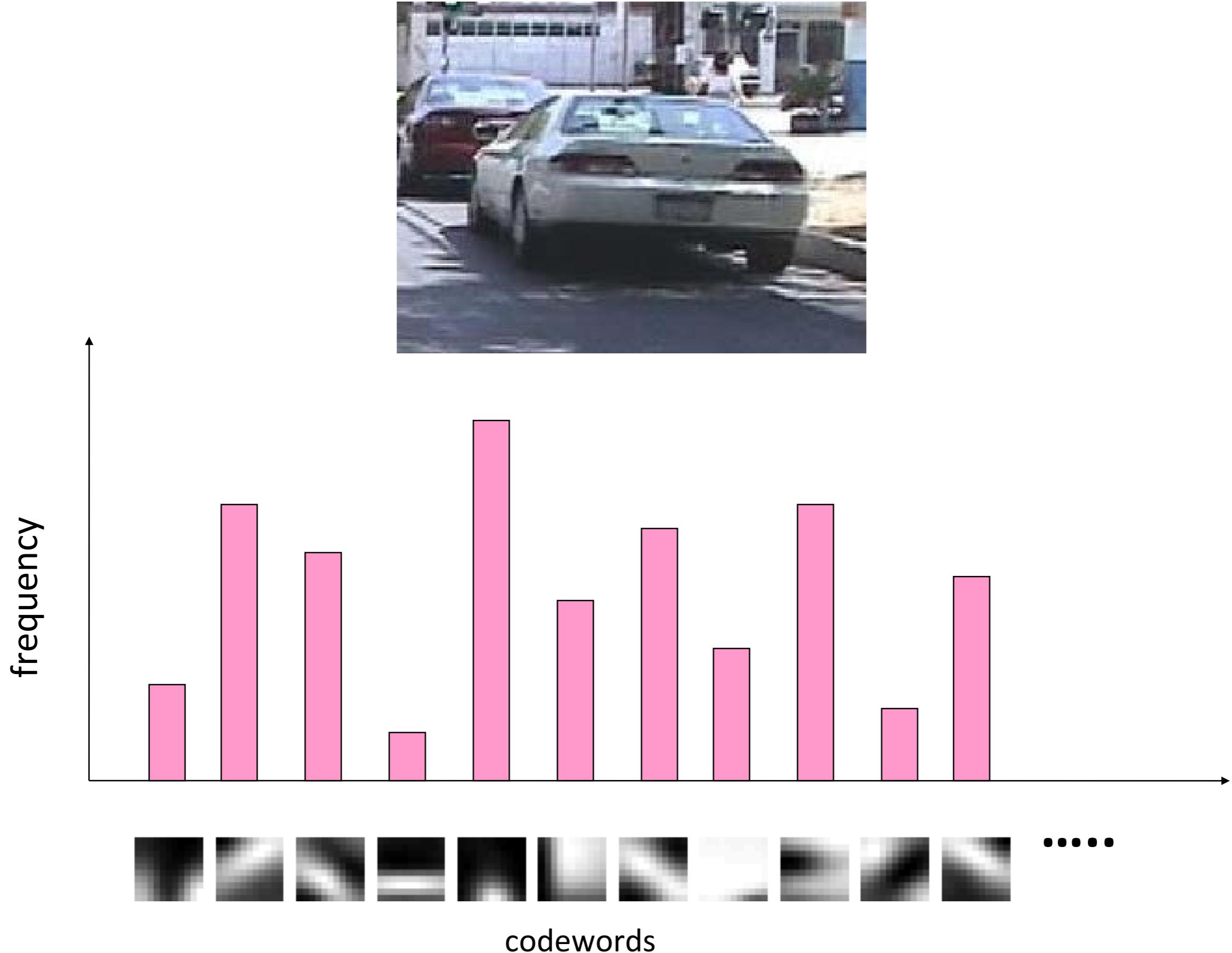
Encode:
build Bags-of-Words (BOW) vectors
for each image



Encode:
build Bags-of-Words (BOW) vectors
for each image

2. Histogram: count the
number of visual word
occurrences



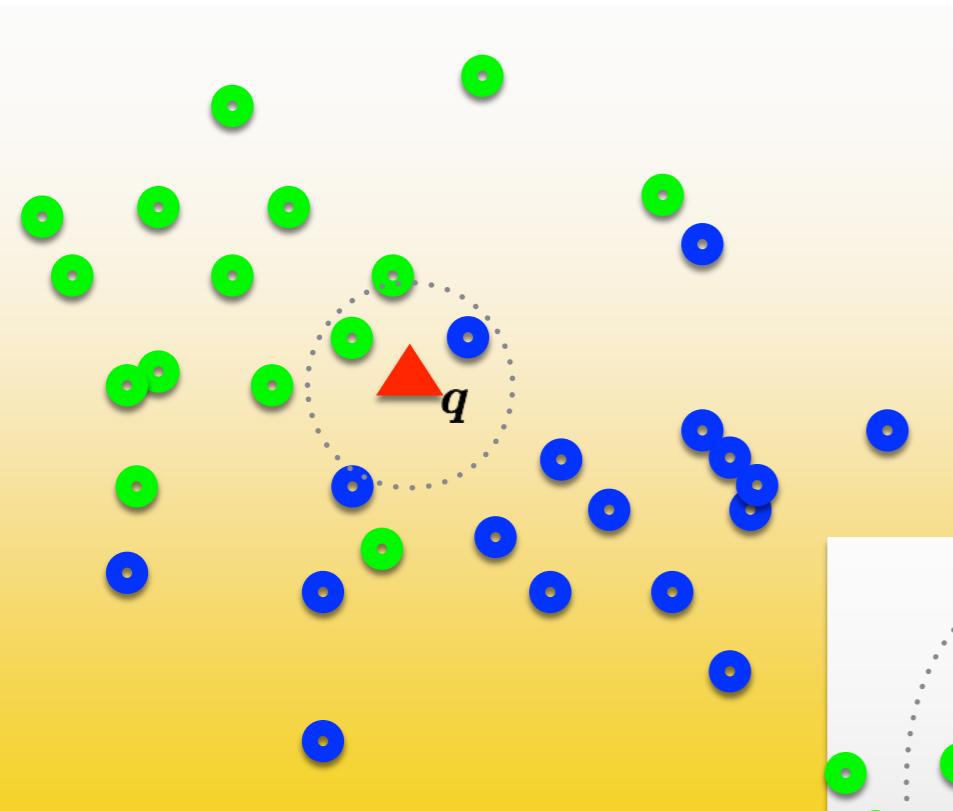


Dictionary Learning:

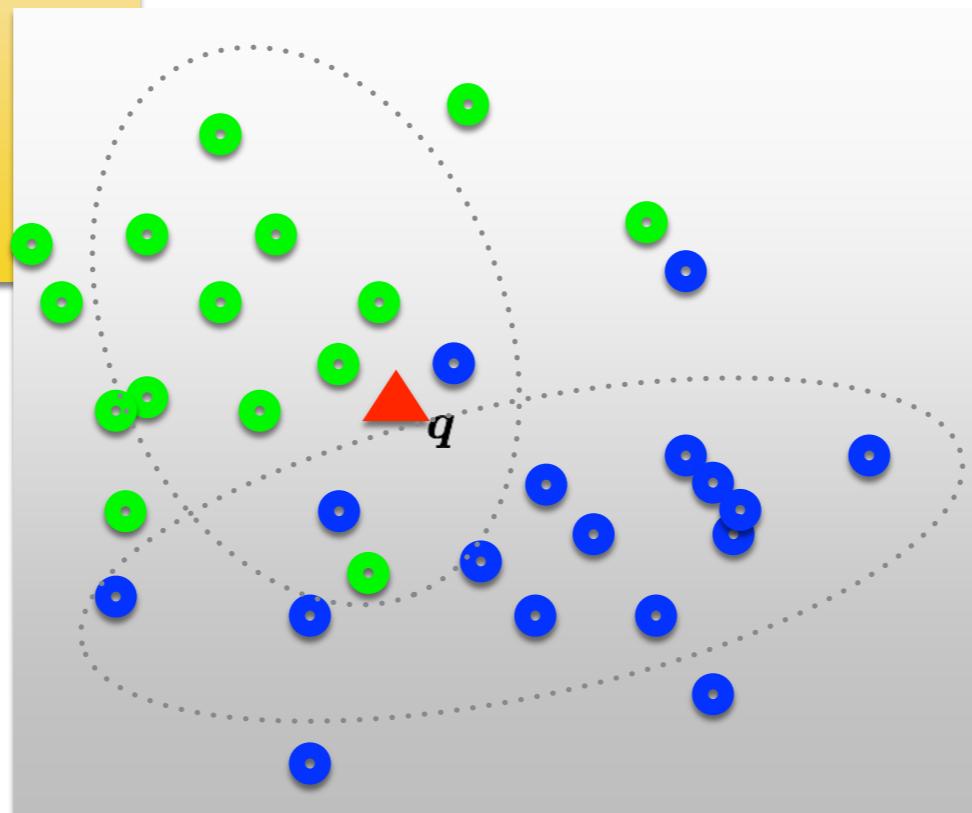
Learn Visual Words using clustering

Encode:
build Bags-of-Words (BOW) vectors
for each image

Classify:
Train and test data using BOWs

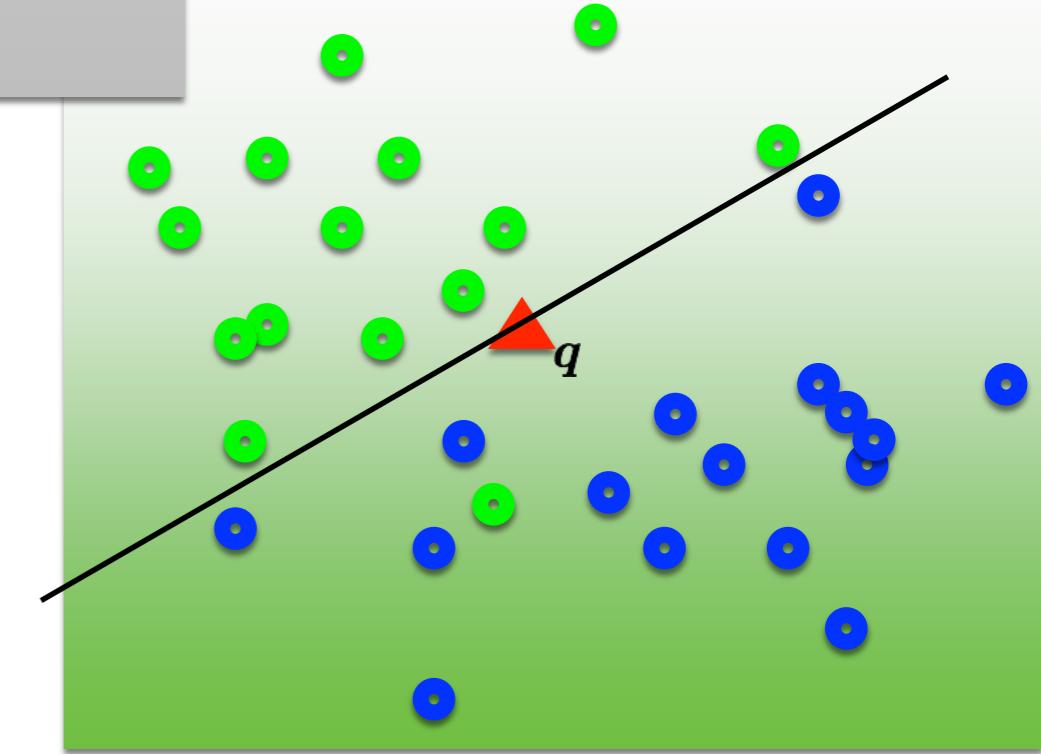


K nearest neighbors



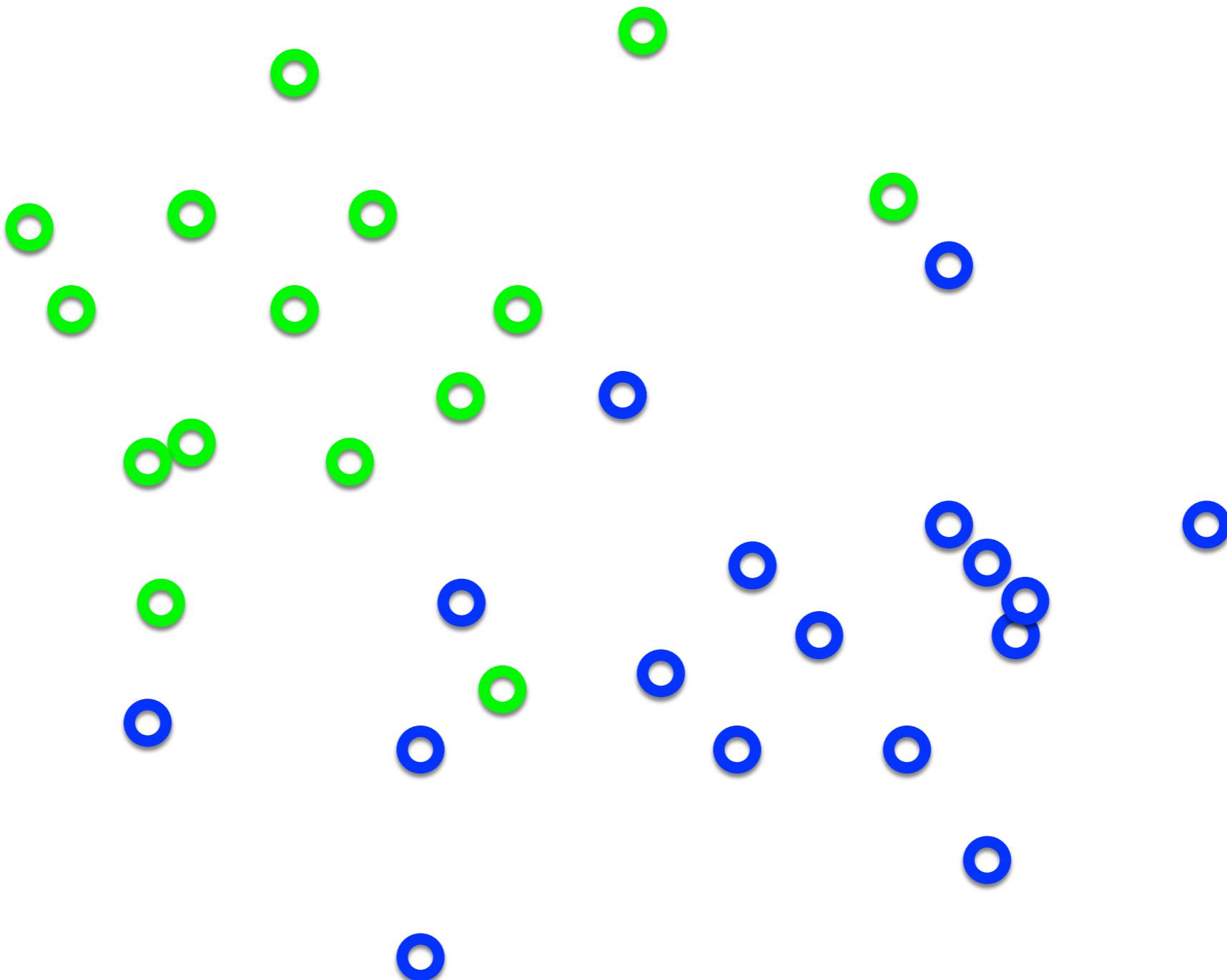
Naïve Bayes

Support Vector Machine

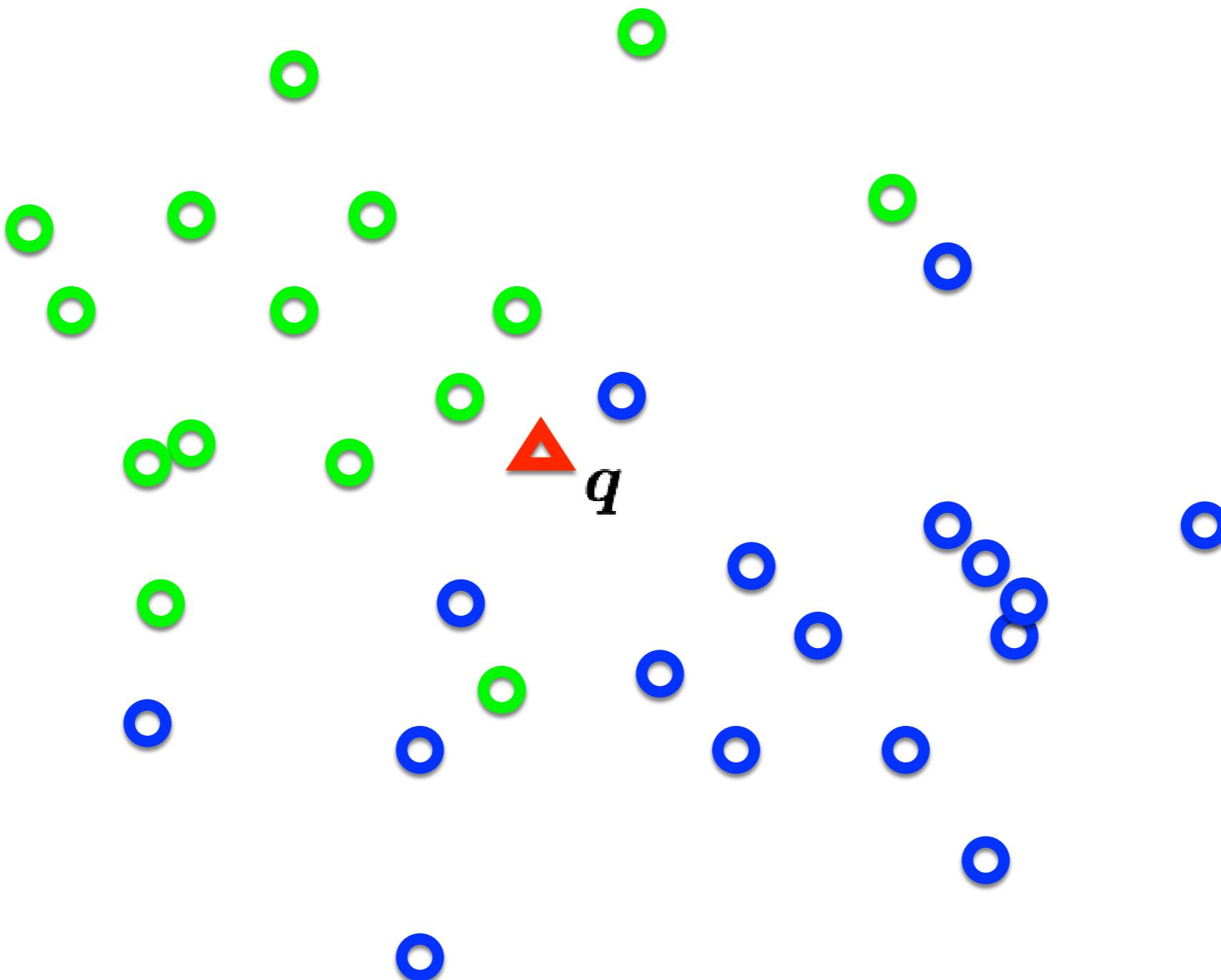


K nearest neighbors

Distribution of data from two classes

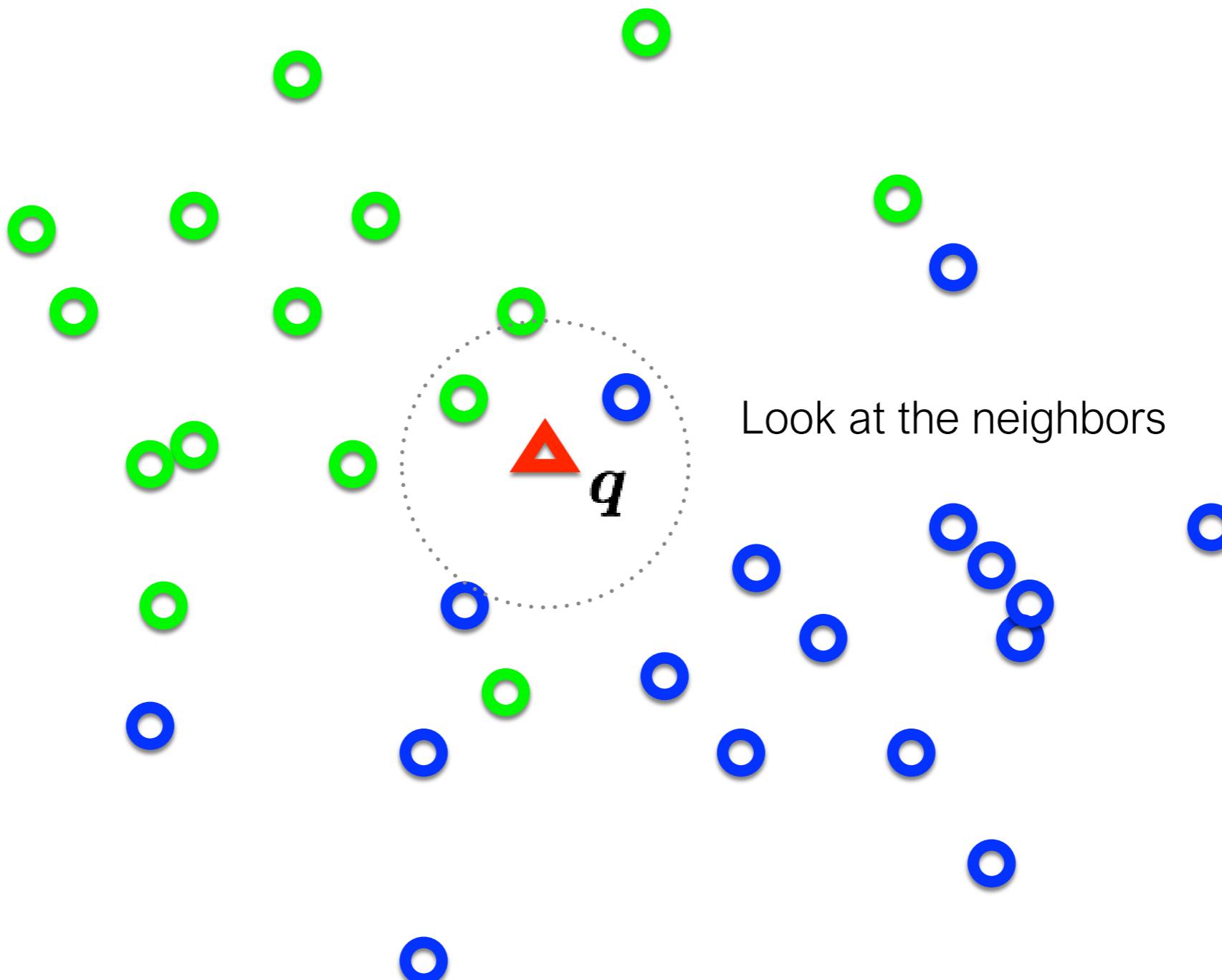


Distribution of data from two classes

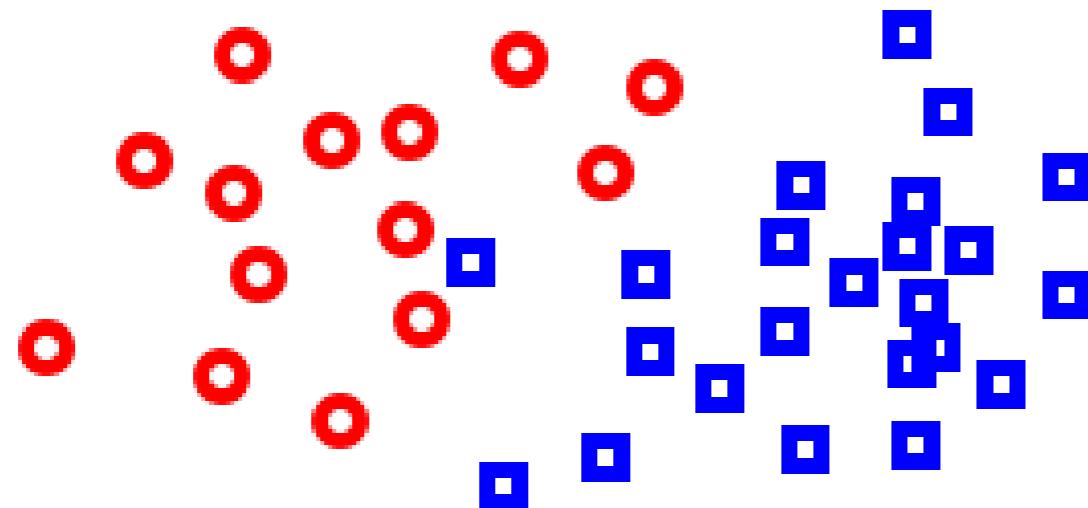


Which class does q belong to?

Distribution of data from two classes



K-Nearest Neighbor (KNN) Classifier



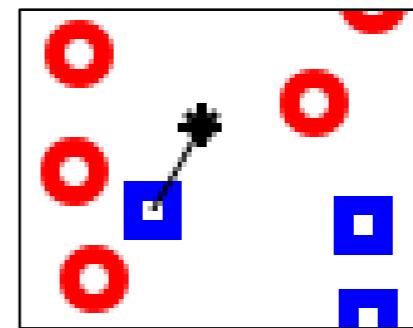
For a given query point q , assign the class of the nearest neighbor

Compute the k nearest neighbors and assign the class by majority vote.

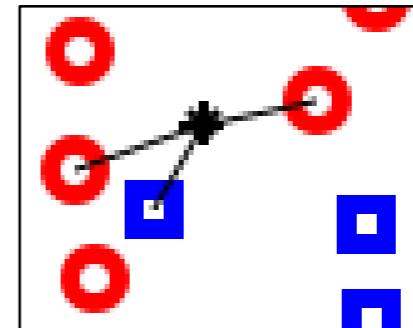
Non-parametric pattern classification approach

Consider a two class problem where each sample consists of two measurements (x,y) .

$k = 1$



$k = 3$



Nearest Neighbor is competitive

40281508803272726475529284686500876171127400776386420940578274711366
50711167679664143112241082634006330117113109975414895351982339901029
846868246793394314470596044461233645968568641865284554770782237018
76953465018828357808571101378507110114527623028596972136418240510226
93477149064842728100783331376131665747595849918501320348220251514889
82049962335648092836457294912840709116759914592504108908989425198980
35517216919955162286714604033223689853854520563283995794671313660901
94368160413174951001162198403649071657525185470670258104571851900607
8857389886823975629288168879180172075190209862393802111142972512199
148534347507488153959769036398212868553941251514414435912233029009
931909754920105149336152522026601203025579580895032590884884546549
6928545799216340783934656219260061287982047750564674307507420899404
12845278113035703193631773084826529739099642972116747590821445161325
90666367728608302983253980019513960141712379749939282718091017796999
2101045282835178112978405078847858498138031795516574935471208160734
28308784084458566309376893495891288681379011470817457121139621280766
41992780136134111560707232522949812161274000822922799275134941856283

MNIST Digit Recognition

- Handwritten digits
- 28x28 pixel images: $d = 784$
- 60,000 training samples
- 10,000 test samples

Yann LeCunn

	Test Error Rate (%)
Linear classifier (1-layer NN)	12.0
K-nearest-neighbors, Euclidean	5.0
K-nearest-neighbors, Euclidean, deskewed	2.4
K-NN, Tangent Distance, 16x16	1.1
K-NN, shape context matching	0.67
1000 RBF + linear classifier	3.6
SVM deg 4 polynomial	1.1
2-layer NN, 300 hidden units	4.7
2-layer NN, 300 HU, [deskewing]	1.6
LeNet-5, [distortions]	0.8
Boosted LeNet-4, [distortions]	0.7

What is the best distance metric between data points?

- Typically Euclidean distance
- Locality sensitive distance metrics
- Important to normalize.
Dimensions have different scales

How many K?

- Typically $k=1$ is good
- Cross-validation (try different k !)

Distance metrics

$$D(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_N - y_N)^2} \quad \text{Euclidean}$$

$$D(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} = \frac{x_1 y_1 + \cdots + x_N y_N}{\sqrt{\sum_n x_n^2} \sqrt{\sum_n y_n^2}} \quad \text{Cosine}$$

$$D(\mathbf{x}, \mathbf{y}) = \frac{1}{2} \sum_n \frac{(x_n - y_n)^2}{(x_n + y_n)} \quad \text{Chi-squared}$$

Choice of distance metric

- Hyperparameter

L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

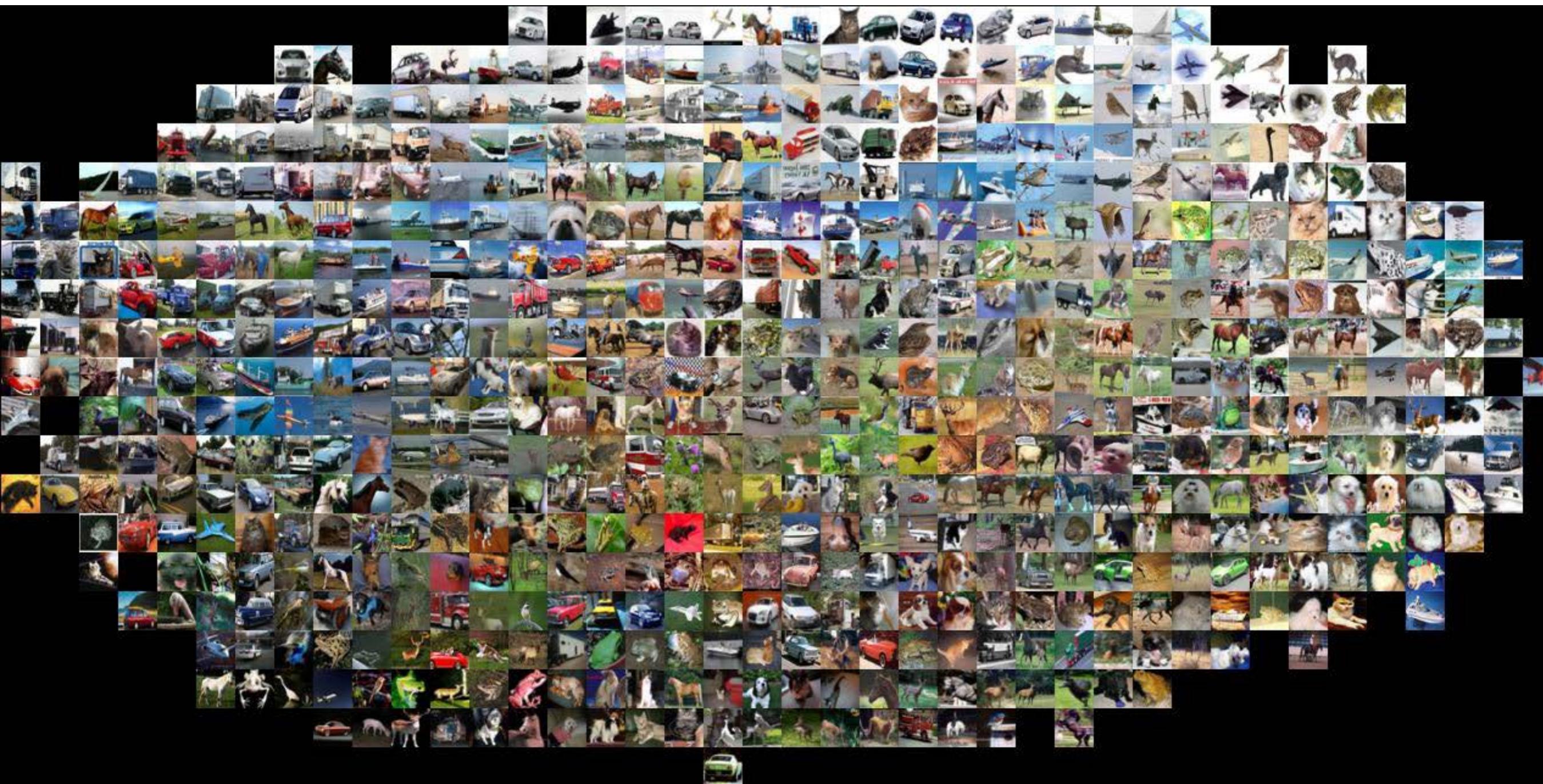
L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

- Two most commonly used special cases of p-norm

$$\|x\|_p = \left(|x_1|^p + \cdots + |x_n|^p \right)^{\frac{1}{p}} \quad p \geq 1, x \in \mathbb{R}^n$$

Visualization: L2 distance



CIFAR-10 and NN results

Example dataset: **CIFAR-10**

10 labels

50,000 training images

10,000 test images.

airplane



automobile



bird



cat



deer



dog



frog



horse



ship



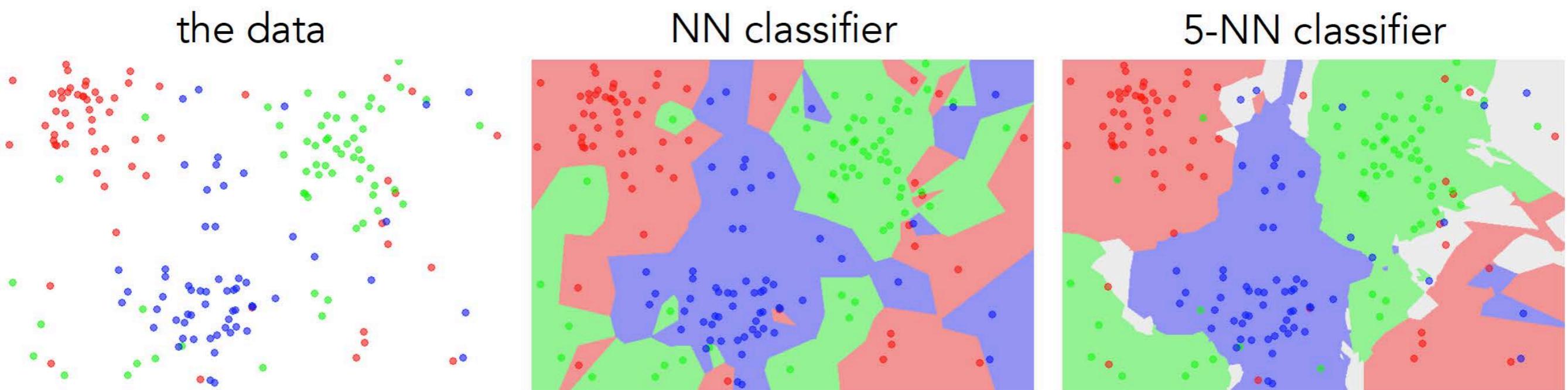
truck

For every test image (first column),
examples of nearest neighbors in rows



k-nearest neighbor

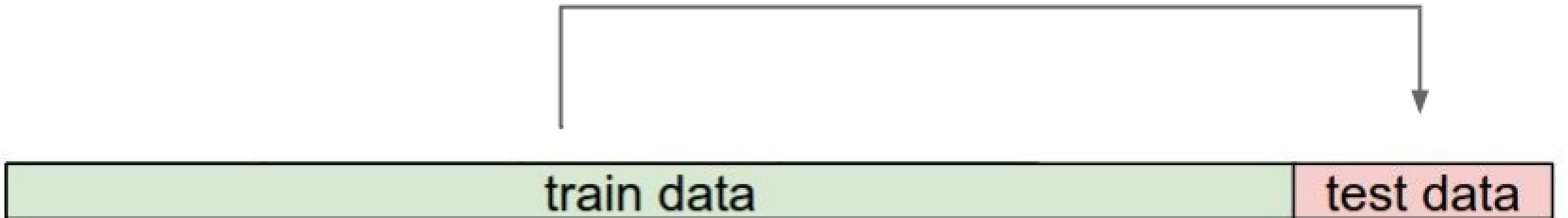
- Find the k closest points from training data
- Labels of the k points “vote” to classify



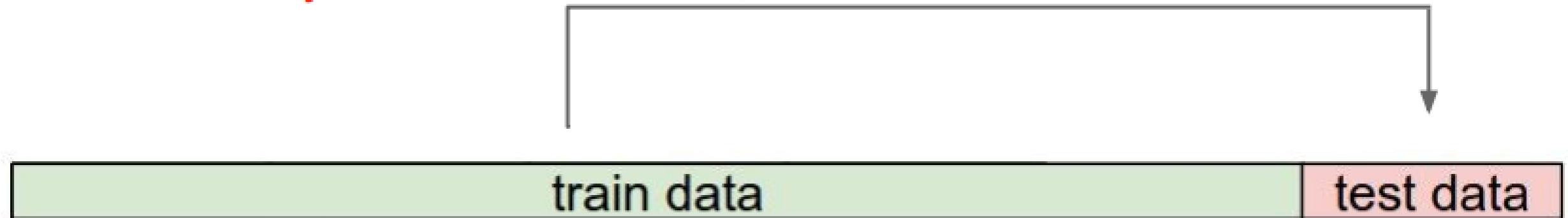
Hyperparameters

- What is the best distance to use?
- What is the best value of k to use?
- i.e., how do we set the hyperparameters?
- Very problem-dependent
- Must try them all and see what works best

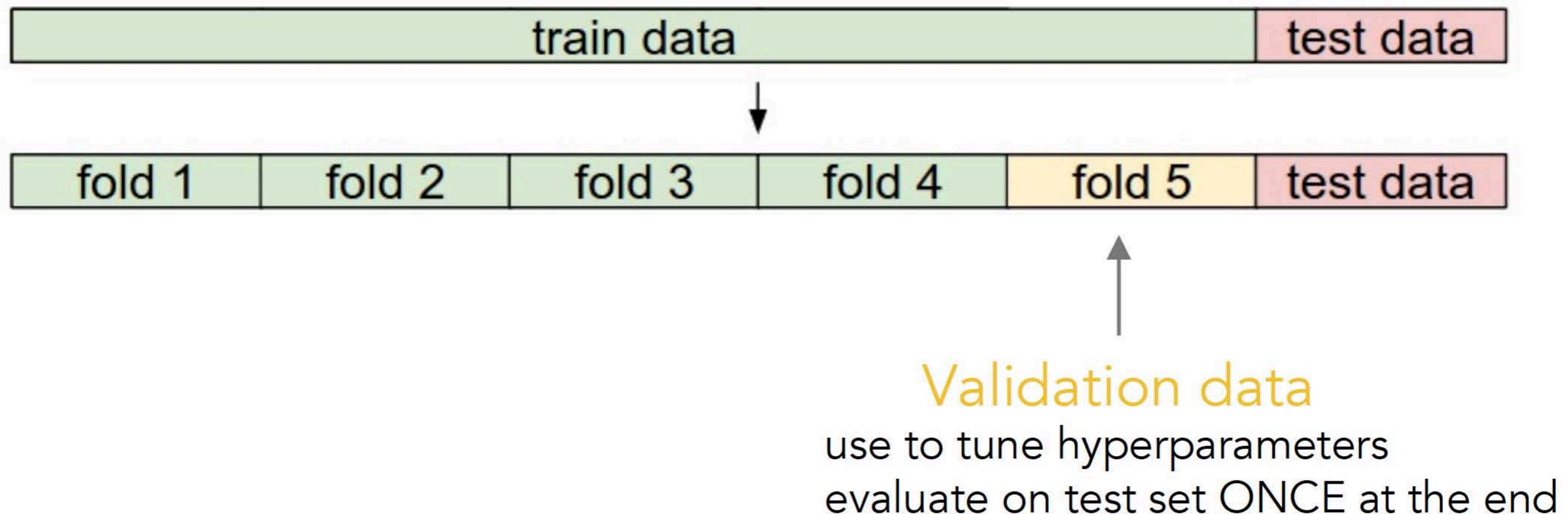
Try out what hyperparameters work best on test set.



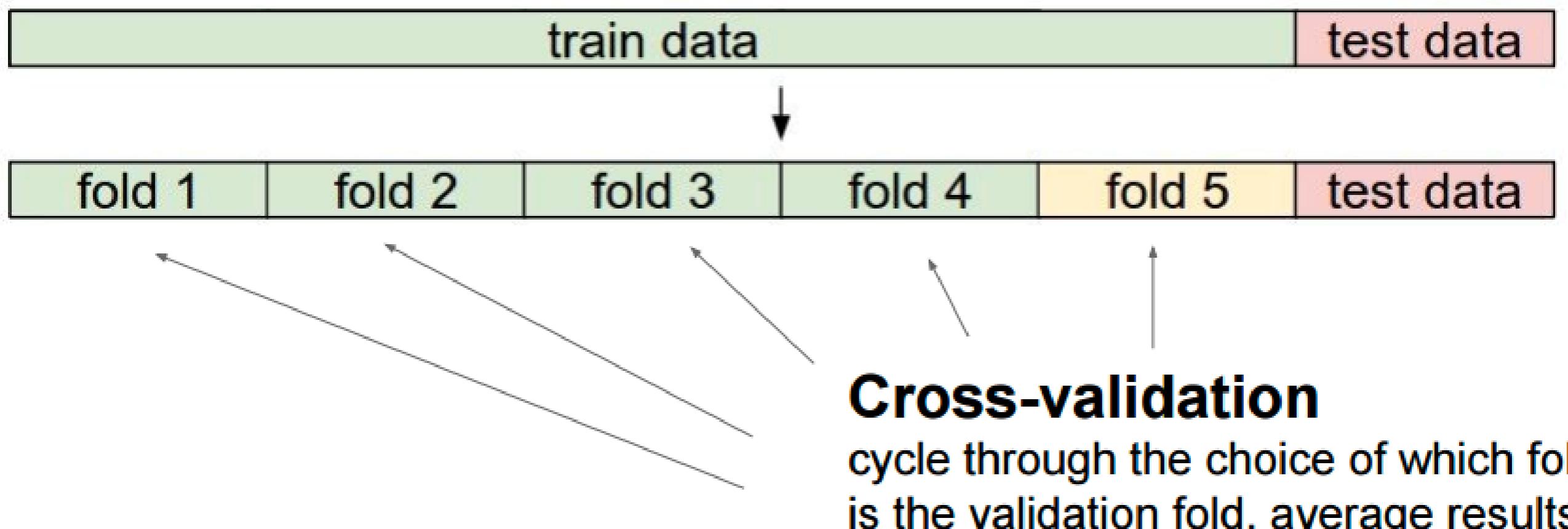
Trying out what hyperparameters work best on test set:
Very bad idea. The test set is a proxy for the generalization performance!
Use only **VERY SPARINGLY**, at the end.

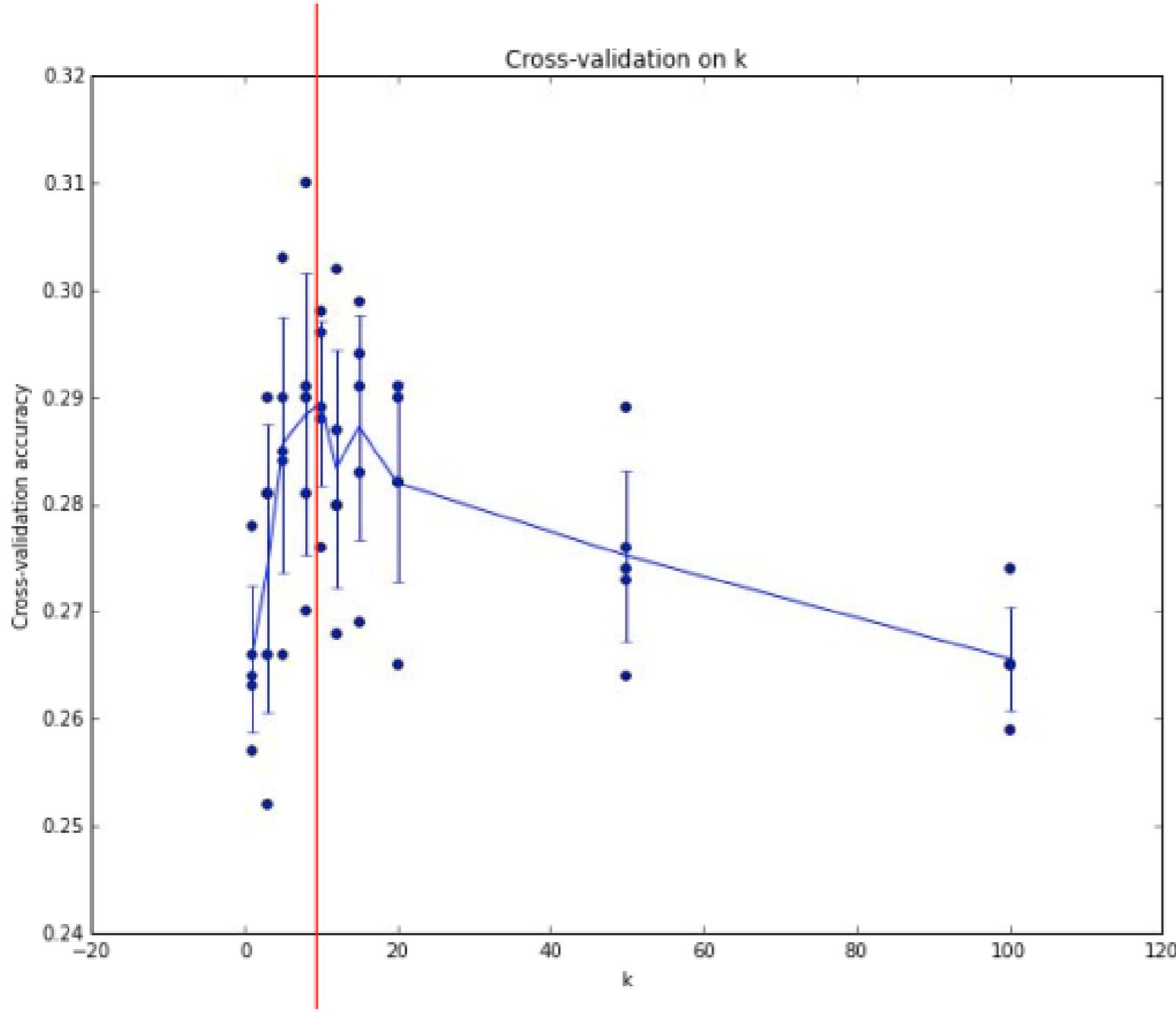


Validation



Cross-validation





Example of
5-fold cross-validation
for the value of k .

Each point: single
outcome.

The line goes
through the mean, bars
indicated standard
deviation

(Seems that $k \approx 7$ works best
for this data)

How to pick hyperparameters?

- Methodology
 - Train and test
 - Train, validate, test
- Train for original model
- Validate to find hyperparameters
- Test to understand generalizability

kNN

Pros

- simple yet effective

Cons

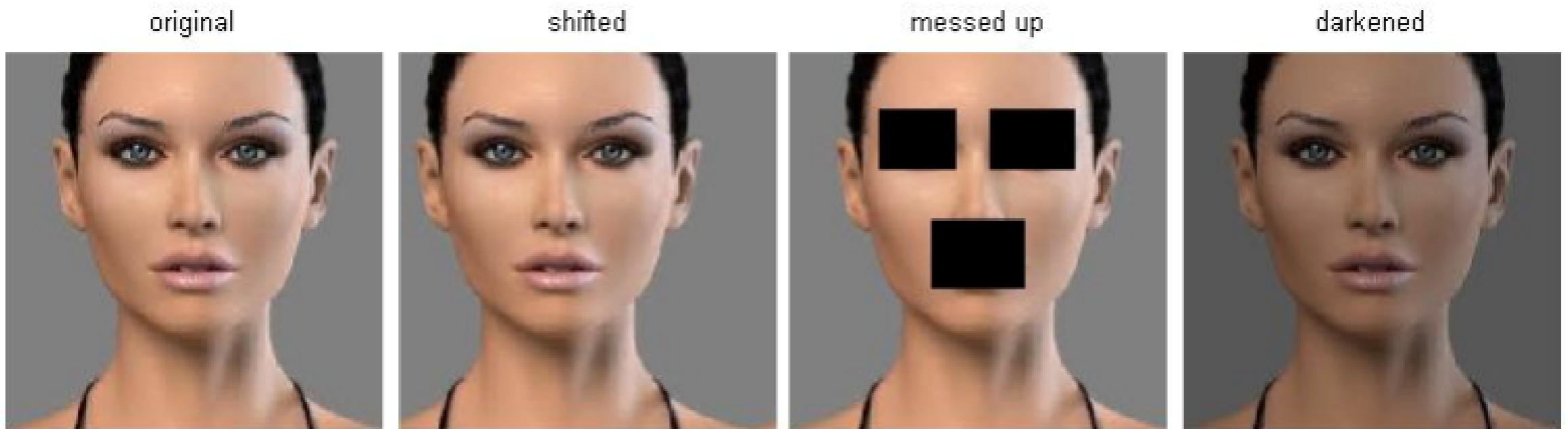
- search is expensive (can be sped-up)
- storage requirements
- difficulties with high-dimensional data

kNN -- Complexity and Storage

- N training images, M test images
- Training: $O(1)$
- Testing: $O(MN)$
- Hmm...
 - Normally need the opposite
 - Slow training (ok), fast testing (necessary)

k-Nearest Neighbor on images **never used**.

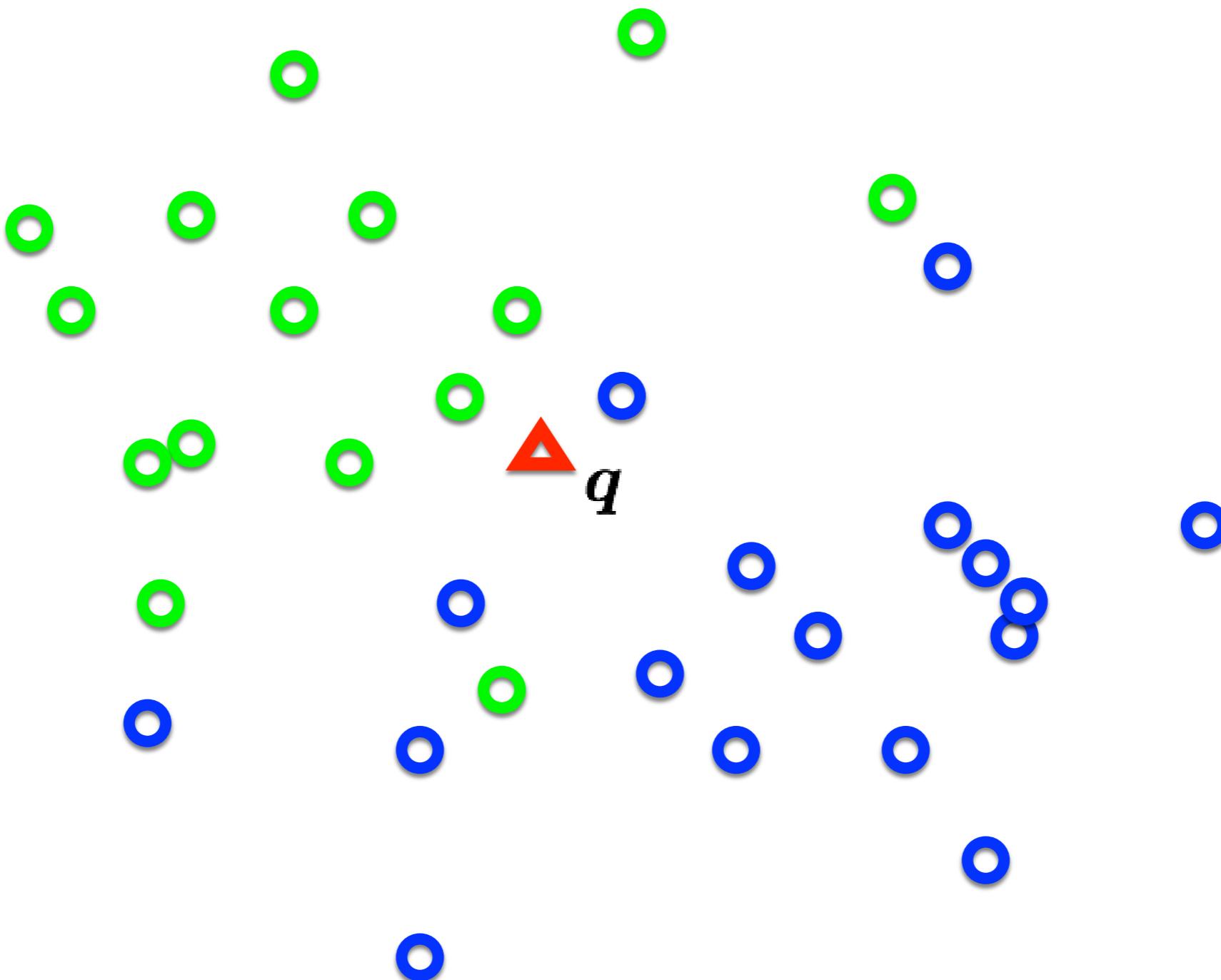
- terrible performance at test time
- distance metrics on level of whole images can be very unintuitive



(all 3 images have same L2 distance to the one on the left)

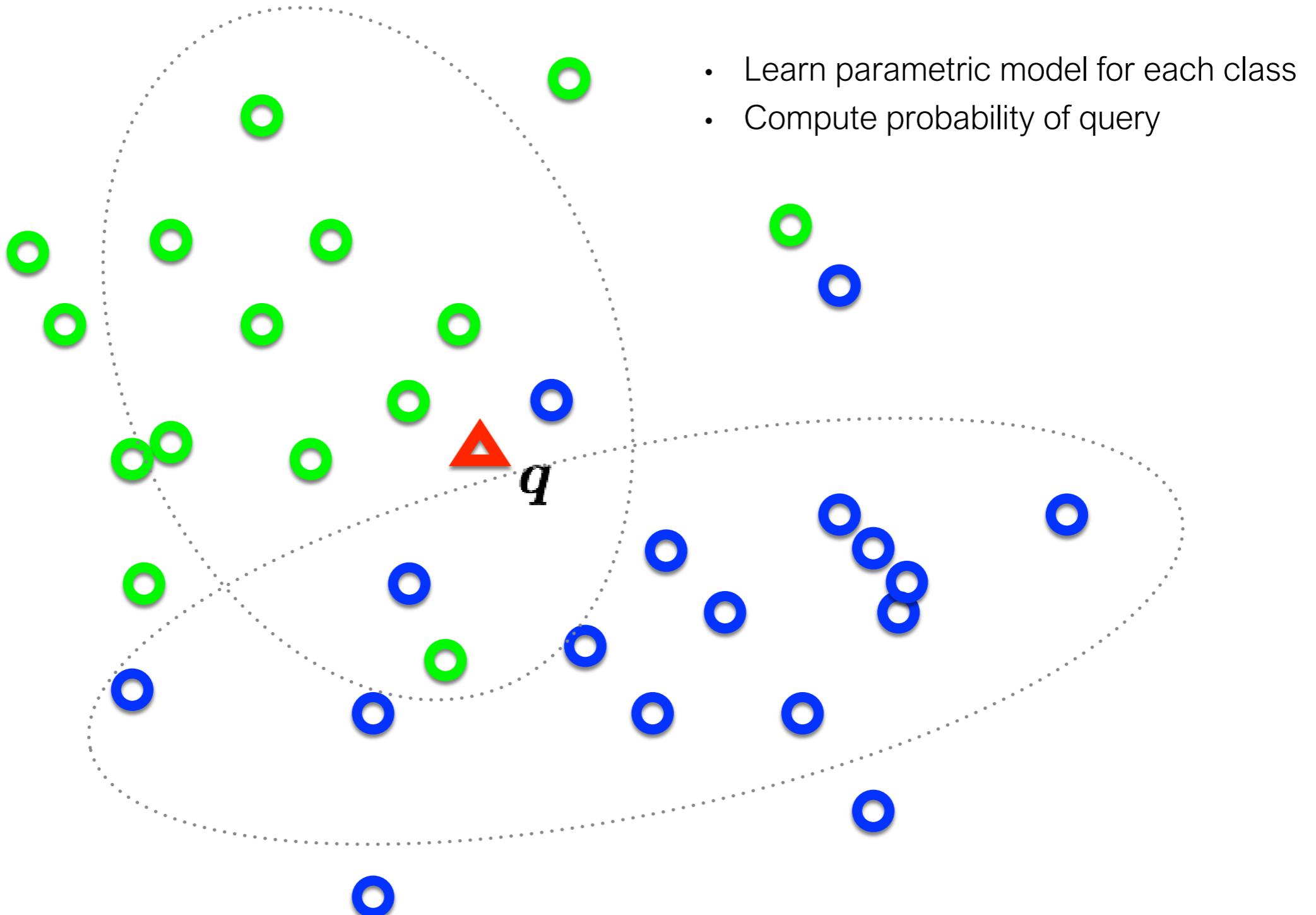
Naïve Bayes

Distribution of data from two classes



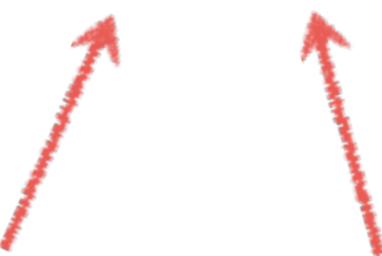
Which class does q belong to?

Distribution of data from two classes



This is called the posterior.
the probability of a class z given the observed features X

$$p(z|X)$$



For classification, z is a
discrete random variable
(e.g., car, person, building)

X is a set of observed features
(e.g., features from a single image)

(it's a function that returns a single probability value)

This is called the posterior:
the probability of a class z given the observed features X

$$p(z|x_1, \dots, x_N)$$



For classification, z is a
discrete random variable
(e.g., car, person, building)

Each x is an observed feature
(e.g., visual words)

(it's a function that returns a single probability value)

Recall:

The posterior can be decomposed according to
Bayes' Rule

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

posterior likelihood prior

In our context...

$$p(\mathbf{z}|\mathbf{x}_1, \dots, \mathbf{x}_N) = \frac{p(\mathbf{x}_1, \dots, \mathbf{x}_N | \mathbf{z})p(\mathbf{z})}{p(\mathbf{x}_1, \dots, \mathbf{x}_N)}$$

The naive Bayes' classifier is solving this optimization

$$\hat{z} = \arg \max_{z \in \mathcal{Z}} p(z | \mathbf{X})$$

MAP (maximum a posteriori) estimate

$$\hat{z} = \arg \max_{z \in \mathcal{Z}} \frac{p(\mathbf{X}|z)p(z)}{p(\mathbf{X})} \quad \text{Bayes' Rule}$$

$$\hat{z} = \arg \max_{z \in \mathcal{Z}} p(\mathbf{X}|z)p(z) \quad \text{Remove constants}$$

To optimize this...we need to compute this

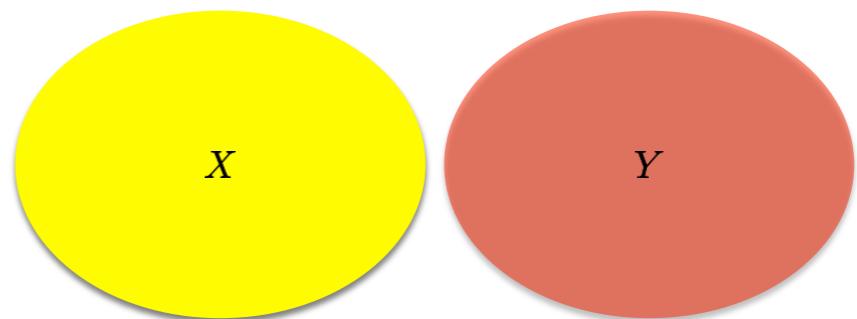
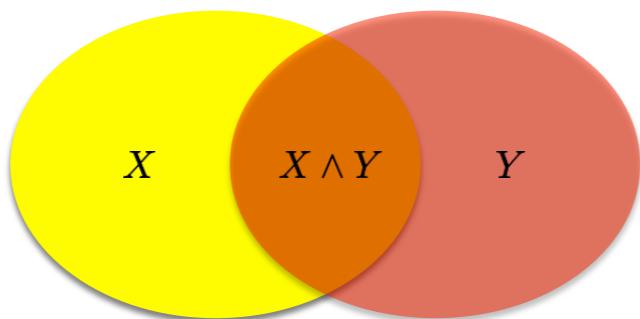


Compute the likelihood...

A naive Bayes' classifier assumes all features are
conditionally independent

$$\begin{aligned} p(\mathbf{x}_1, \dots, \mathbf{x}_N | \mathbf{z}) &= p(\mathbf{x}_1 | \mathbf{z})p(\mathbf{x}_2, \dots, \mathbf{x}_N | \mathbf{z}) \\ &= p(\mathbf{x}_1 | \mathbf{z})p(\mathbf{x}_2 | \mathbf{z})p(\mathbf{x}_3, \dots, \mathbf{x}_N | \mathbf{z}) \\ &= p(\mathbf{x}_1 | \mathbf{z})p(\mathbf{x}_2 | \mathbf{z}) \cdots p(\mathbf{x}_N | \mathbf{z}) \end{aligned}$$

Recall:



$$p(x, y) = p(x|y)p(y) \quad p(x, y) = p(x)p(y)$$

To compute the MAP estimate

Given (1) a set of known parameters

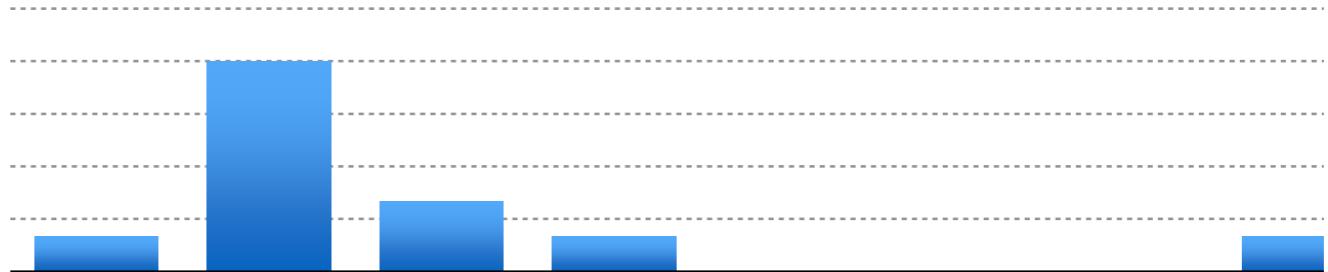
$$p(\mathbf{z}) \quad p(\mathbf{x}|\mathbf{z})$$

(2) observations

$$\{x_1, x_2, \dots, x_N\}$$

Compute which \mathbf{z} has the largest probability

$$\hat{\mathbf{z}} = \arg \max_{\mathbf{z} \in \mathcal{Z}} p(\mathbf{z}) \prod_n p(x_n | \mathbf{z})$$



count	1	6	2	1	0	0	0	1
word	Tartan	robot	CHIMP	CMU	bio	soft	ankle	sensor
p(x z)	0.09	0.55	0.18	0.09	0.0	0.0	0.0	0.09

$$\begin{aligned}
 p(X|z) &= \prod_v p(x_v|z)^{c(w_v)} \\
 &= (0.09)^1 (0.55)^6 \cdots (0.09)^1
 \end{aligned}$$

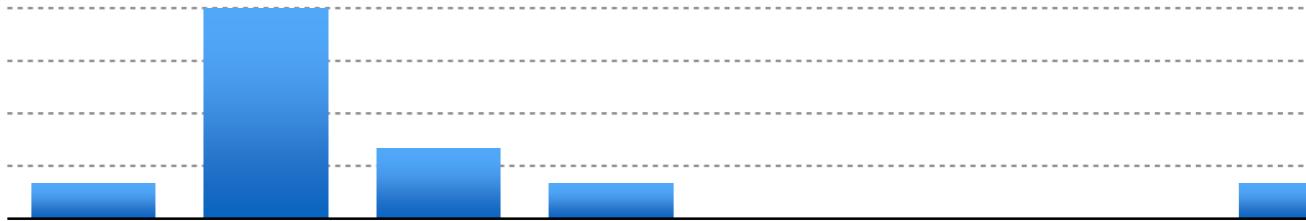
Numbers get really small so use log probabilities

$$\log p(X|z = \text{'grandchallenge'}) = -2.42 - 3.68 - 3.43 - 2.42 - 0.07 - 0.07 - 0.07 - 2.42 = -14.58$$

$$\log p(X|z = \text{'softrobot'}) = -7.63 - 9.37 - 15.18 - 2.97 - 0.02 - 0.01 - 0.02 - 2.27 = -37.48$$

* typically add pseudo-counts (0.001)

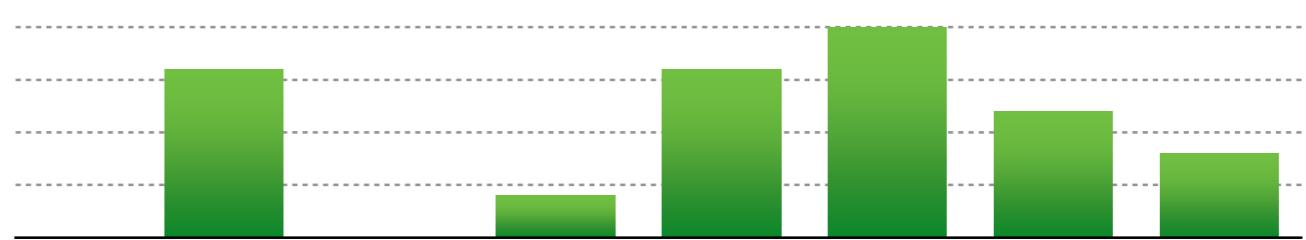
** this is an example for computing the likelihood, need to multiply times **prior** to get posterior



p(x z)	0.09	0.55	0.18	0.09	0.0	0.0	0.09	
word	Tartan	robot	CHIMP	CMU	bio	soft	ankle	sensor
count	1	6	2	1	0	0	0	1

$$\log p(X|z=\text{grand challenge}) = -14.58$$

$$\log p(X|z=\text{bio inspired}) = -37.48$$



p(x z)	0.0	0.21	0.0	0.05	0.21	0.26	0.16	0.11
word	Tartan	robot	CHIMP	CMU	bio	soft	ankle	sensor
count	0	4	0	1	4	5	3	2

$$\log p(X|z=\text{grand challenge}) = -94.06$$

$$\log p(X|z=\text{bio inspired}) = -32.41$$

* typically add pseudo-counts (0.001)

** this is an example for computing the likelihood, need to multiply times prior to get posterior

Support Vector Machine

Image Classification



(assume given set of discrete labels)
{dog, cat, truck, plane, ...}



cat

Score function



class scores

Linear Classifier

define a **score function**

data (histogram)

$$f(x_i, W, b) = Wx_i + b$$

class scores

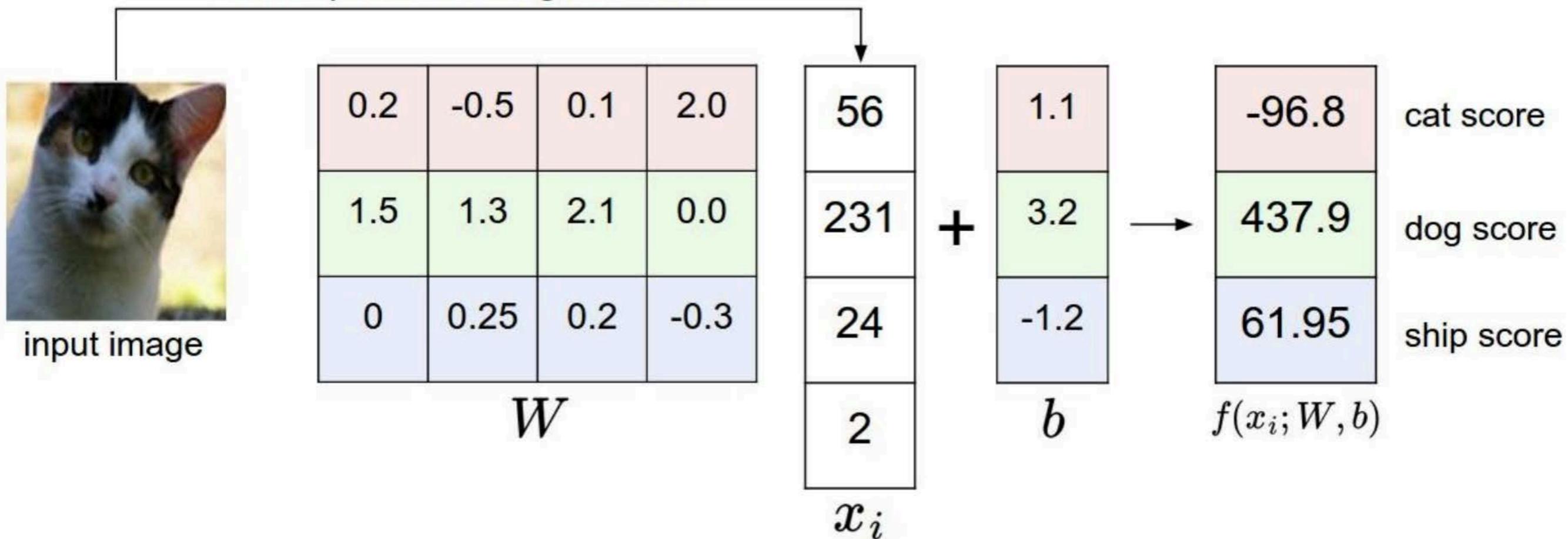
“weights”

“parameters”

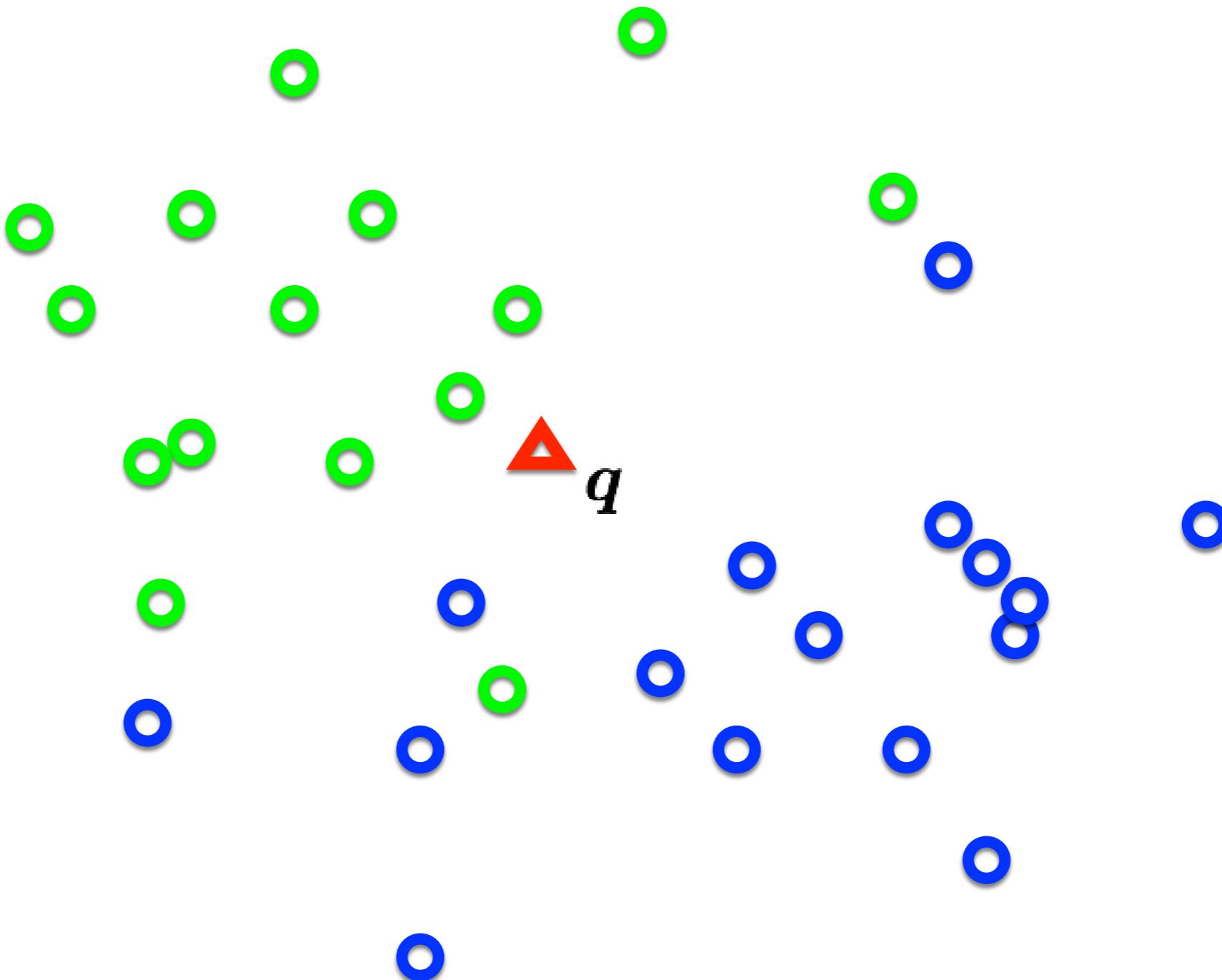
“bias vector”

Example with an image with 4 pixels, and 3 classes (**cat/dog/ship**)

Convert image to histogram representation

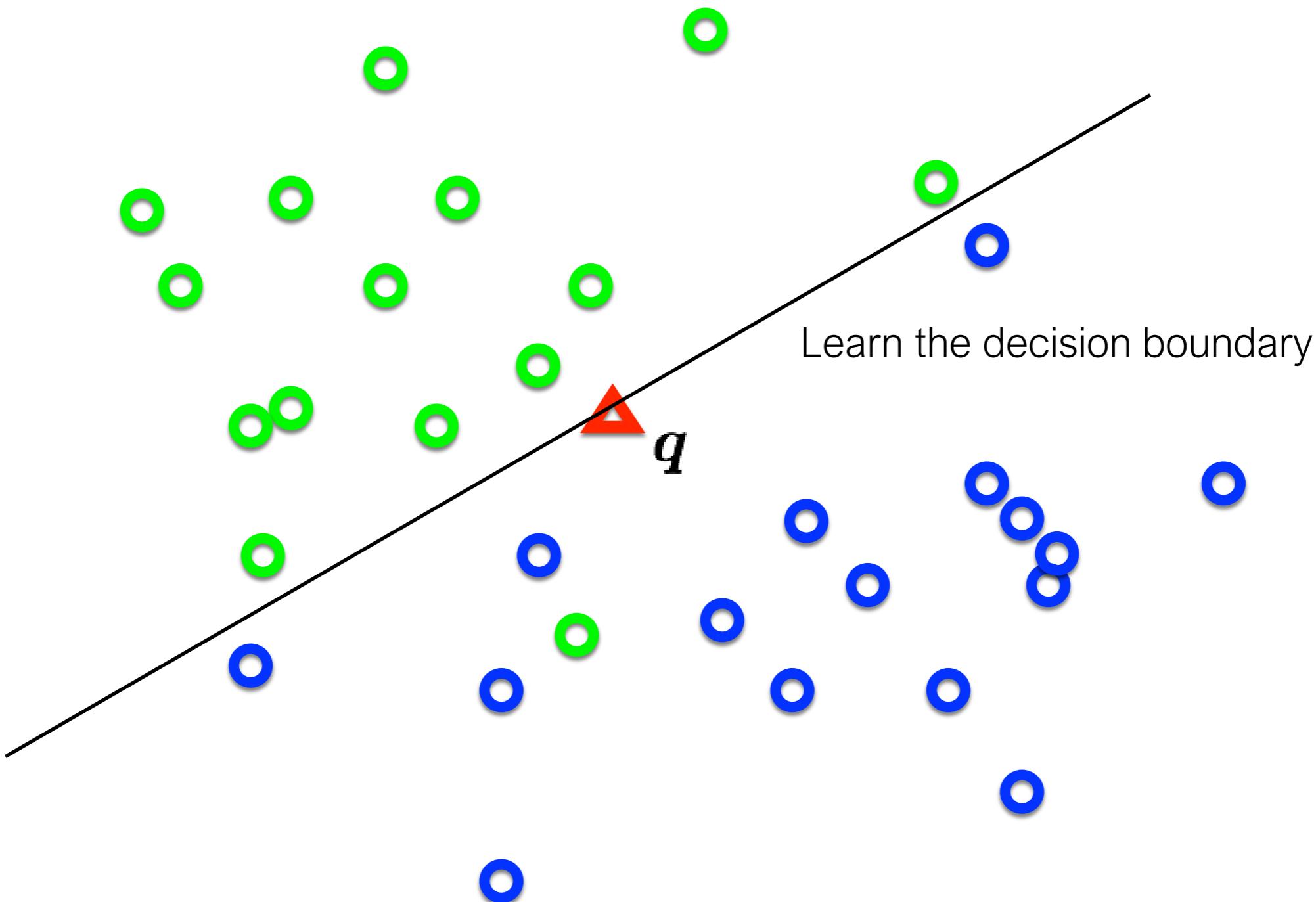


Distribution of data from two classes



Which class does q belong to?

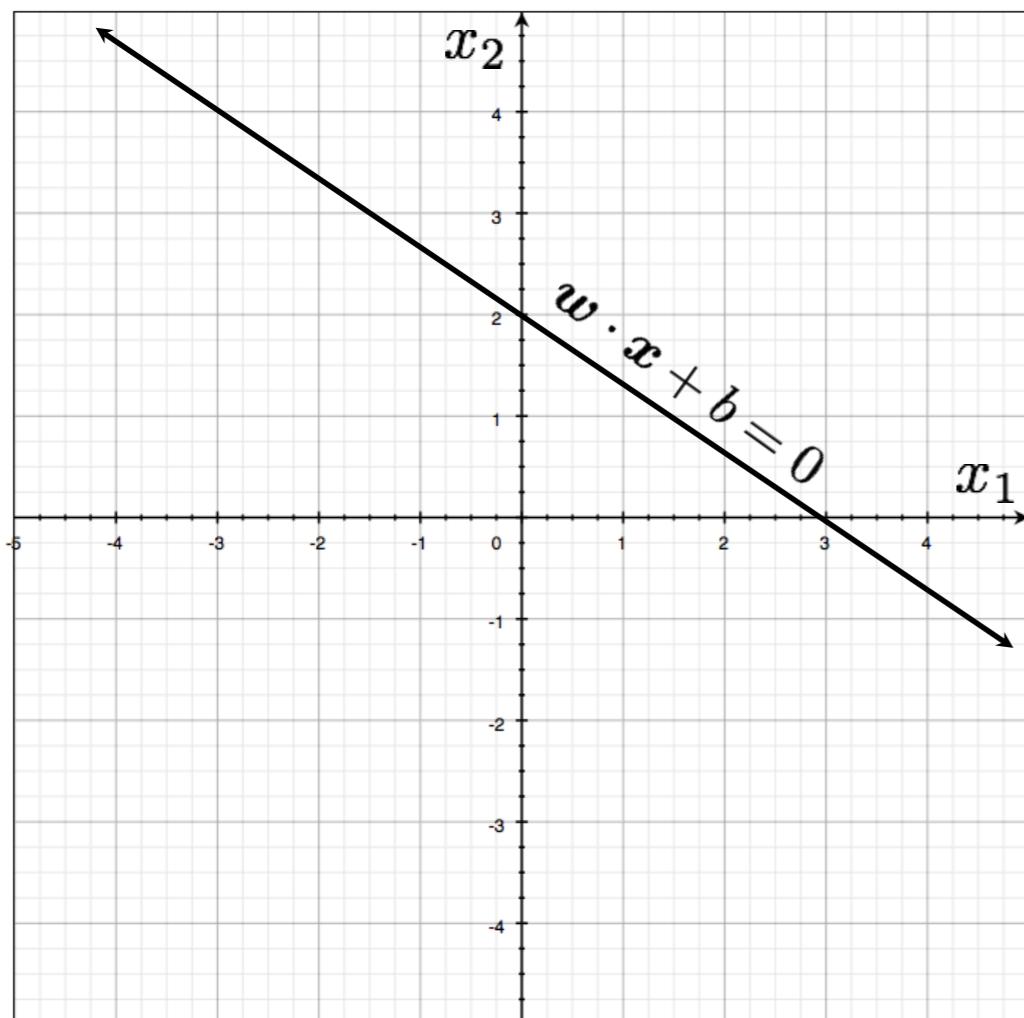
Distribution of data from two classes



First we need to understand hyperplanes...

Hyperplanes (lines) in 2D

$$w_1x_1 + w_2x_2 + b = 0$$



a line can be written as
dot product plus a bias

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

$$\mathbf{w} \in \mathcal{R}^2$$

another version, add a weight 1 and
push the bias inside

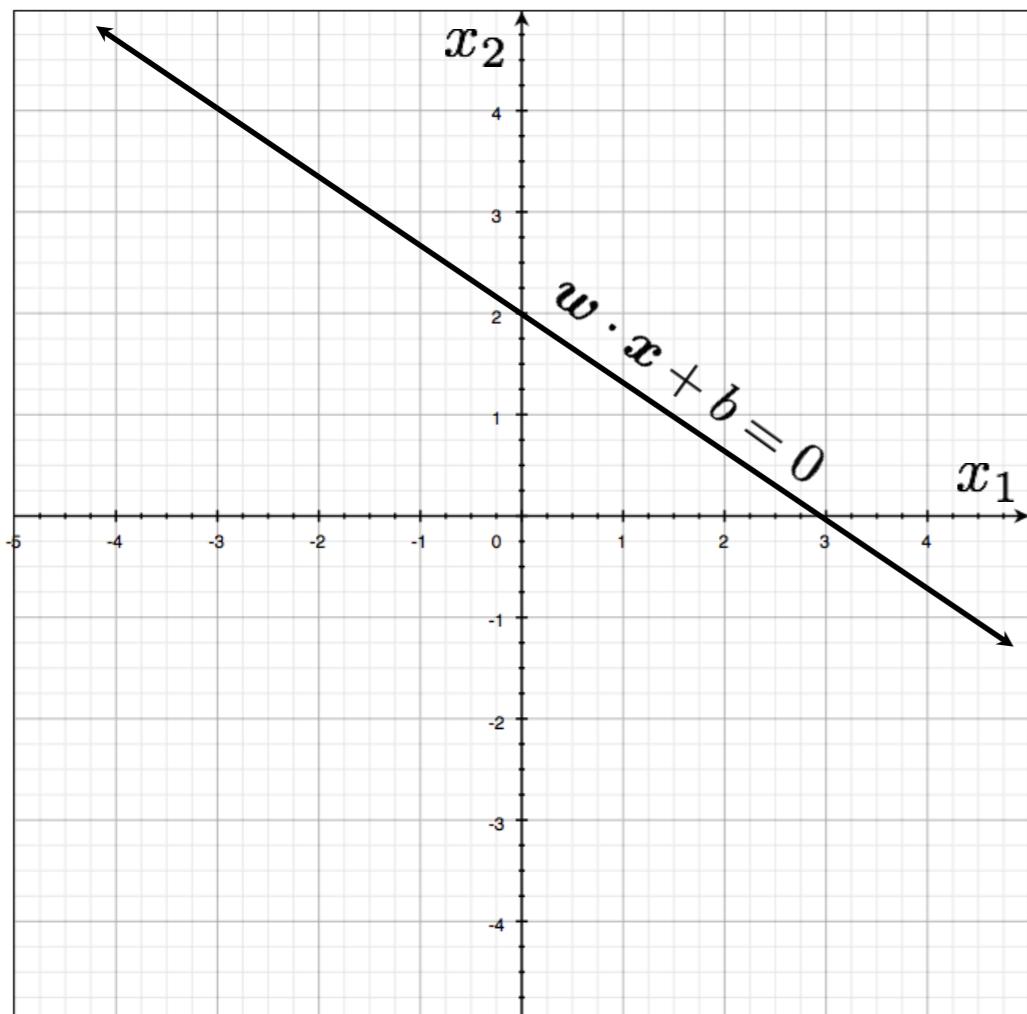
$$\mathbf{w} \cdot \mathbf{x} = 0$$

$$\mathbf{w} \in \mathcal{R}^3$$

Hyperplanes (lines) in 2D

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \quad (\text{offset/bias outside}) \quad \mathbf{w} \cdot \mathbf{x} = 0 \quad (\text{offset/bias inside})$$

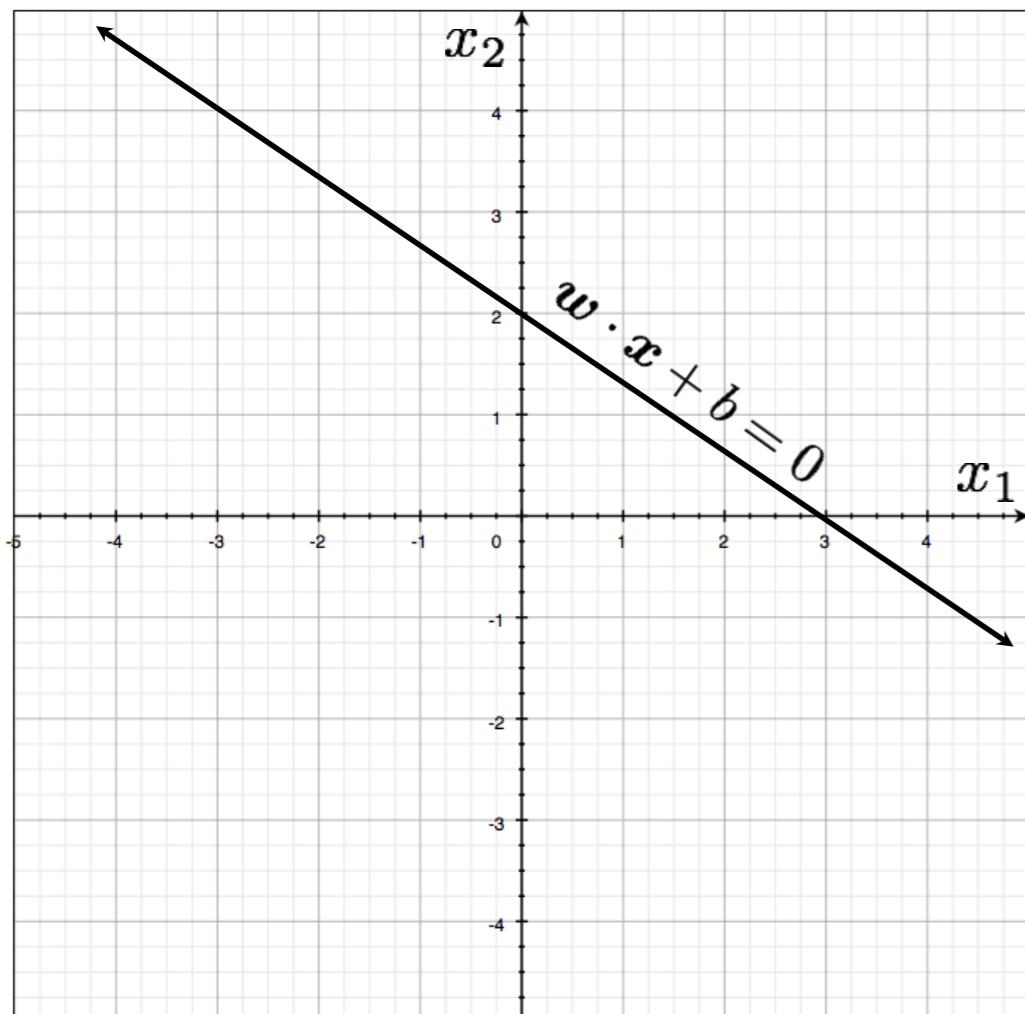
$$w_1x_1 + w_2x_2 + b = 0$$



Hyperplanes (lines) in 2D

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \quad (\text{offset/bias outside}) \quad \mathbf{w} \cdot \mathbf{x} = 0 \quad (\text{offset/bias inside})$$

$$w_1x_1 + w_2x_2 + b = 0$$



Important property:
Free to choose any normalization of w

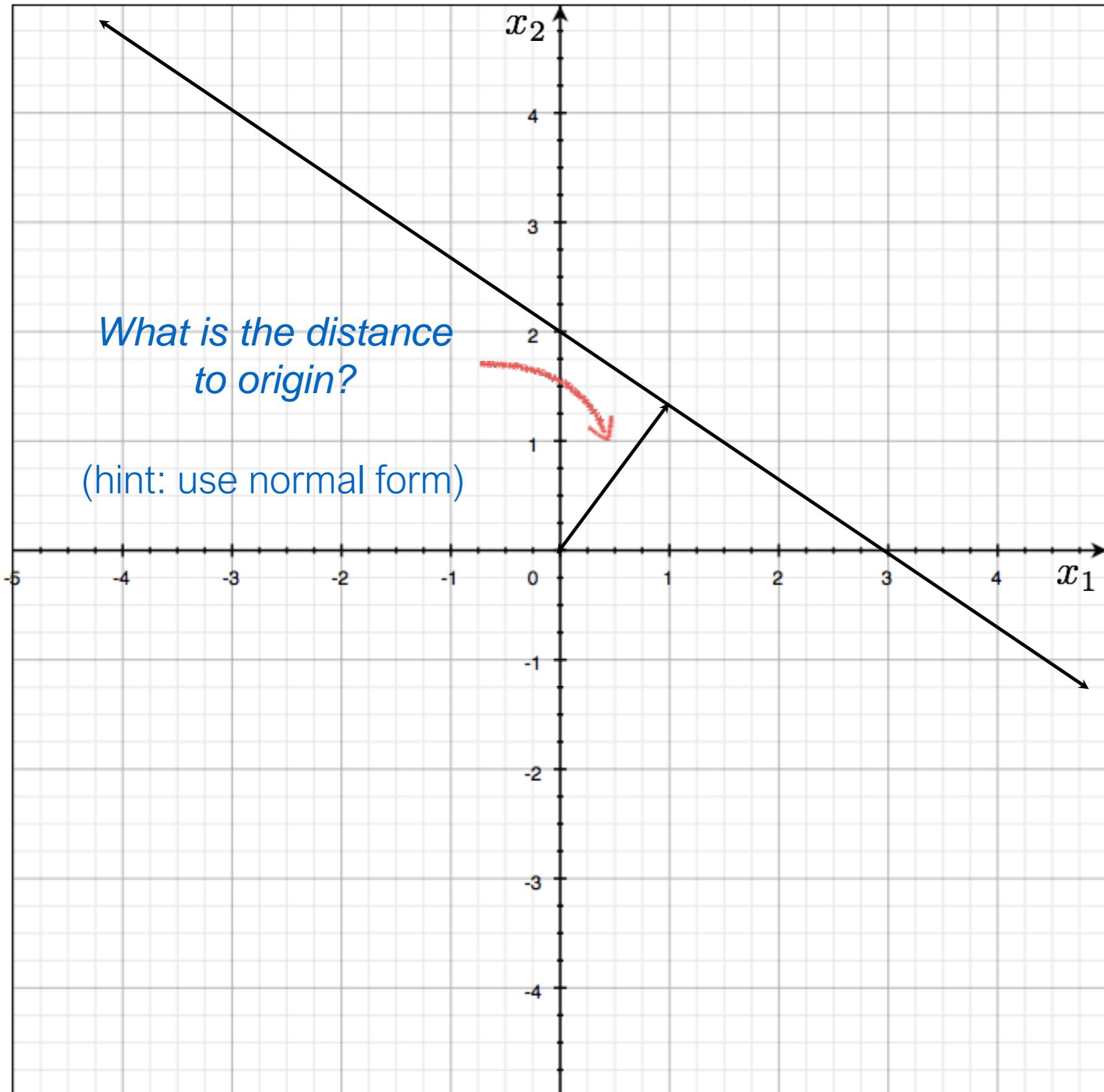
The line

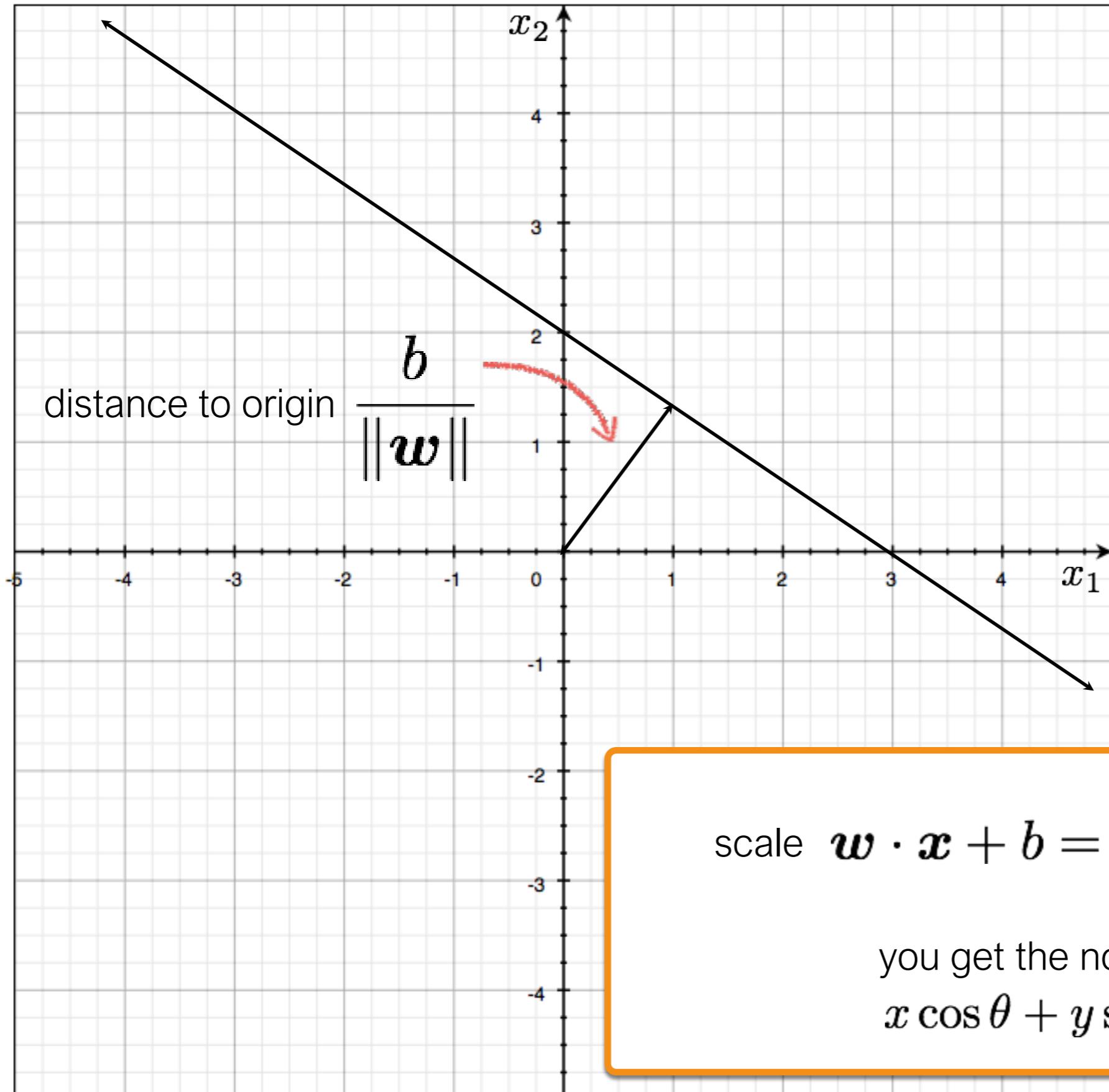
$$w_1x_1 + w_2x_2 + b = 0$$

and the line

$$\lambda(w_1x_1 + w_2x_2 + b) = 0$$

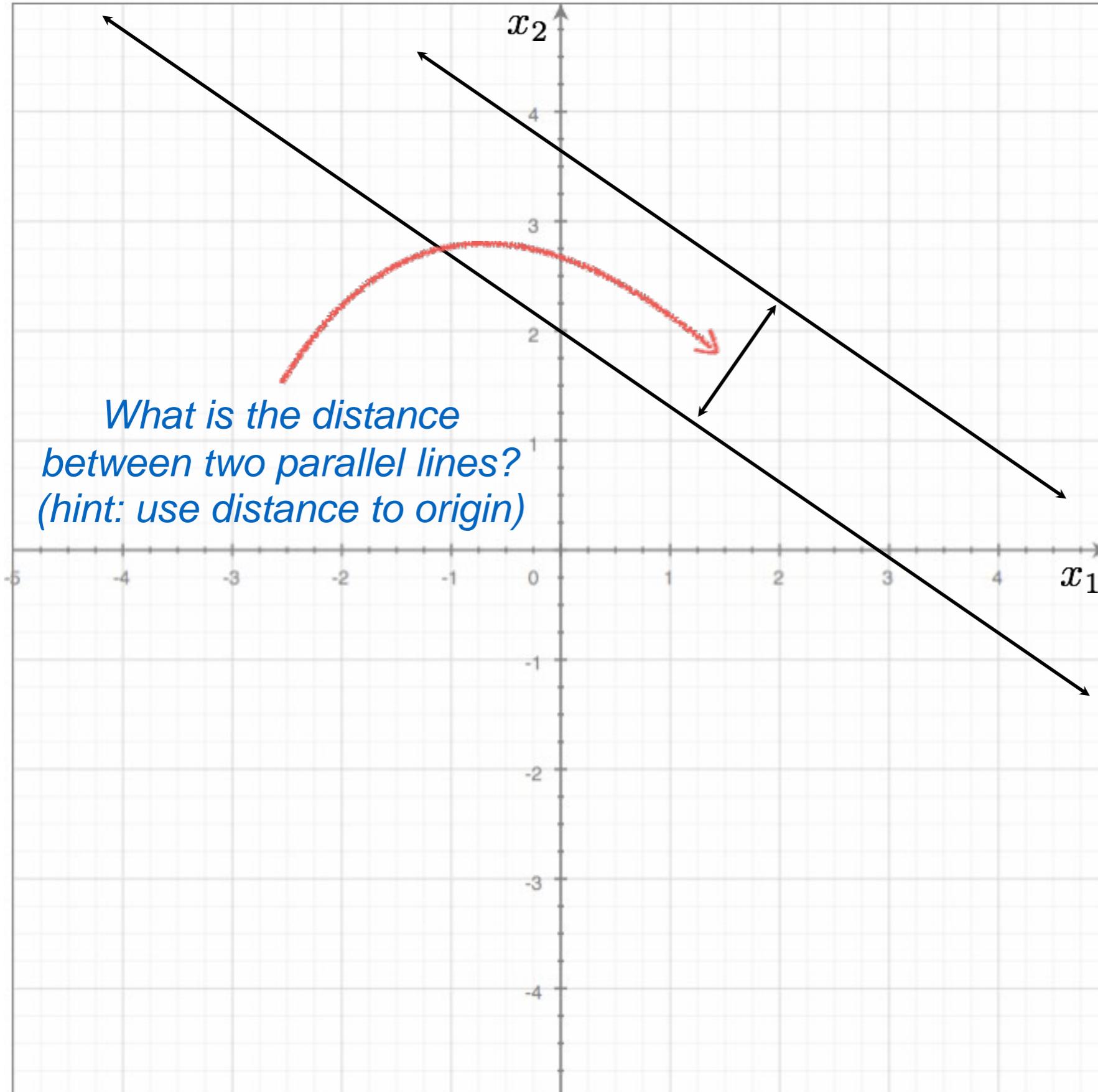
define the same line





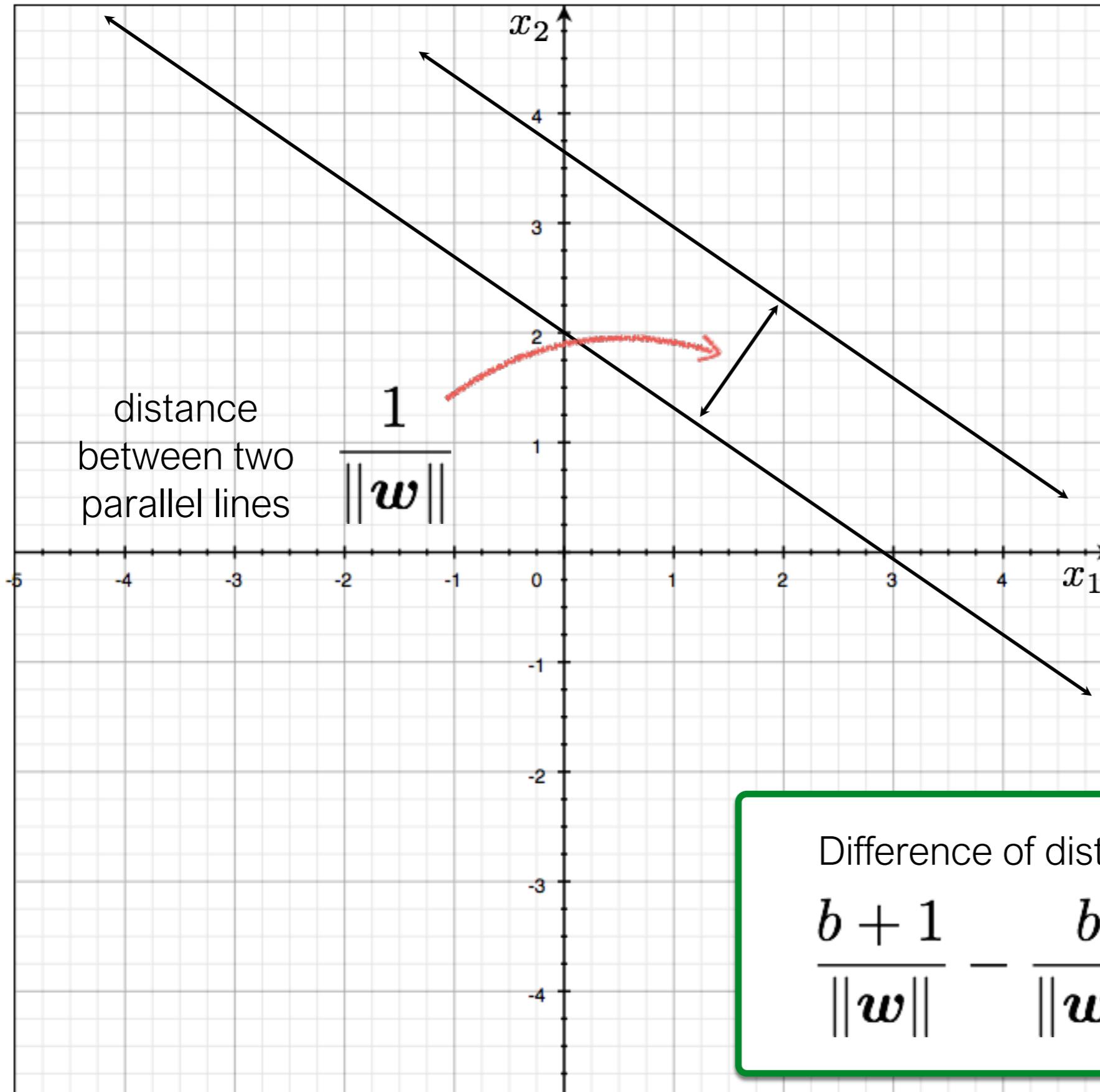
scale $w \cdot x + b = 0$ by $\frac{1}{\|w\|}$

you get the normal form
 $x \cos \theta + y \sin \theta = \rho$



$$\mathbf{w} \cdot \mathbf{x} + b = -1$$

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

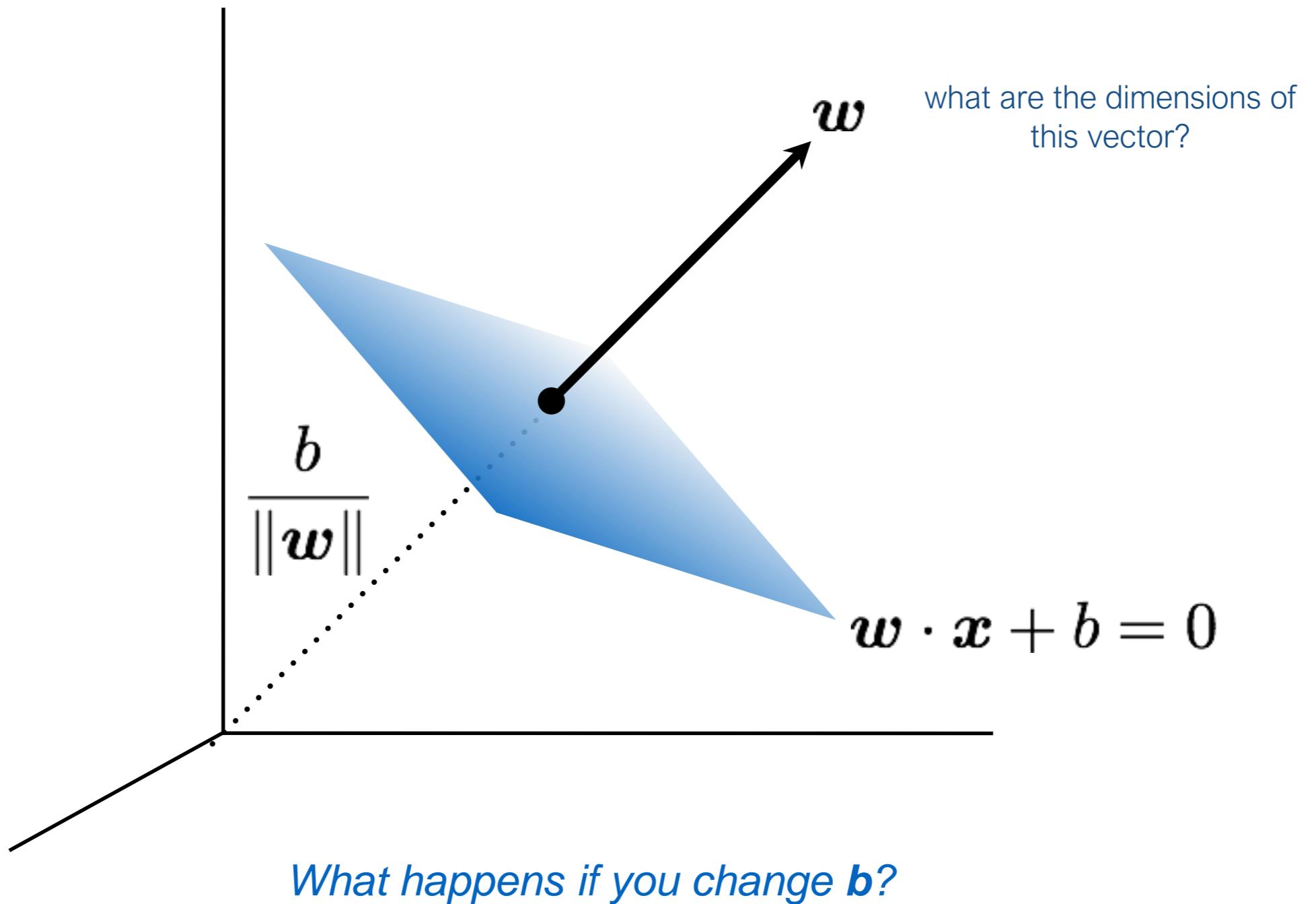


Difference of distance to origin

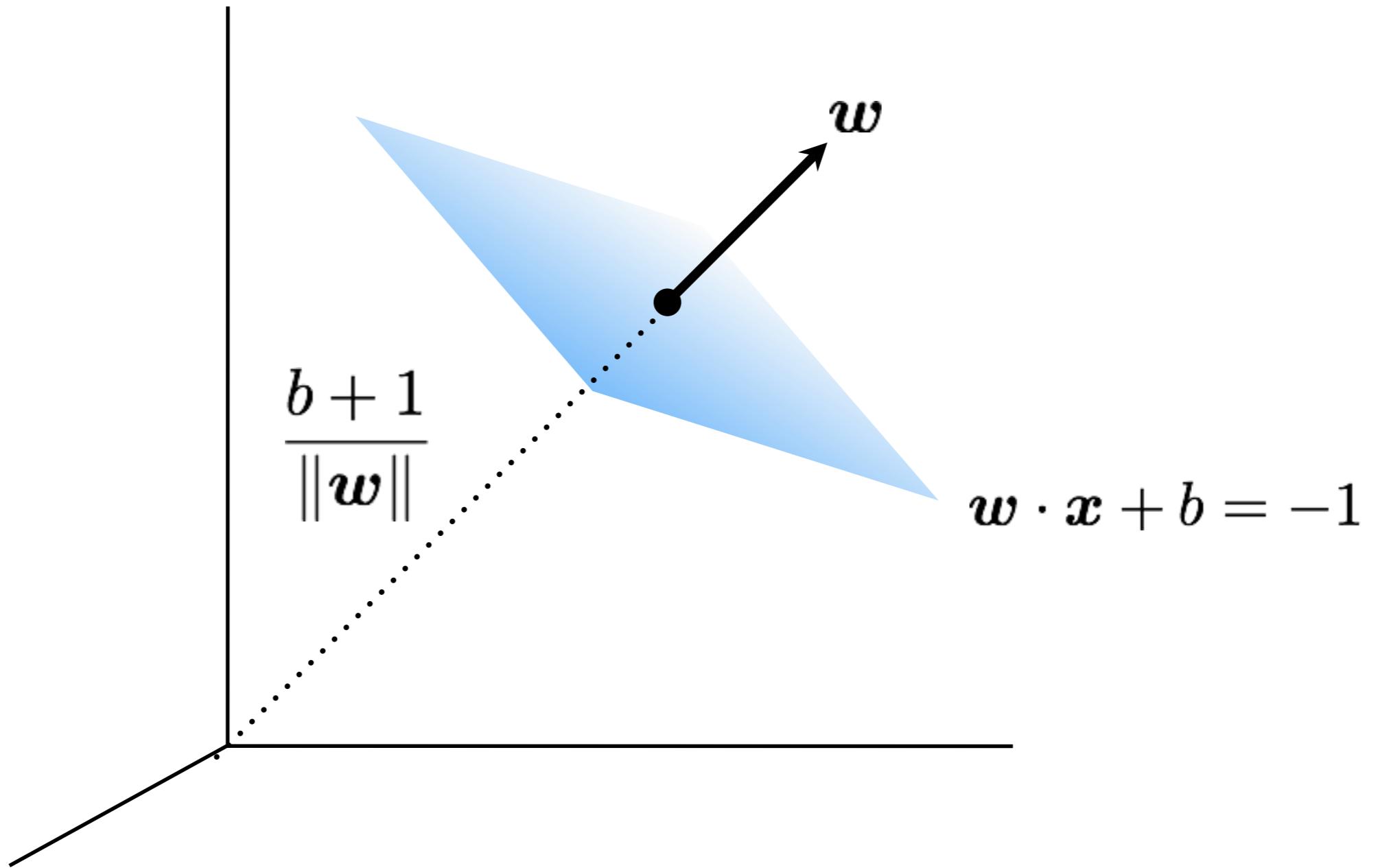
$$\frac{b+1}{\|w\|} - \frac{b}{\|w\|} = \frac{1}{\|w\|}$$

Now we can go to 3D ...

Hyperplanes (planes) in 3D

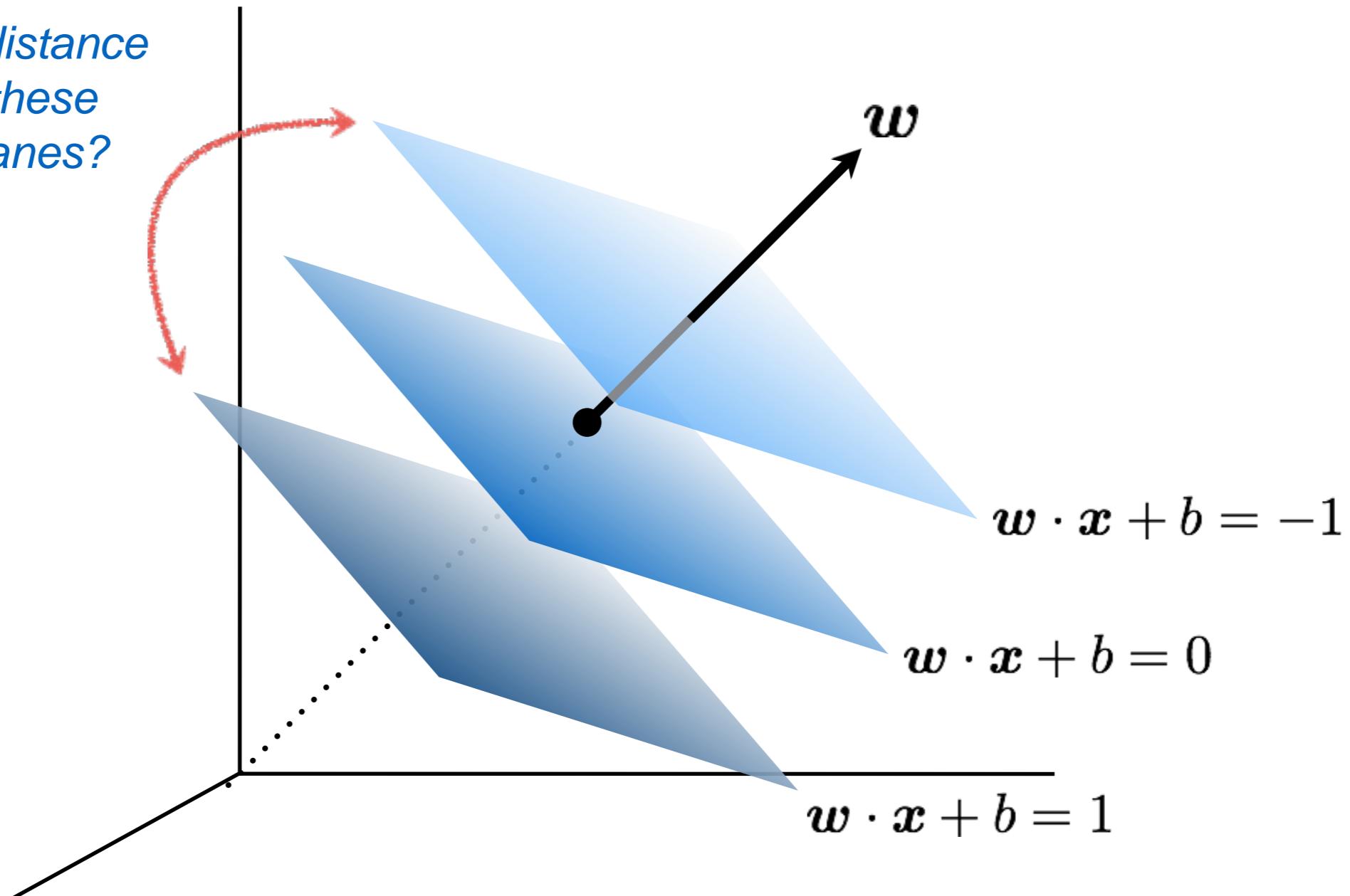


Hyperplanes (planes) in 3D

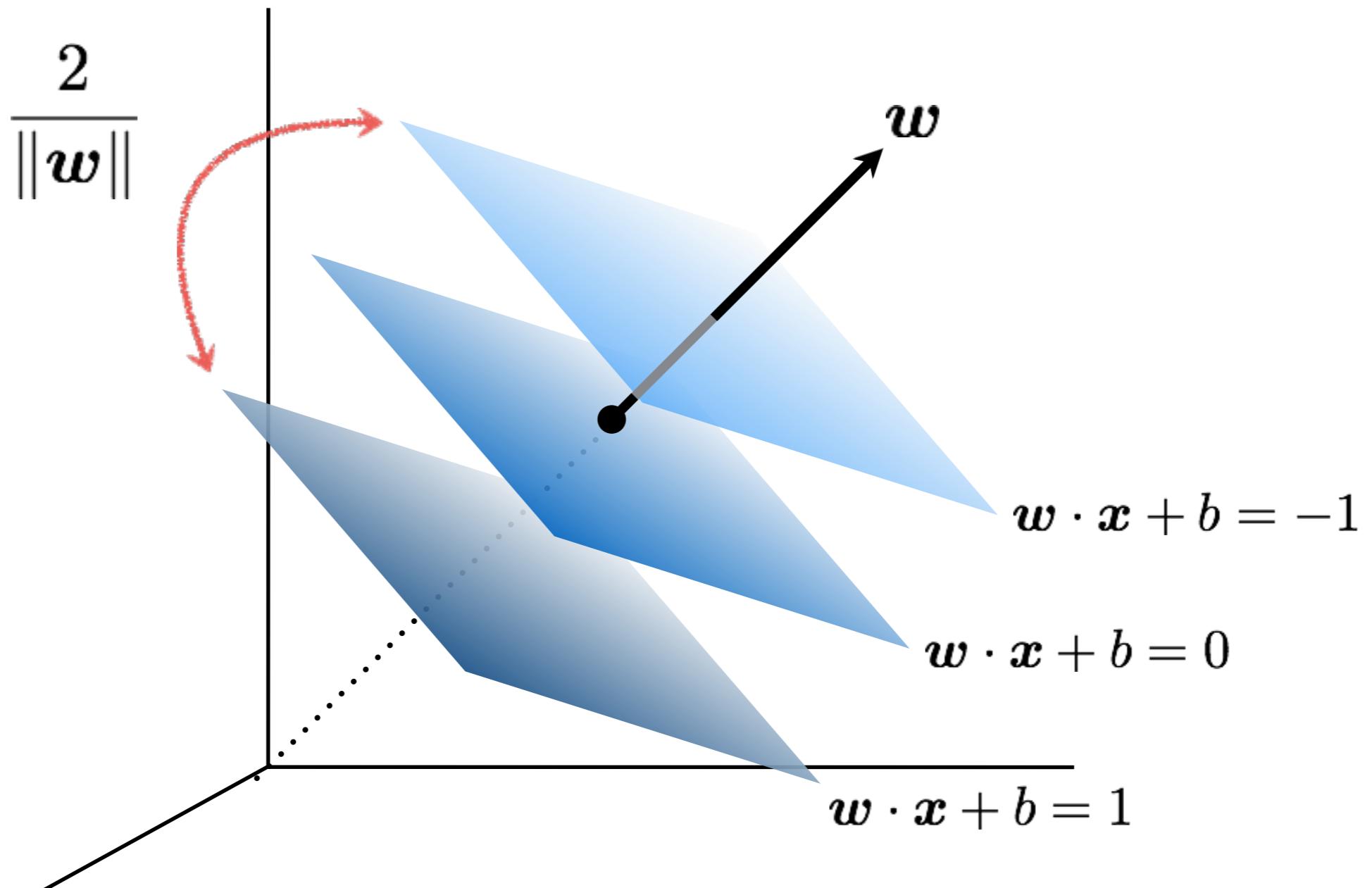


Hyperplanes (planes) in 3D

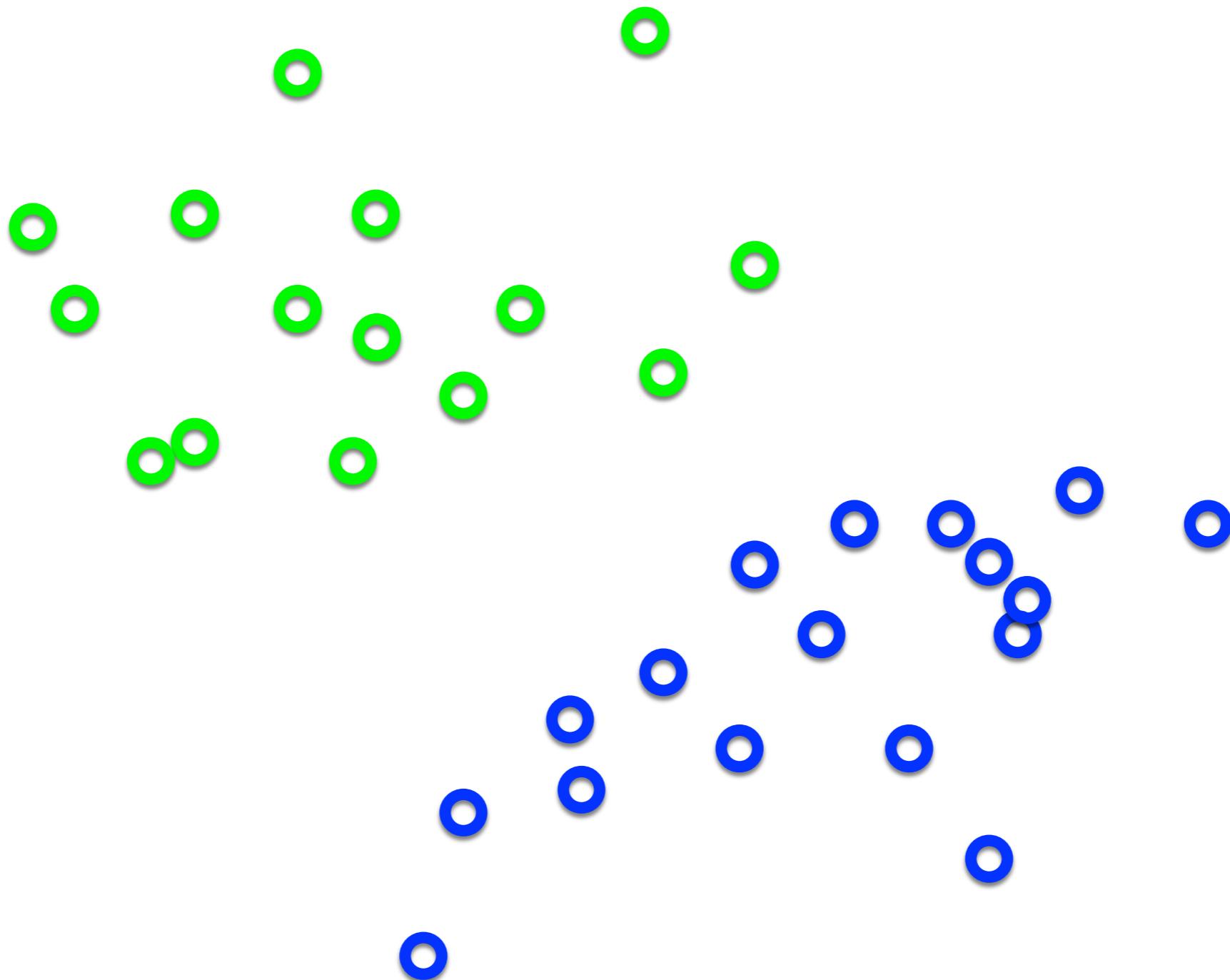
*What's the distance
between these
parallel planes?*



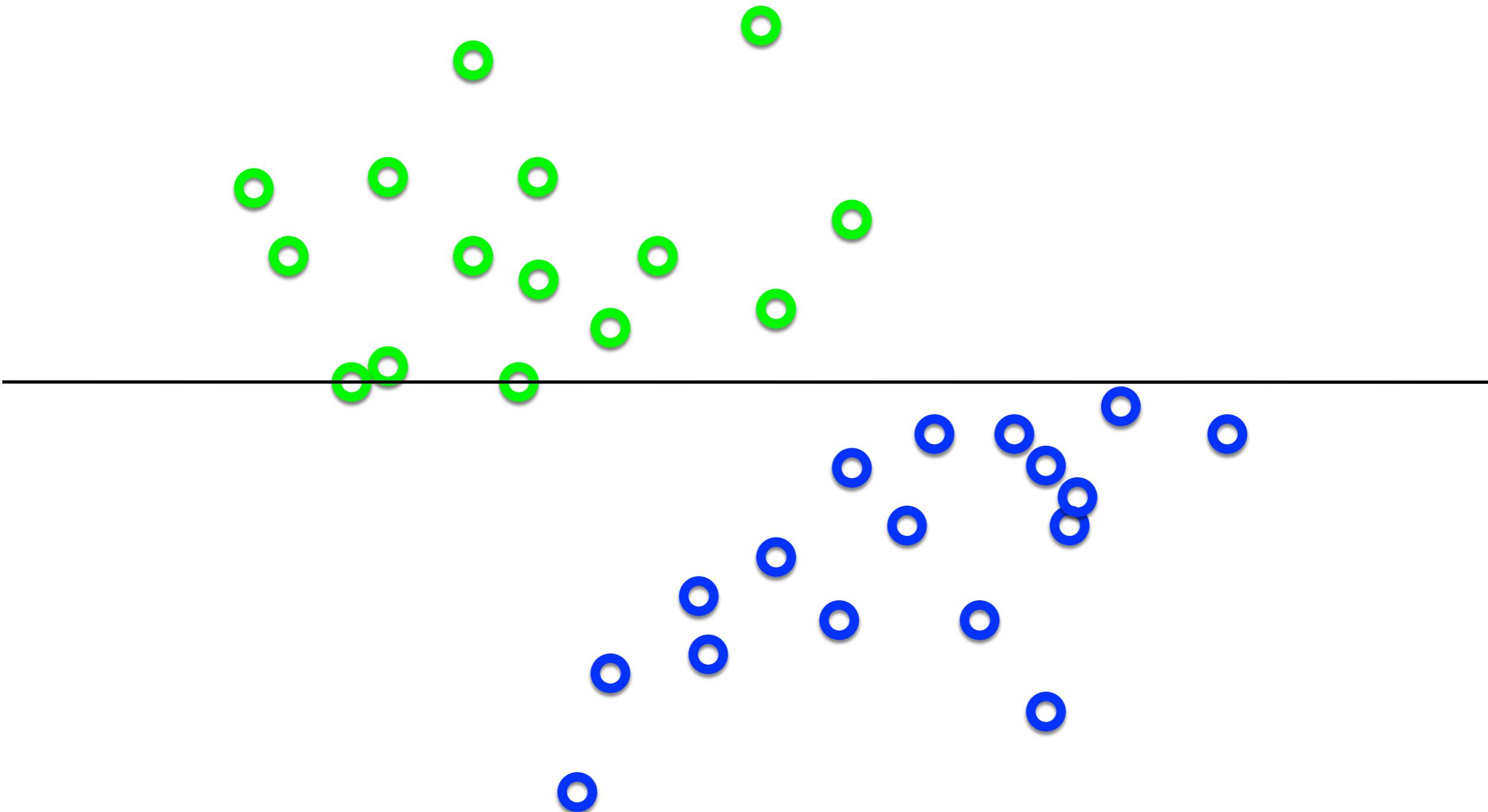
Hyperplanes (planes) in 3D



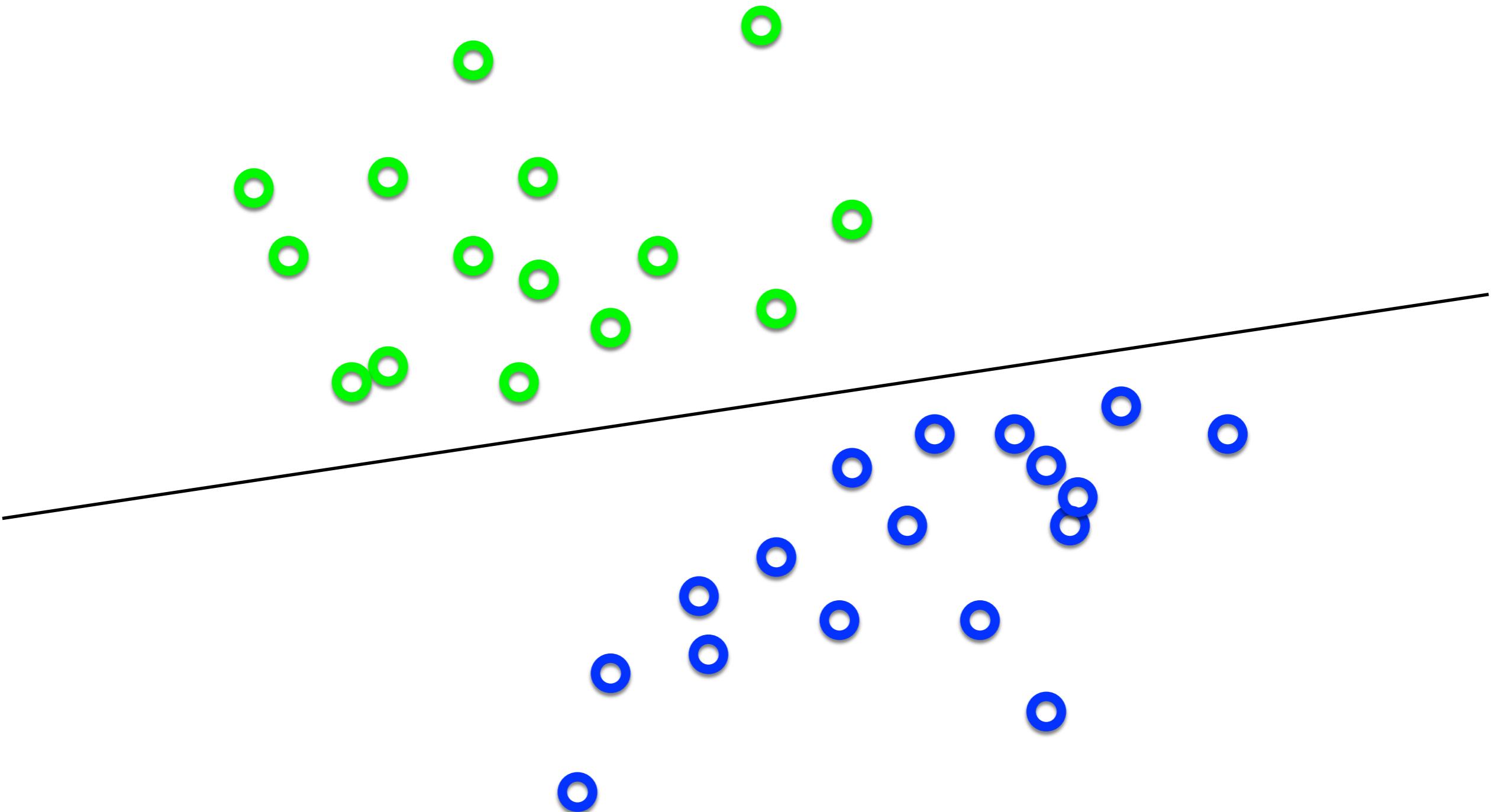
What's the best **w**?



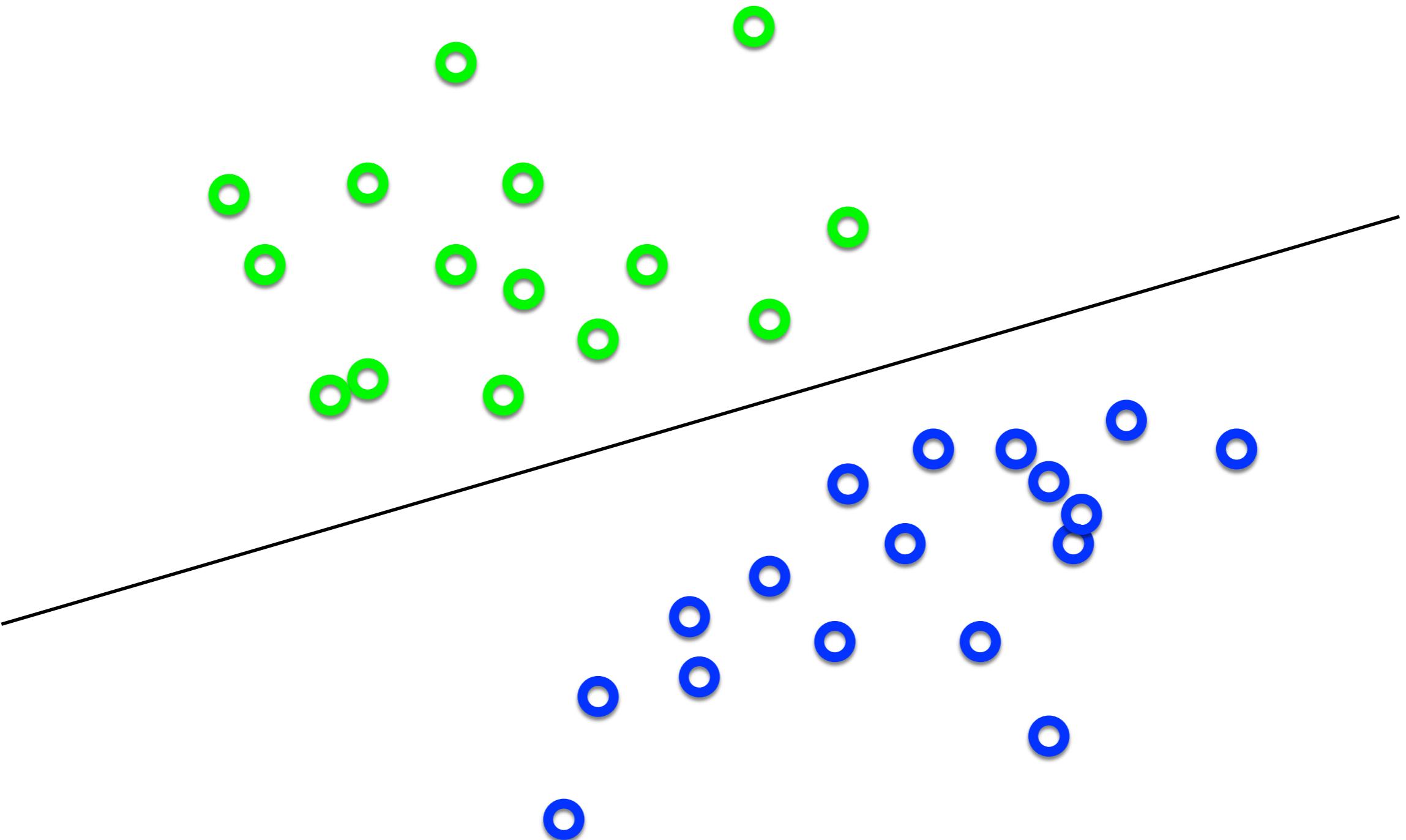
What's the best w ?



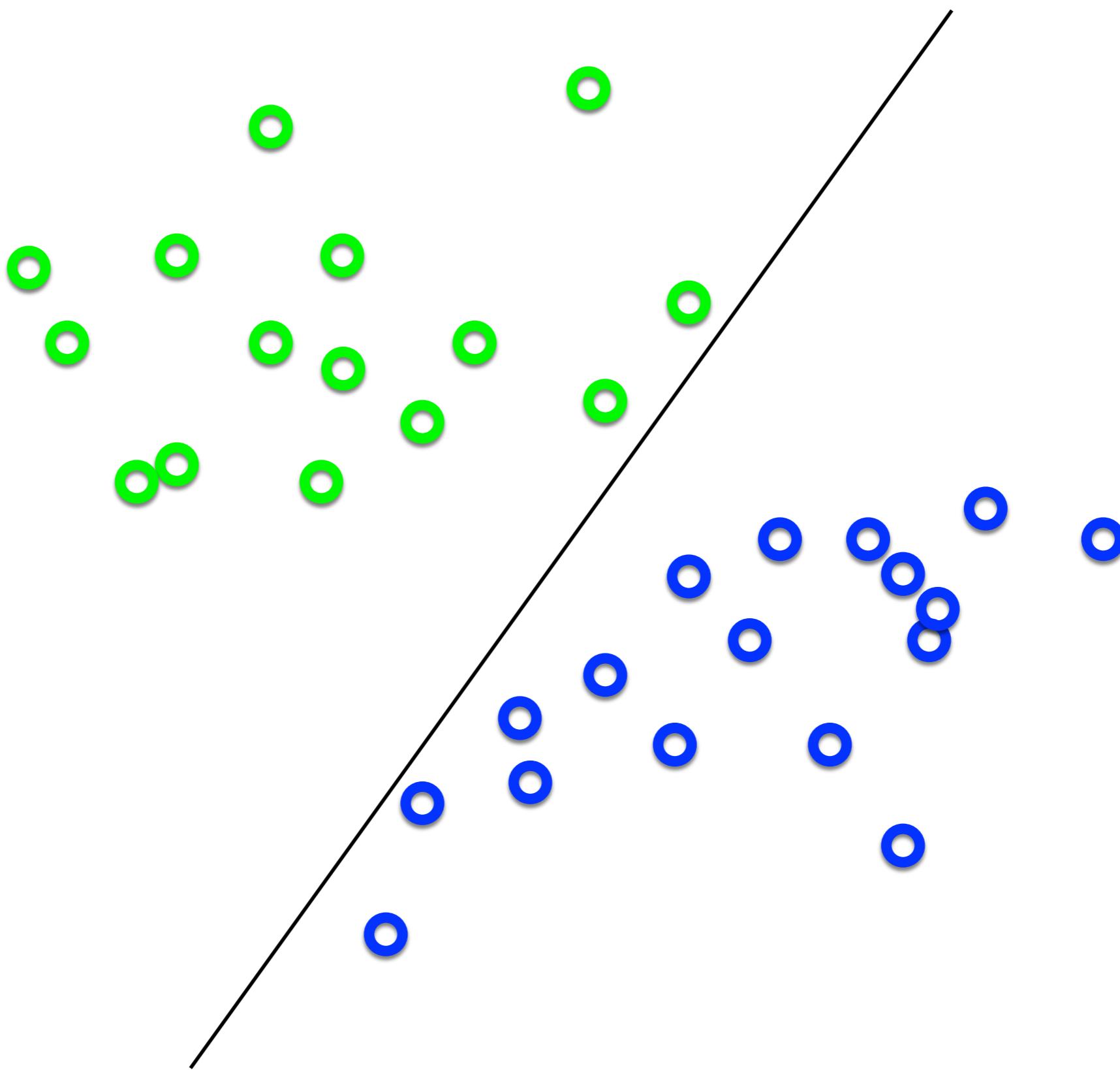
What's the best \mathbf{w} ?



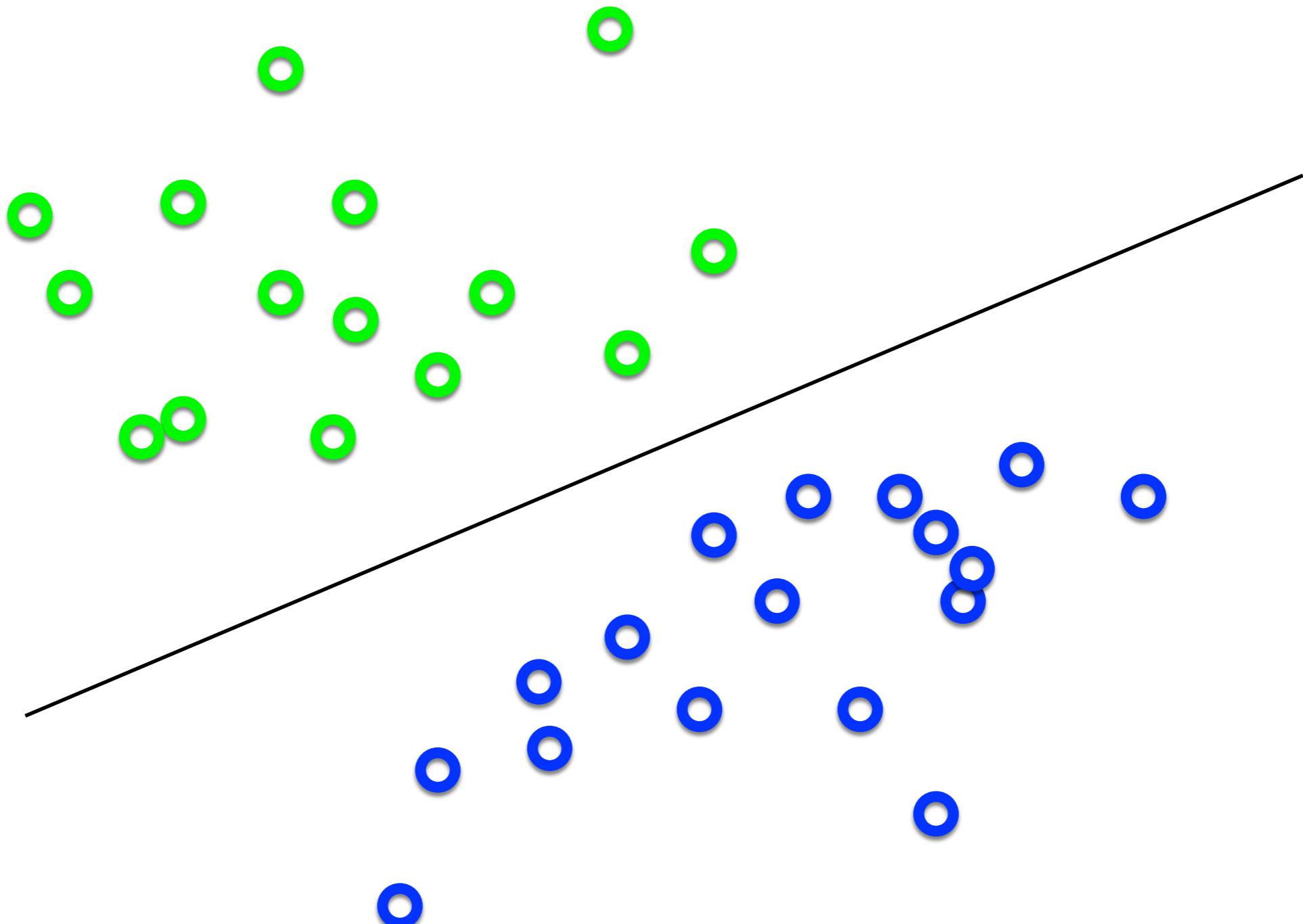
What's the best \mathbf{w} ?



What's the best **w**?

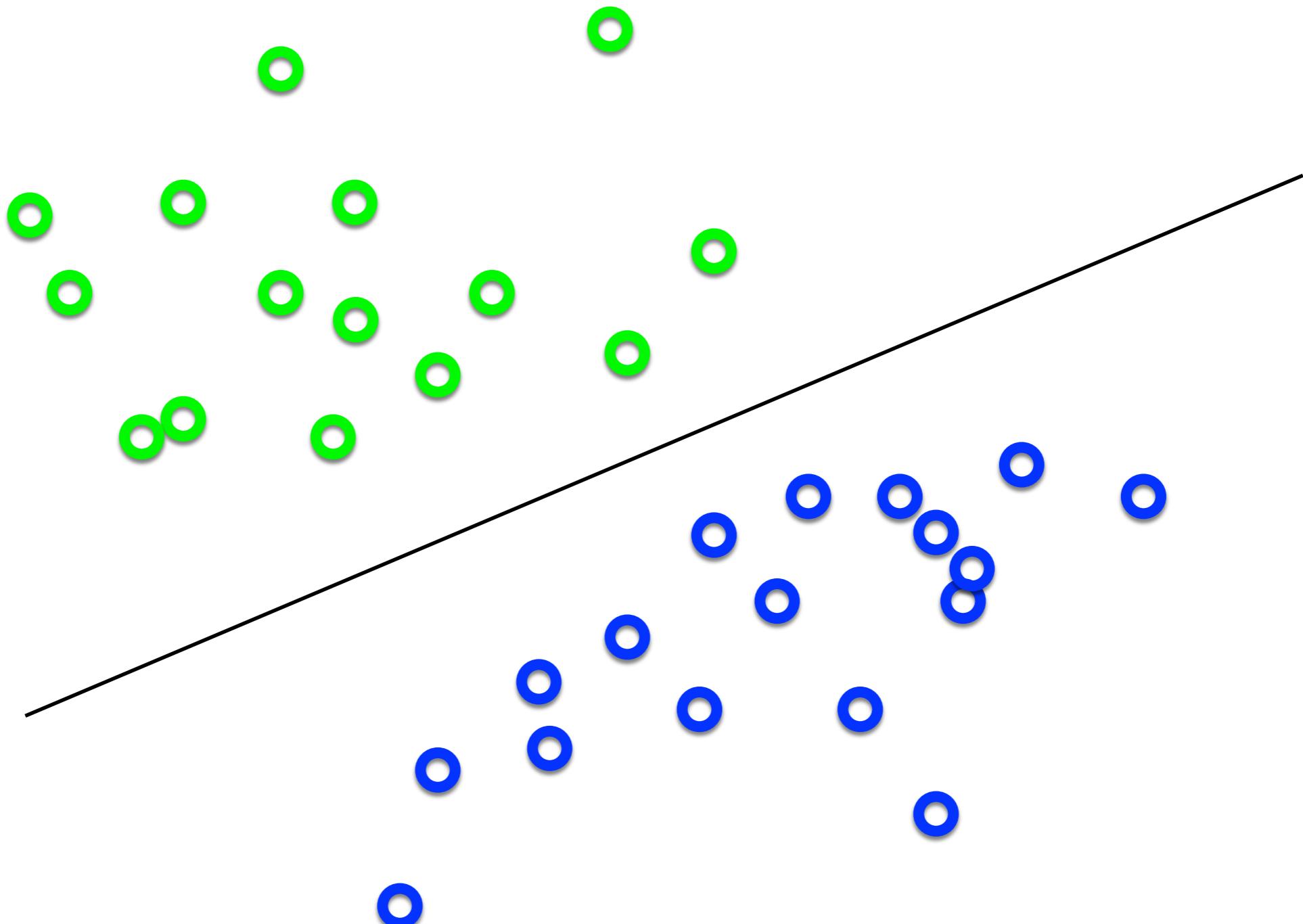


What's the best \mathbf{w} ?



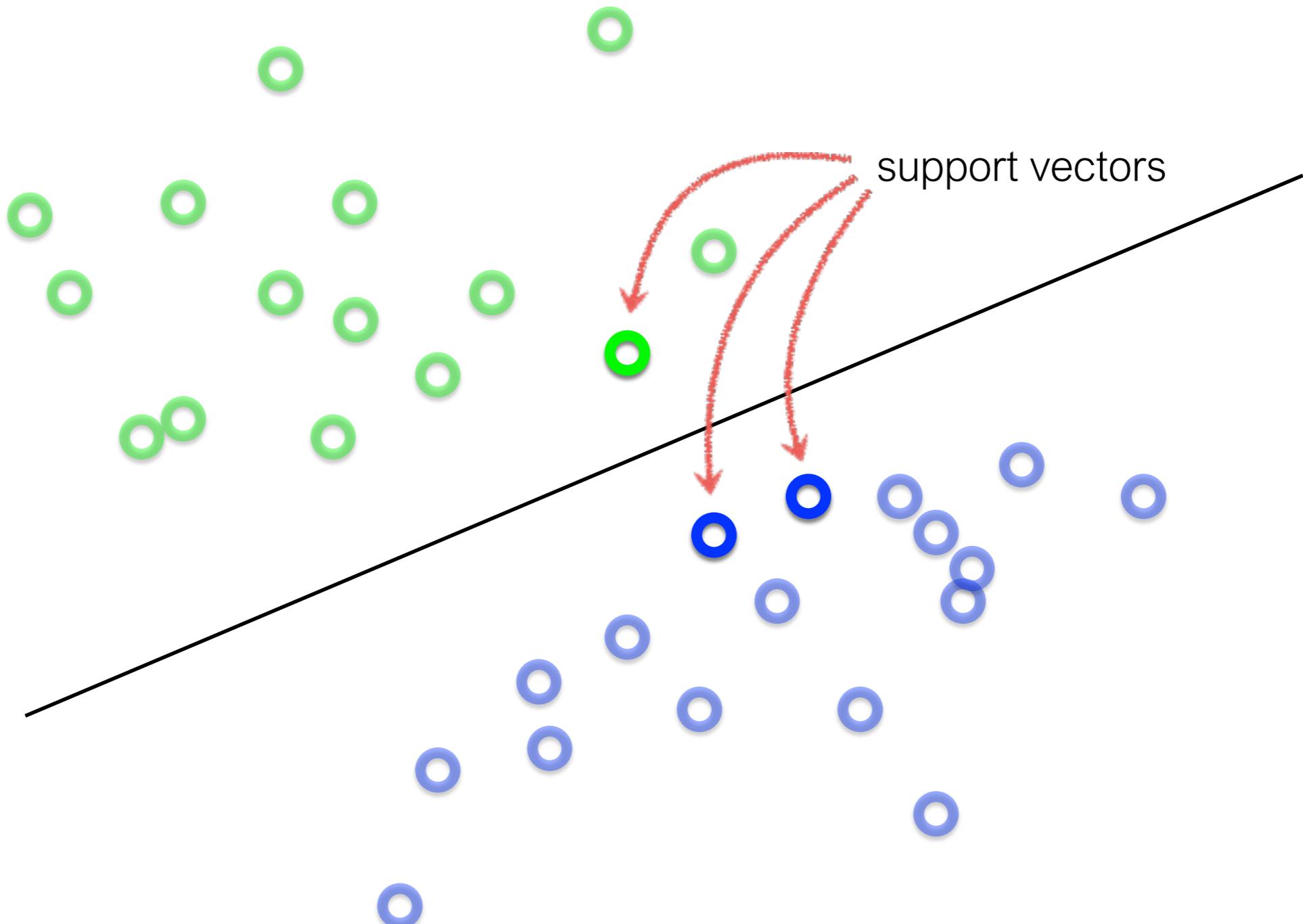
Intuitively, the line that is the
farthest from all interior points

What's the best \mathbf{w} ?



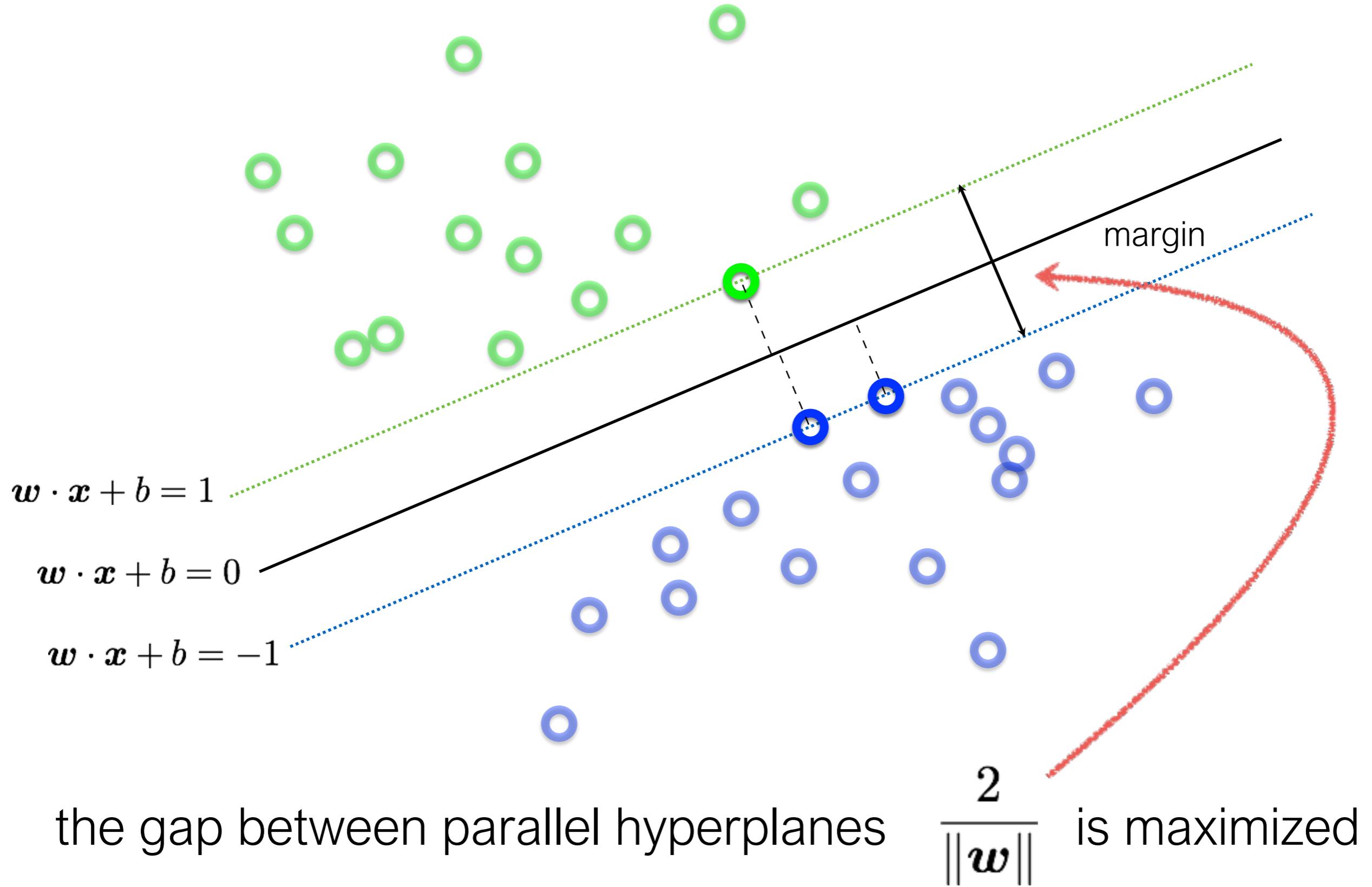
Maximum Margin solution:
most stable to perturbations of data

What's the best \mathbf{w} ?



Want a hyperplane that is far away from ‘inner points’

Find hyperplane \mathbf{w} such that ...



Can be formulated as a maximization problem

$$\max_{\mathbf{w}} \frac{2}{\|\mathbf{w}\|}$$

$$\text{subject to } \mathbf{w} \cdot \mathbf{x}_i + b \begin{cases} \geq +1 & \text{if } y_i = +1 \\ \leq -1 & \text{if } y_i = -1 \end{cases} \text{ for } i = 1, \dots, N$$

What does this constraint mean?



label of the data point

Why is it $+1$ and -1 ?

Can be formulated as a maximization problem

$$\max_{\mathbf{w}} \frac{2}{\|\mathbf{w}\|}$$

subject to $\mathbf{w} \cdot \mathbf{x}_i + b \begin{cases} \geq +1 & \text{if } y_i = +1 \\ \leq -1 & \text{if } y_i = -1 \end{cases}$ for $i = 1, \dots, N$

Equivalently,

Where did the 2 go?

$$\min_{\mathbf{w}} \|\mathbf{w}\|$$

subject to $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$ for $i = 1, \dots, N$

What happened to the labels?

‘Primal formulation’ of a linear SVM

$$\min_{\mathbf{w}} \|\mathbf{w}\|$$

Objective Function

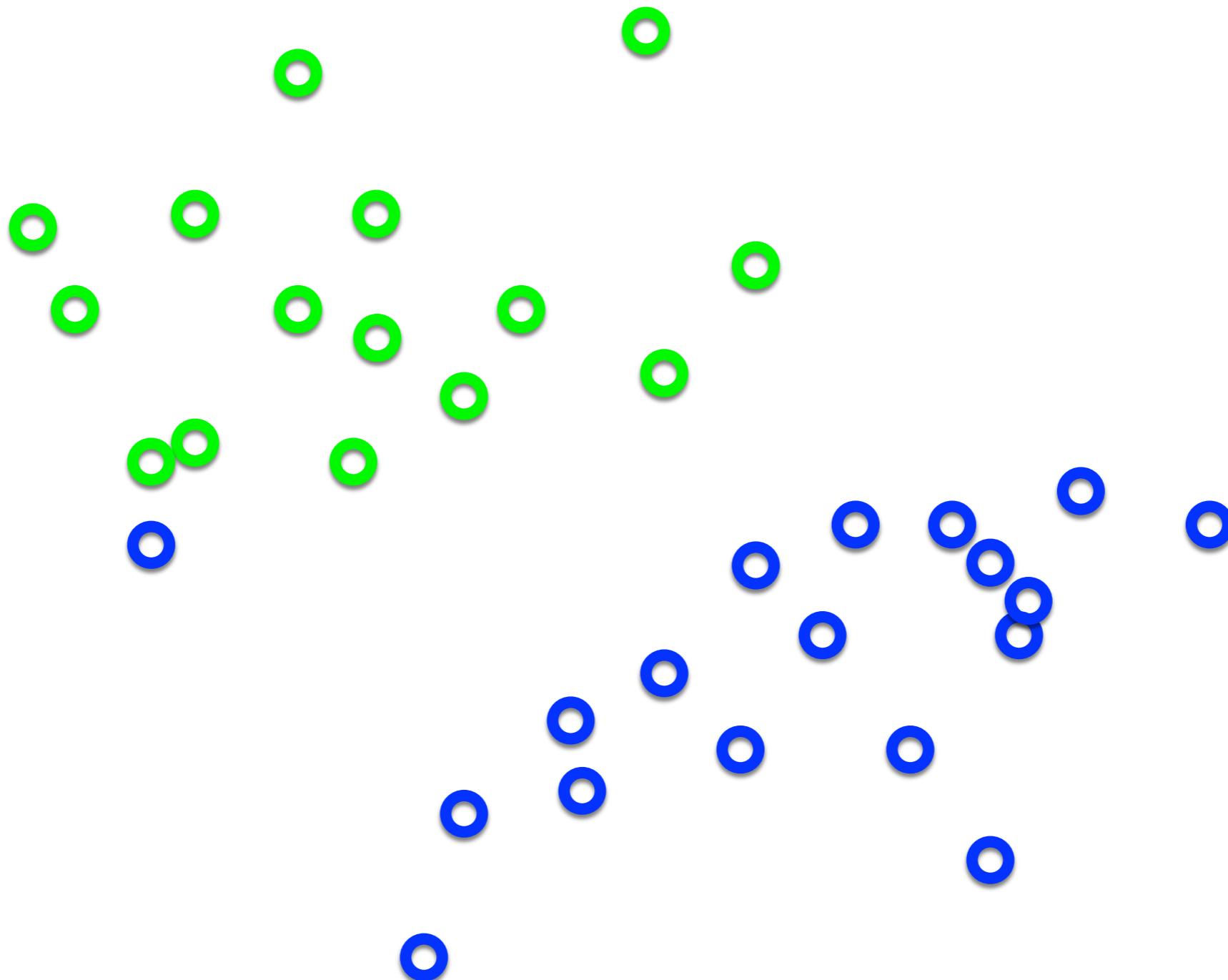
subject to $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$ for $i = 1, \dots, N$

Constraints

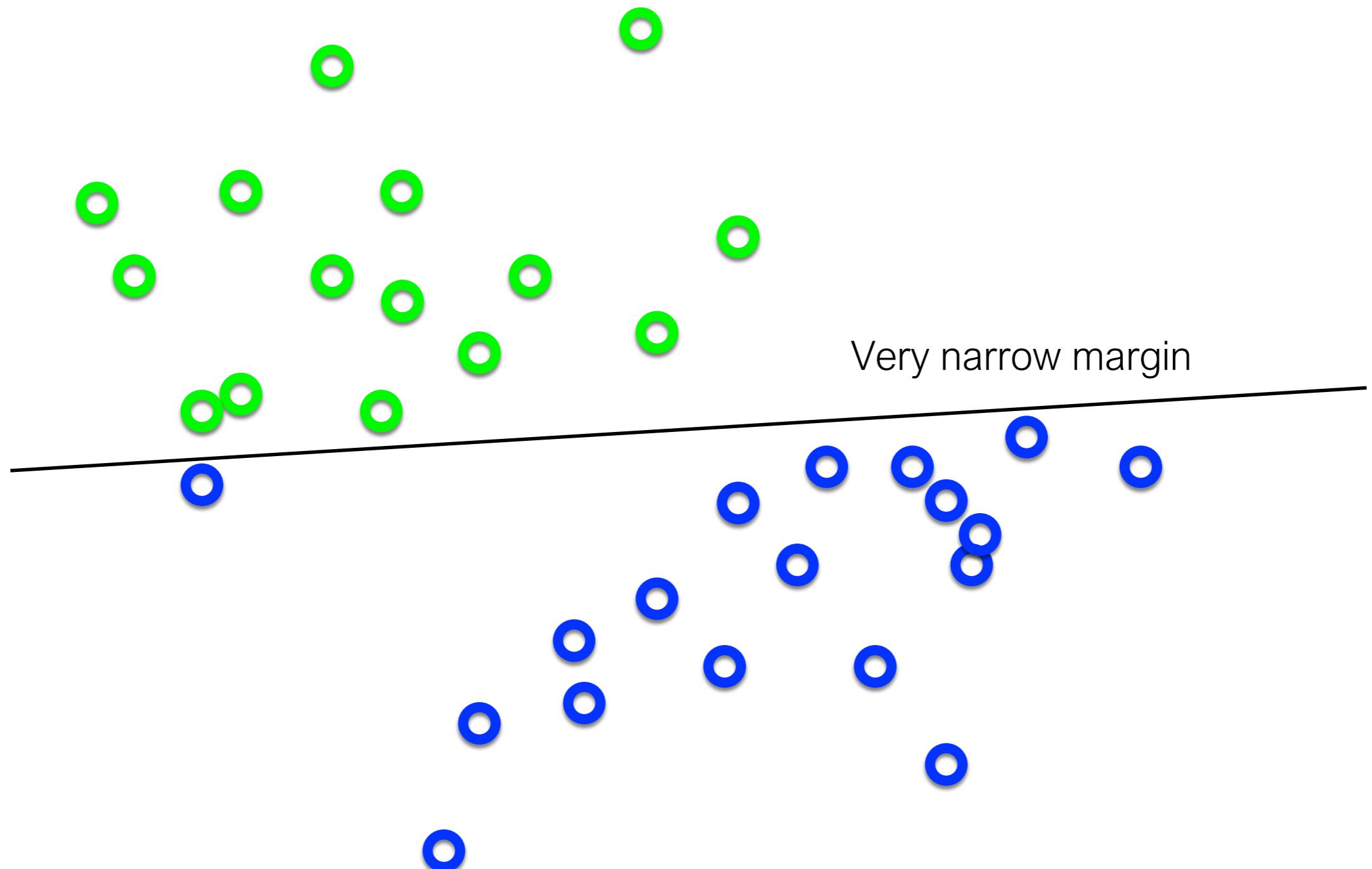
This is a convex quadratic programming (QP) problem
(a unique solution exists)

‘soft’ margin

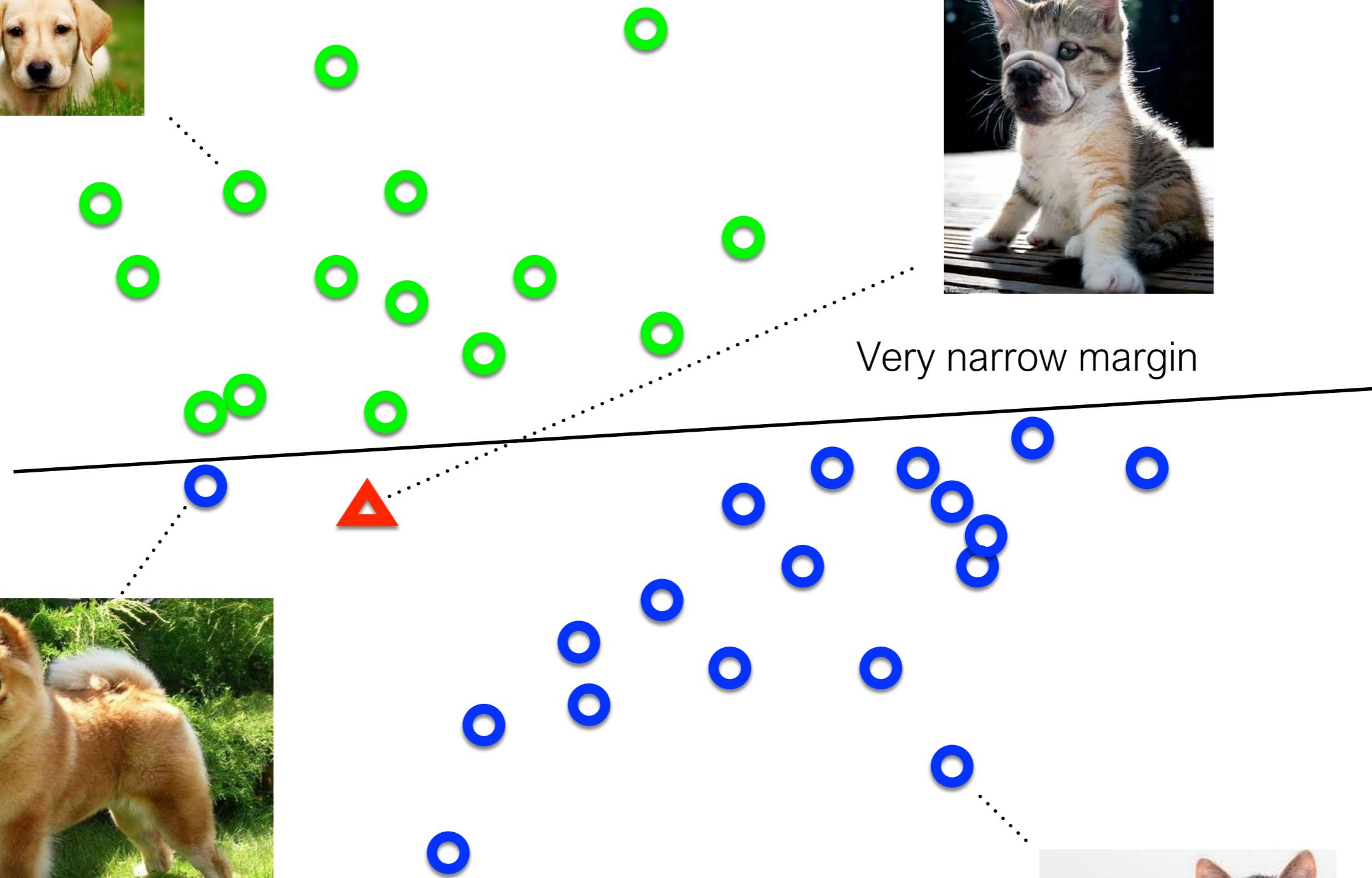
What's the best **w**?



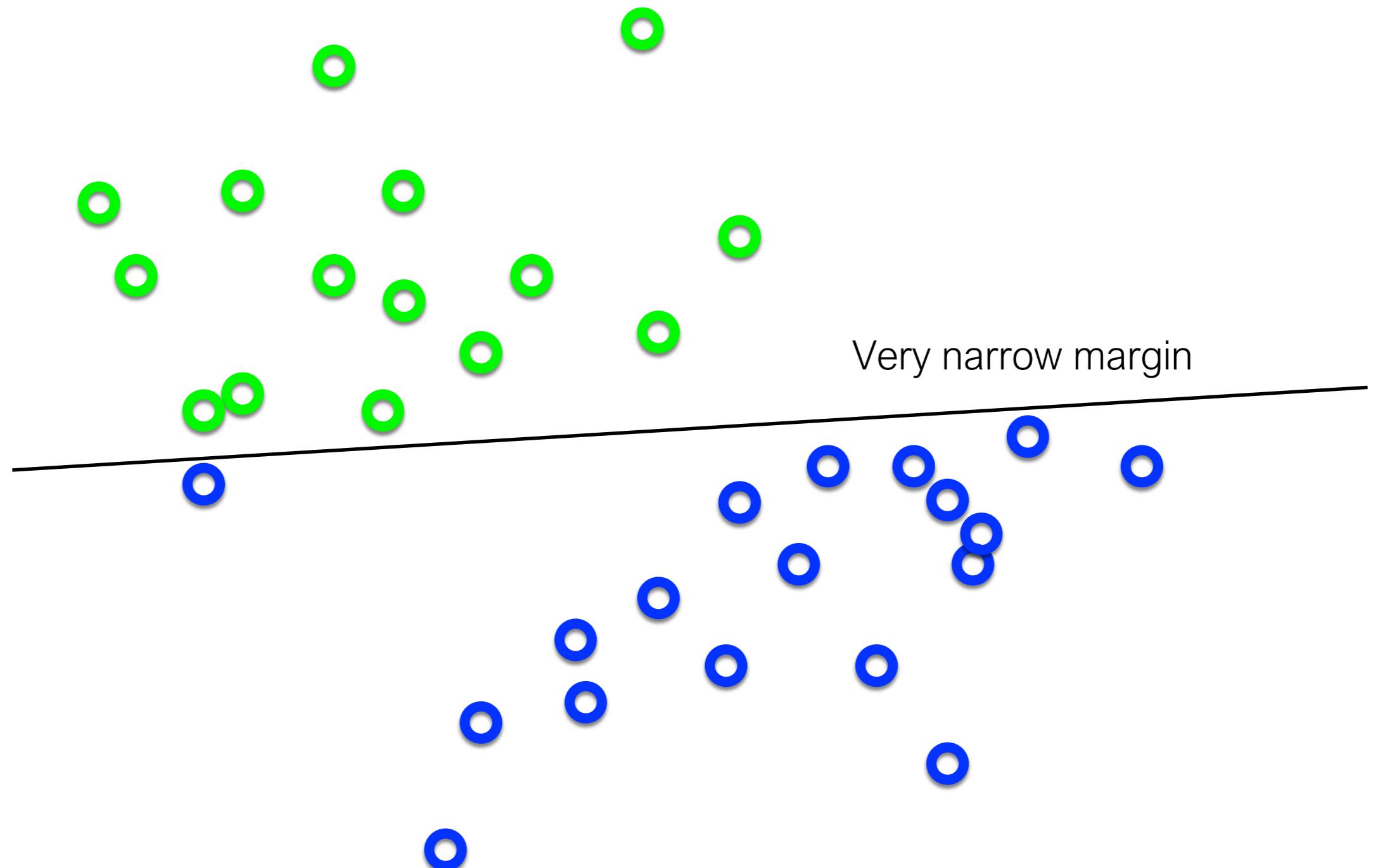
What's the best \mathbf{w} ?



Separating cats and dogs

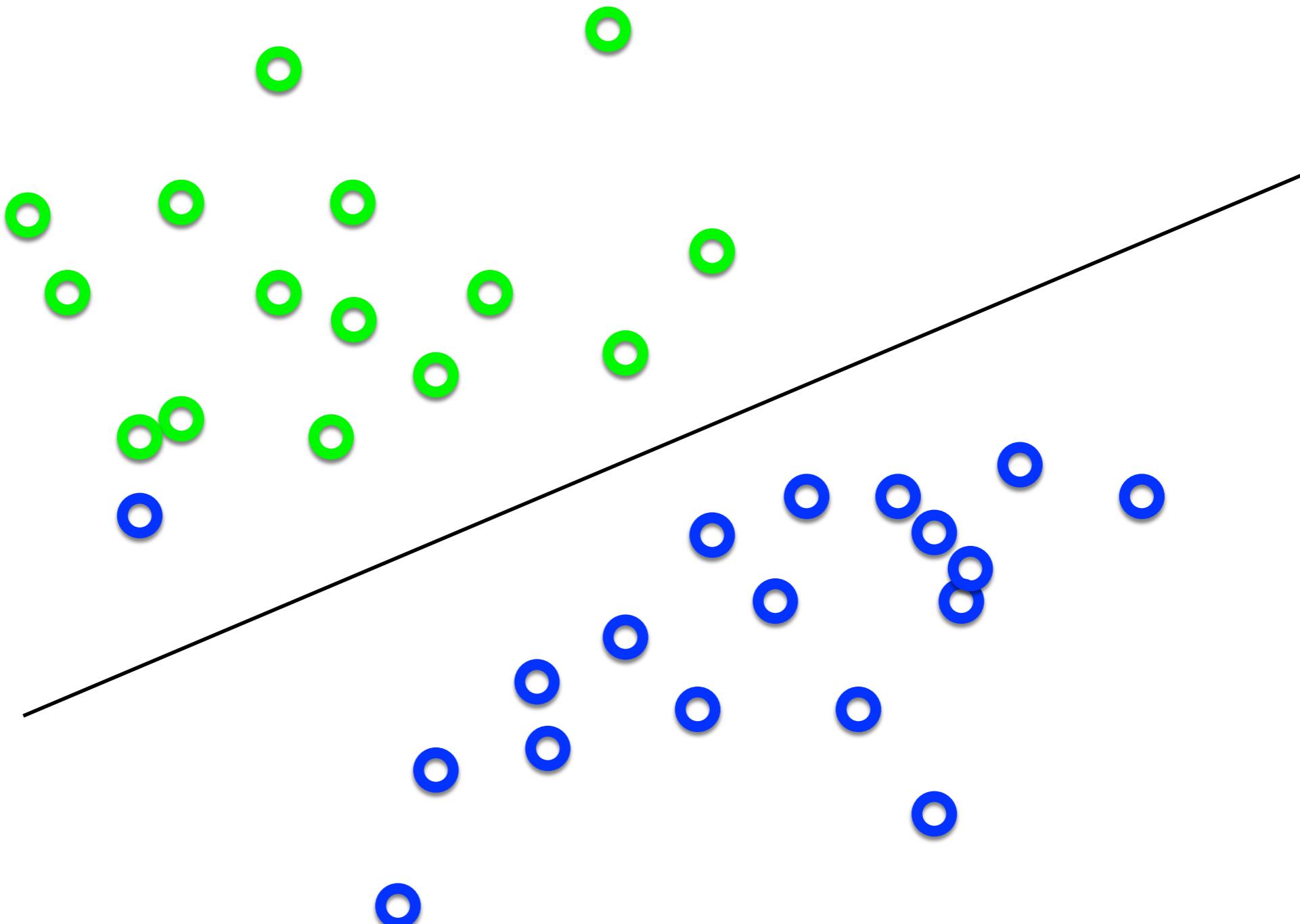


What's the best \mathbf{w} ?



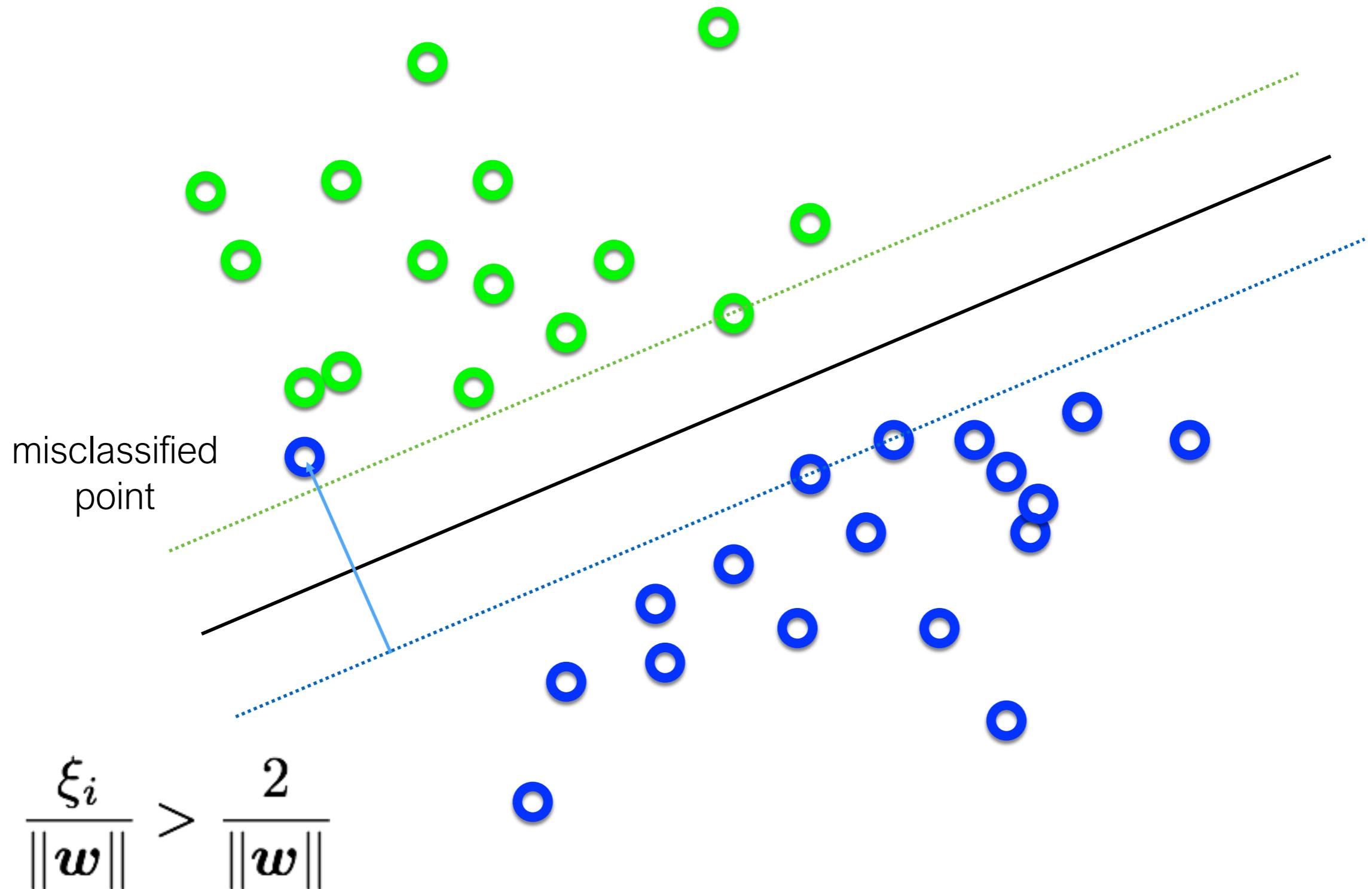
Intuitively, we should allow for some misclassification if we can get more robust classification

What's the best **w**?



Trade-off between the MARGIN and the MISTAKES
(might be a better solution)

Adding slack variables $\xi_i \geq 0$



‘soft’ margin

objective

$$\min_{\boldsymbol{w}, \boldsymbol{\xi}} \|\boldsymbol{w}\|^2 + C \sum_i \xi_i$$

subject to

$$y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 - \xi_i$$

for $i = 1, \dots, N$

‘soft’ margin

objective

$$\min_{\mathbf{w}, \xi} \|\mathbf{w}\|^2 + C \sum_i \xi_i$$

subject to

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i \quad \text{for } i = 1, \dots, N$$

The slack variable allows for mistakes,
as long as the inverse margin is minimized.

‘soft’ margin

objective

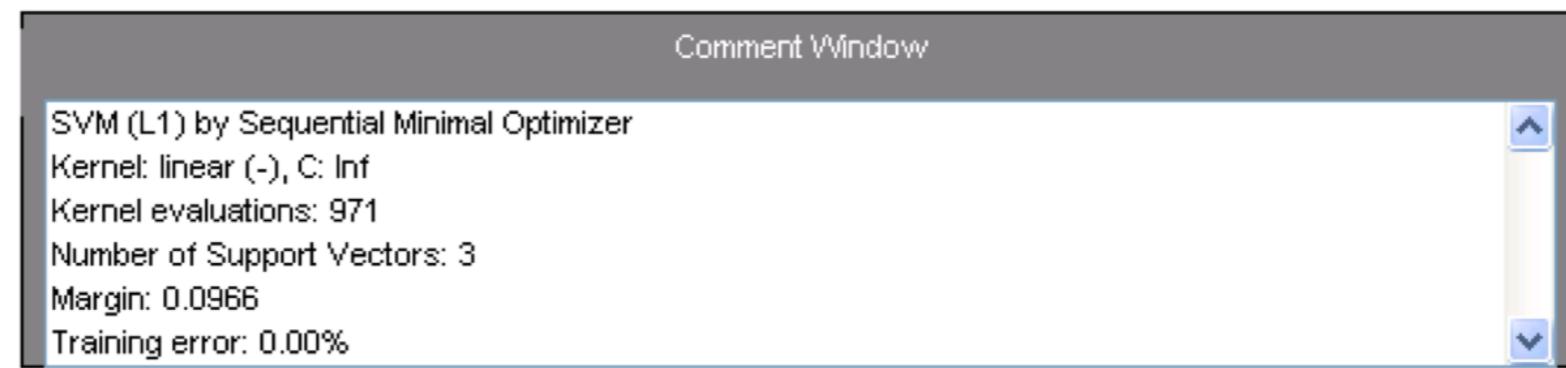
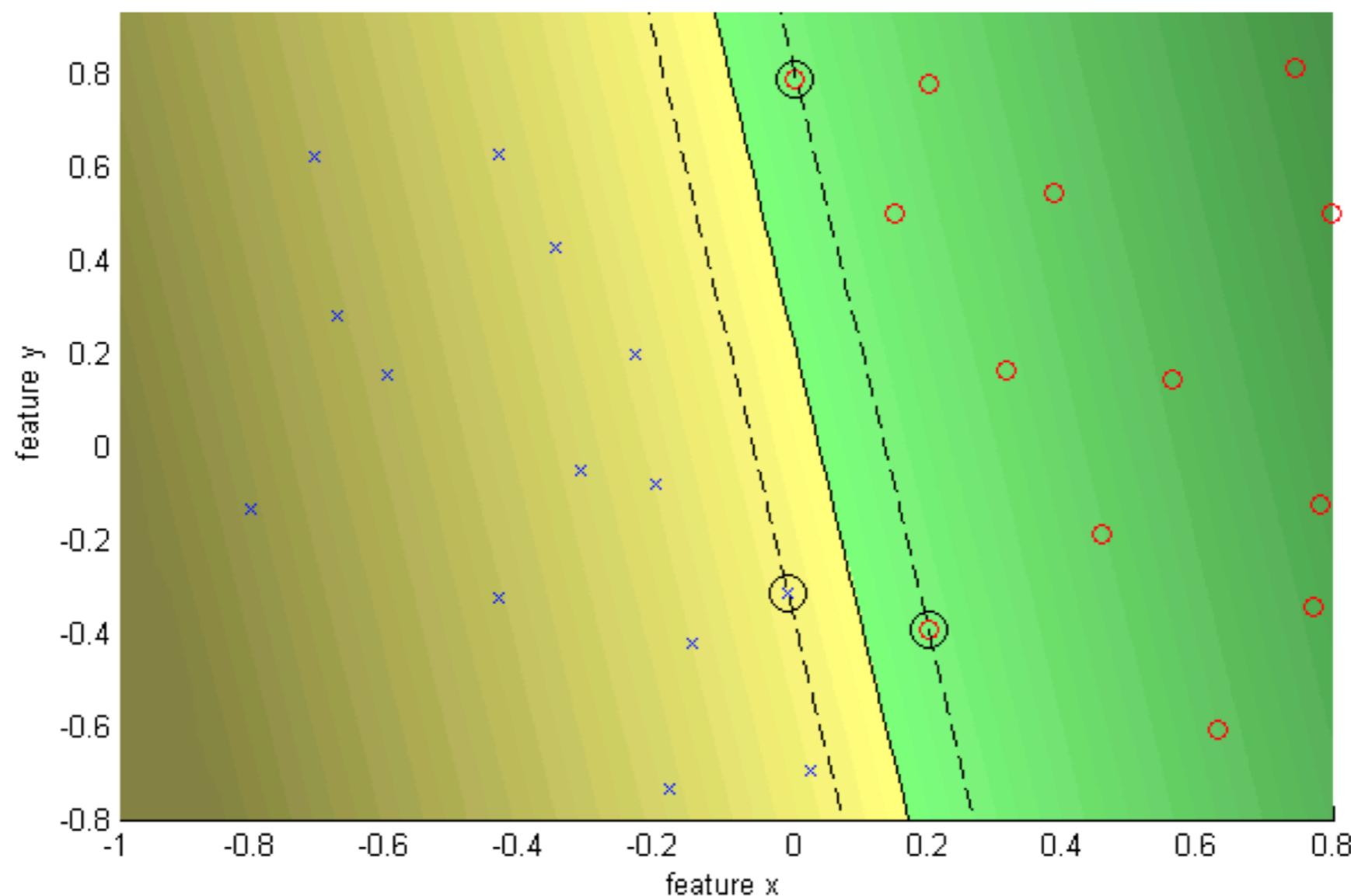
$$\min_{\mathbf{w}, \boldsymbol{\xi}} \|\mathbf{w}\|^2 + C \sum_i \xi_i$$

subject to

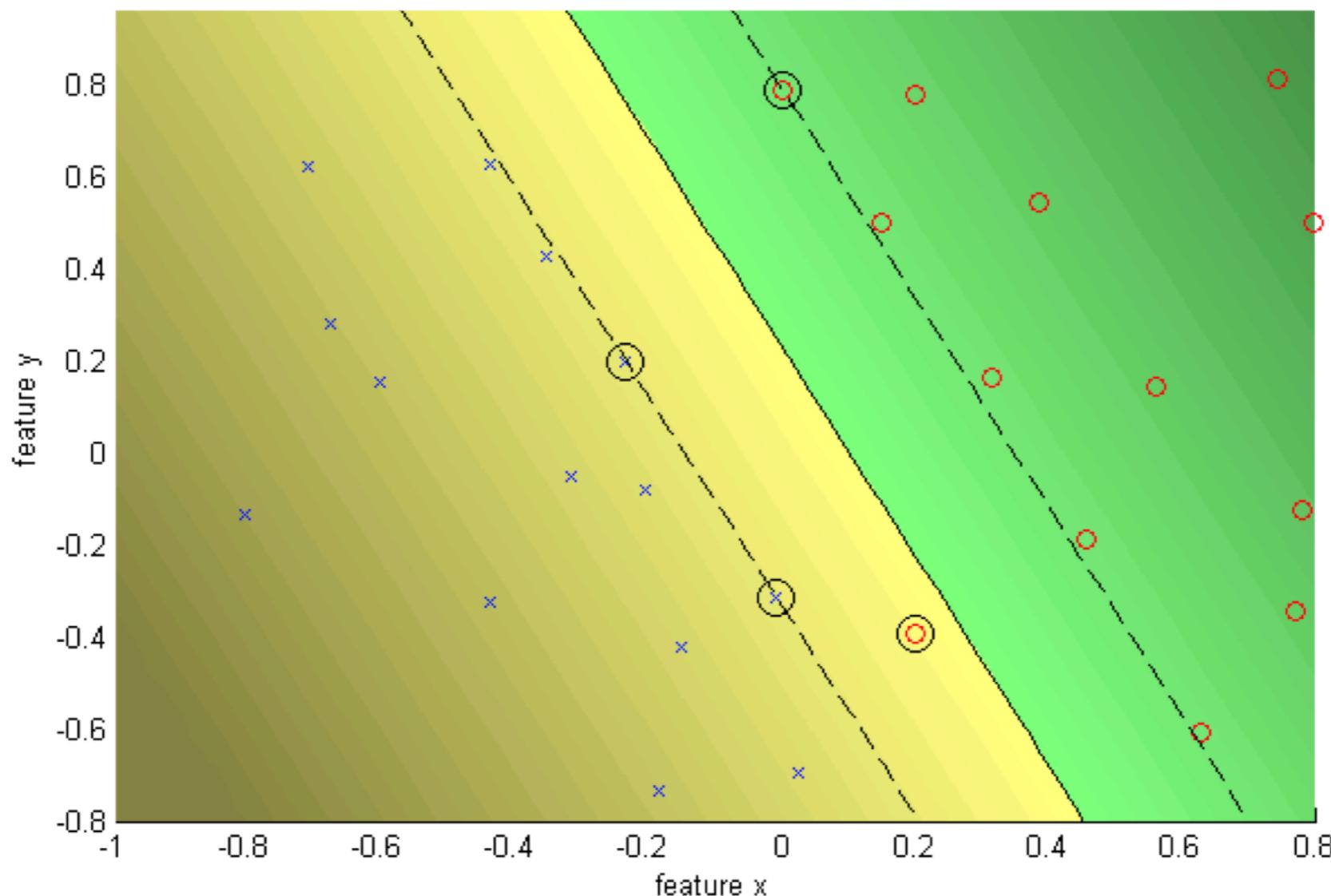
$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i \quad \text{for } i = 1, \dots, N$$

- Every constraint can be satisfied if slack is large
- C is a regularization parameter
 - Small C: ignore constraints (larger margin)
 - Big C: constraints (small margin)
- Still QP problem (unique solution)

$C = \text{Infinity}$ hard margin



$C = 10$ soft margin



Comment Window

SVM (L1) by Sequential Minimal Optimizer
Kernel: linear (-), C: 10.0000
Kernel evaluations: 2645
Number of Support Vectors: 4
Margin: 0.2265
Training error: 3.70%

Bag of words applications: Video google

Example queries on the movie 'Run Lola Run'

Object query

Query frame with outlined query region and its close-up



Retrieved key-frames from three different shots



Bag of words applications: Video google

A tie query in the movie 'Groundhog Day'

Query



Query frame with the outlined query region

Close up of the region

Examples of retrieved results

Key-frames from first 15 retrieved shots, one key-frame per shot.
right column shows the close-up of the region.



Bag of words applications: Photo Tourism

Photo Tourism

Exploring photo collections in 3D

Noah Snavely Steven M. Seitz Richard Szeliski
University of Washington *Microsoft Research*

SIGGRAPH 2006

References

Basic reading:

- Szeliski, Chapter 14.