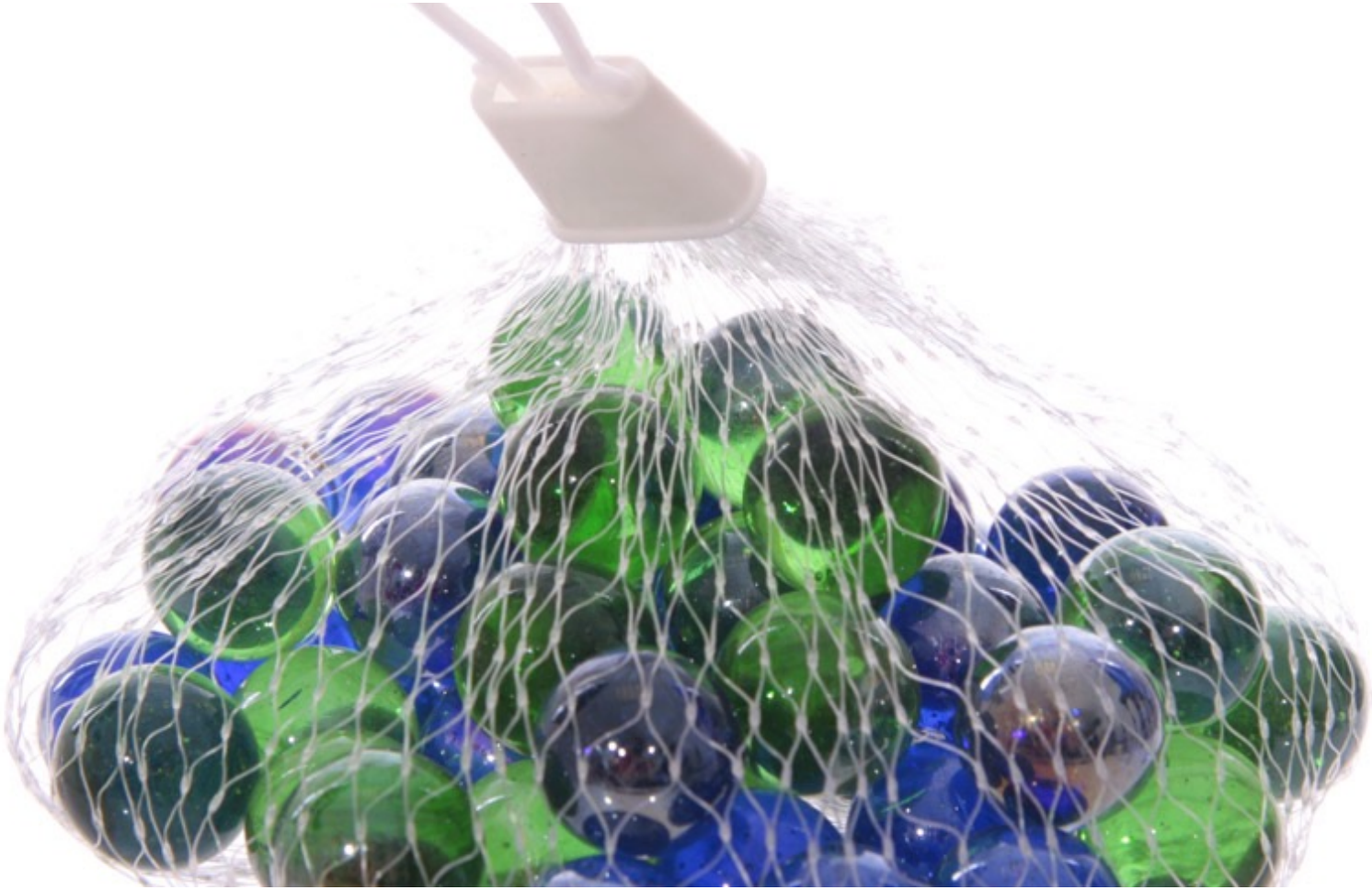
Image classification (II)

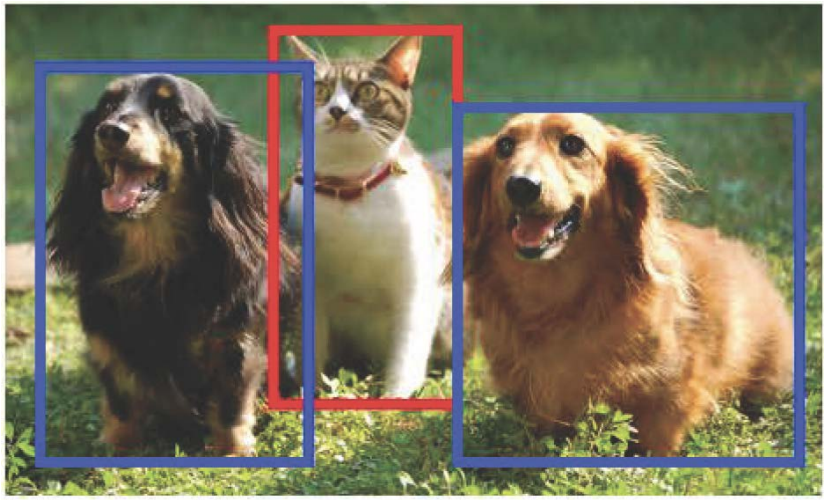


Computer Vision
Fall 2022, Lecture 11

Classification

CAT

Object Detection



CAT, DOG

Instance Segmentation

CAT, DOG

Data-driven approach

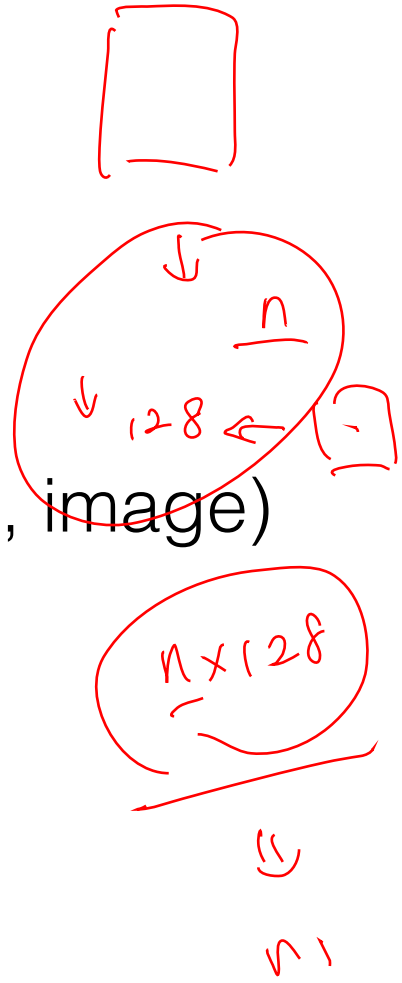
- Collect a database of images with labels
- Use ML to train an image classifier
- Evaluate the classifier on test images

```
def train(images, labels):
  # Machine learning!
  return model
```

```
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```

Example training set

Bag-of-Words



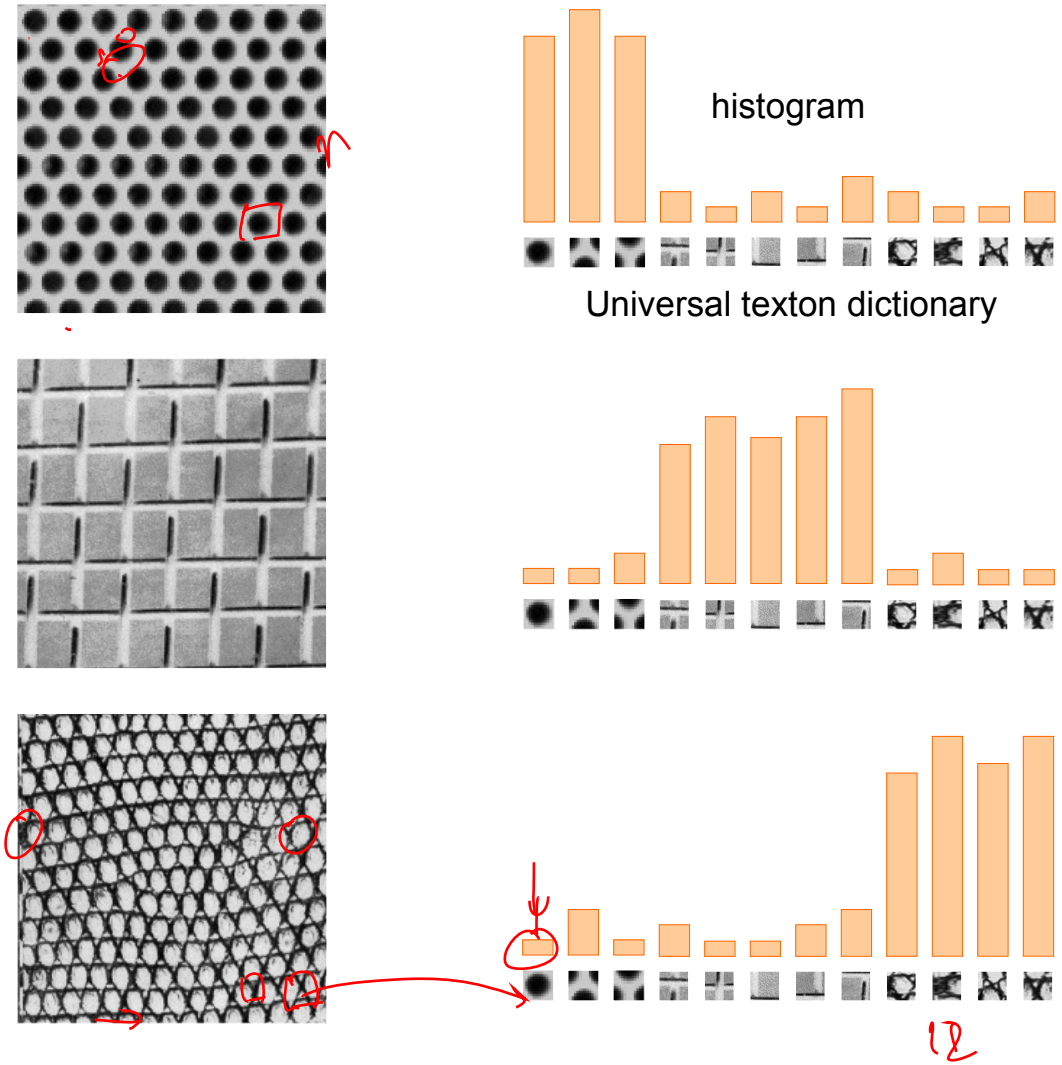
represent a data item (document, texture, image)
as a histogram over words

an old idea

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

Texture recognition

same dictionary



Standard BOW pipeline

(for image classification)

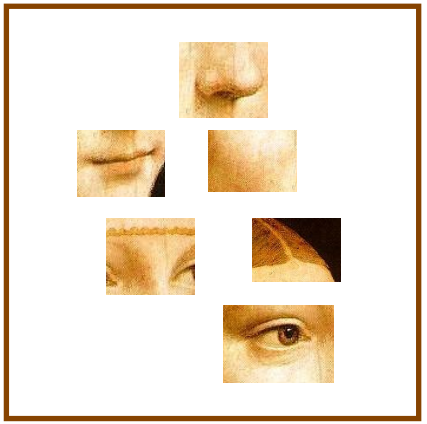
Dictionary Learning:
Learn Visual Words using clustering

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

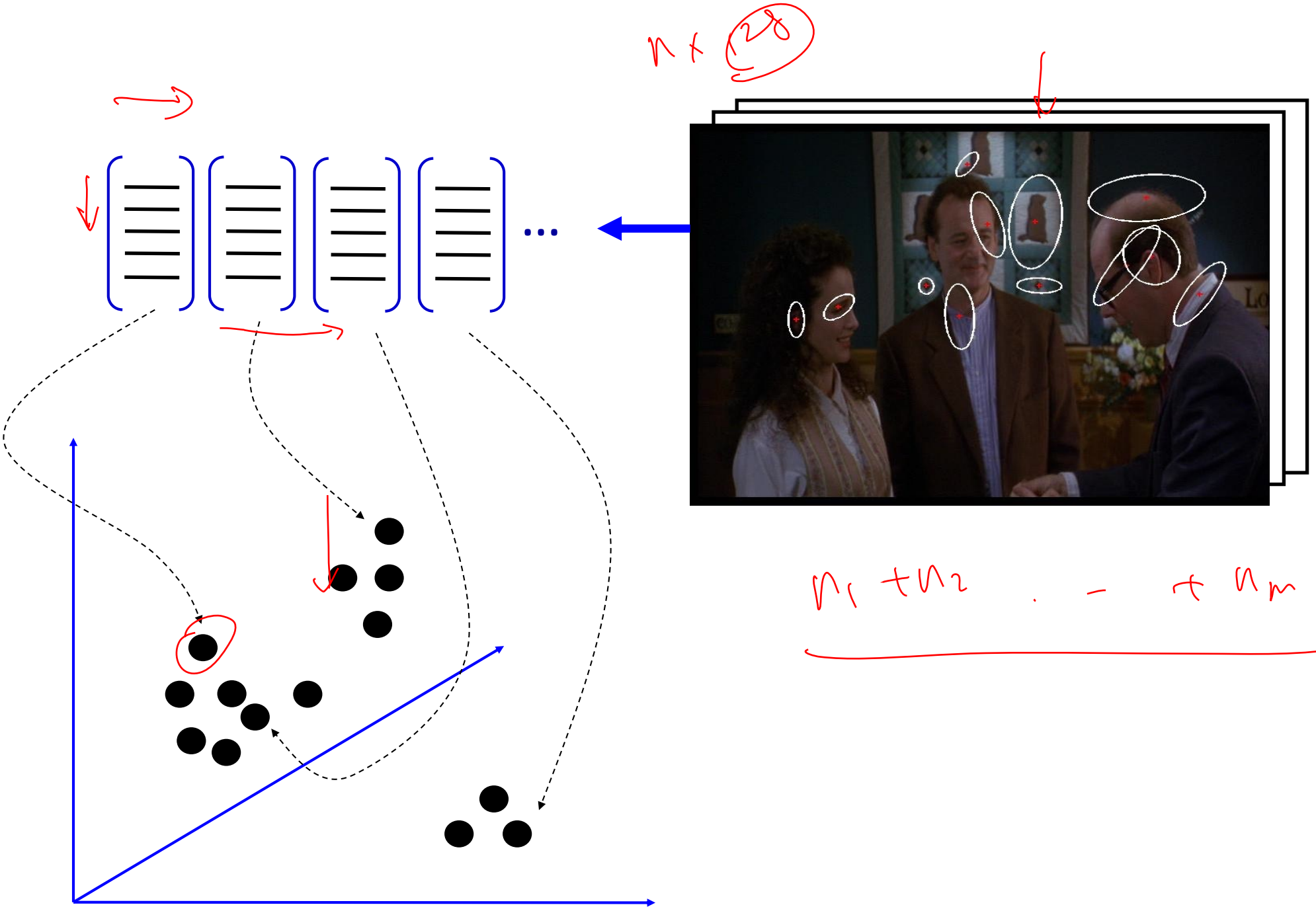


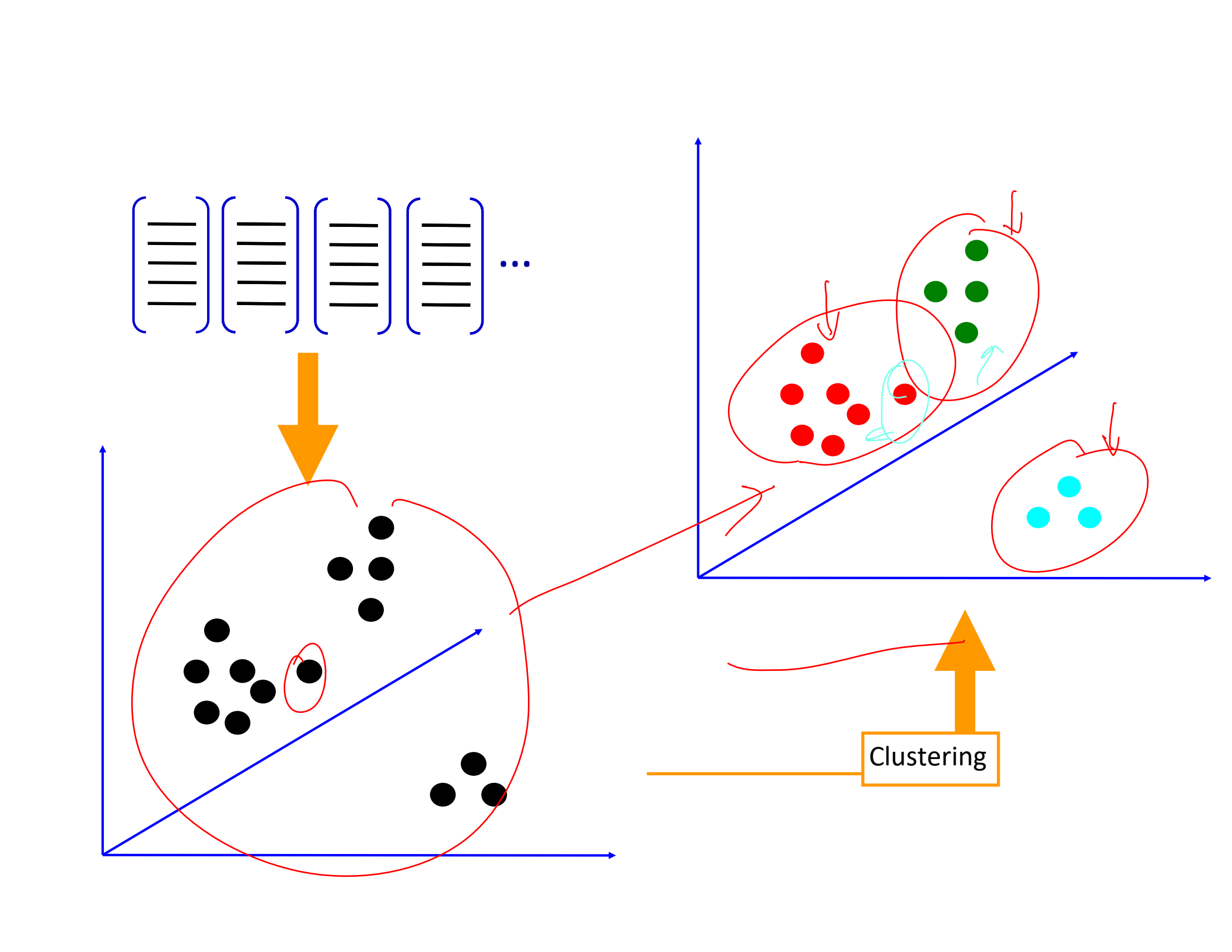
how many per image?

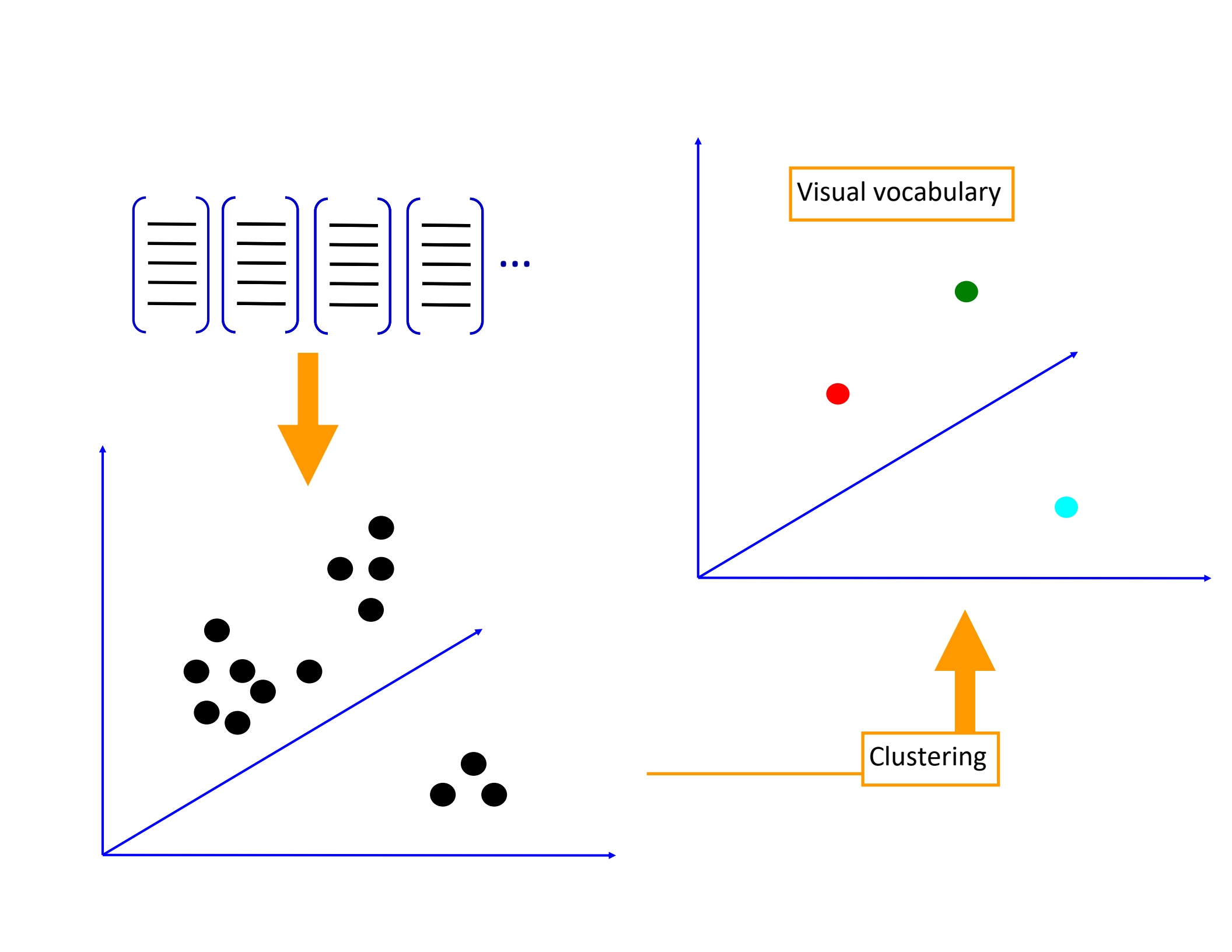
Dictionary Learning:
Learn Visual Words using clustering

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

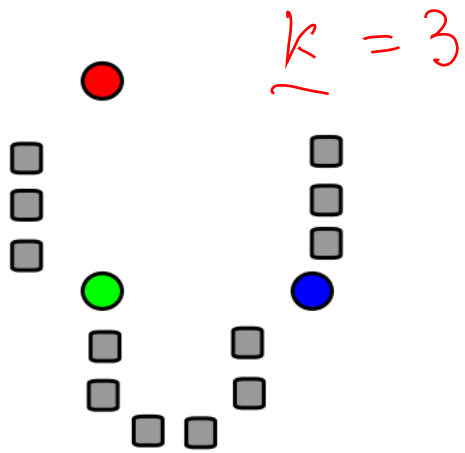
How many words we need?



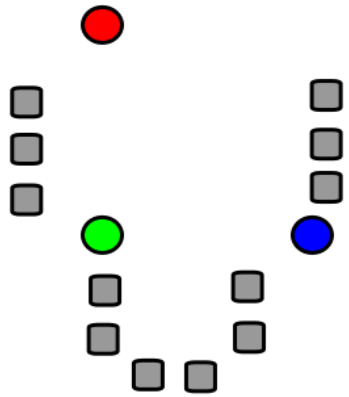

...
Clustering


Visual vocabulary
Clustering

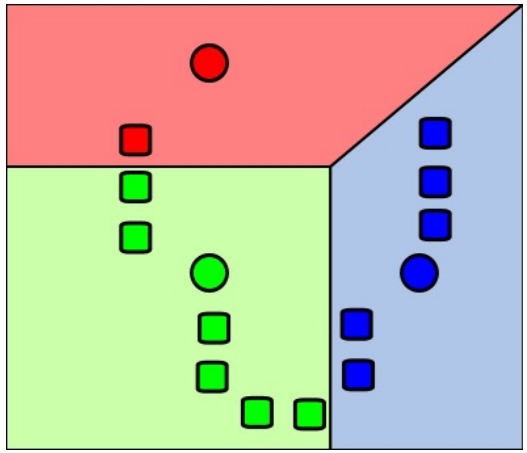
K-means clustering



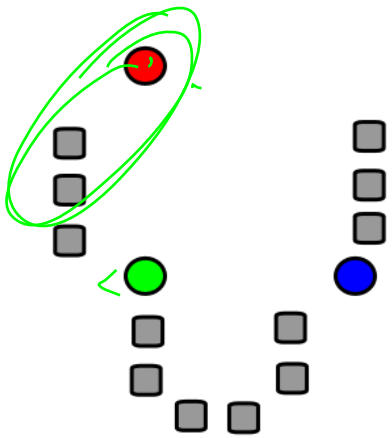
1. Select initial centroids at random



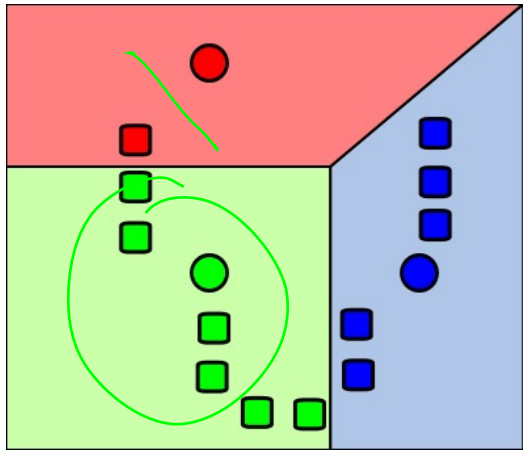
1. Select initial centroids at random



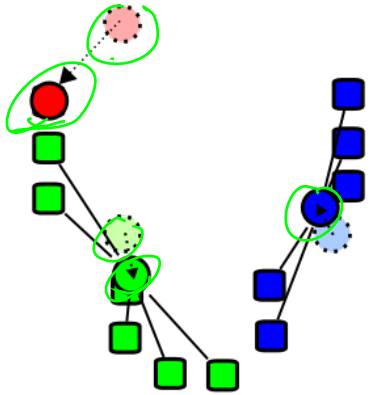
2. Assign each object to the cluster with the nearest centroid.



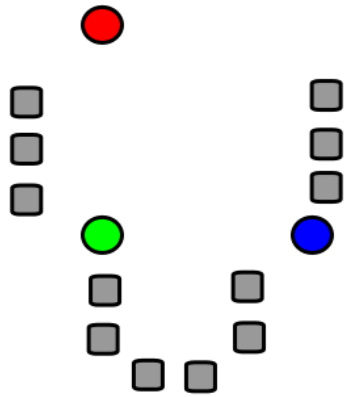
1. Select initial centroids at random



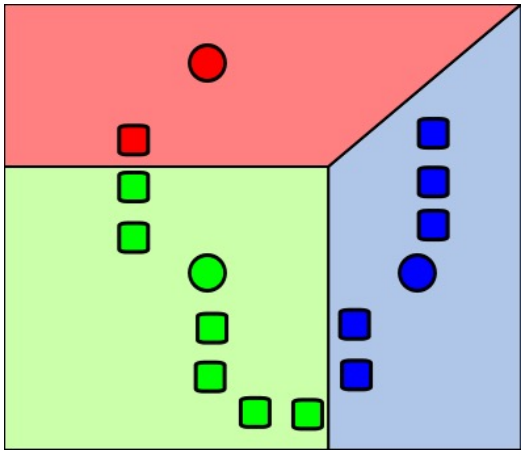
2. Assign each object to the cluster with the nearest centroid.



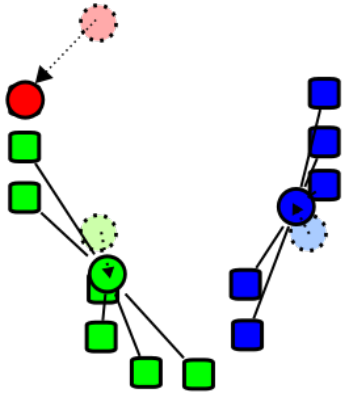
3. Compute each centroid as the mean of the objects assigned to it (go to 2)



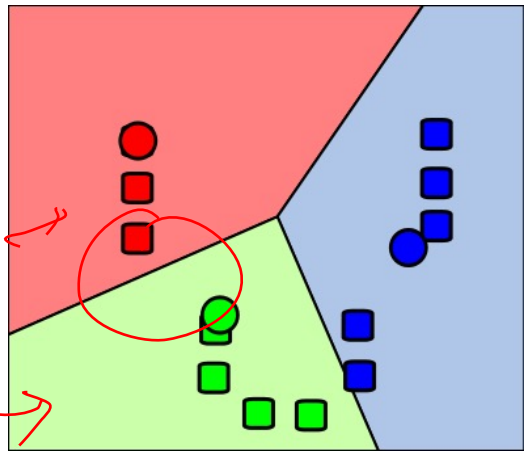
1. Select initial centroids at random



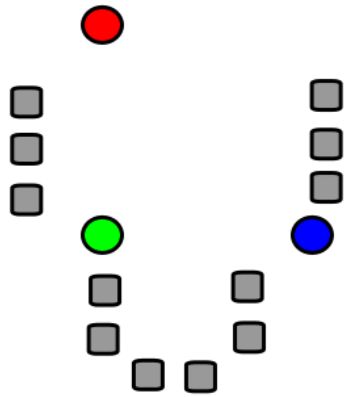
2. Assign each object to the cluster with the nearest centroid.



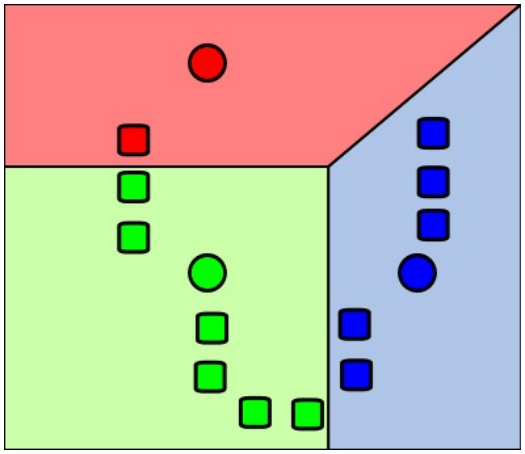
3. Compute each centroid as the mean of the objects assigned to it (go to 2)



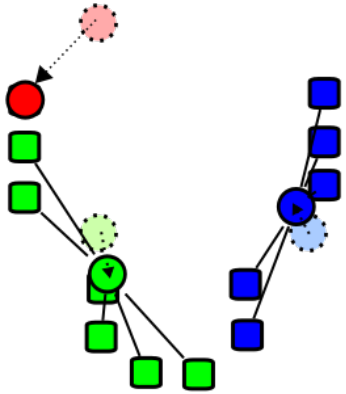
2. Assign each object to the cluster with the nearest centroid.



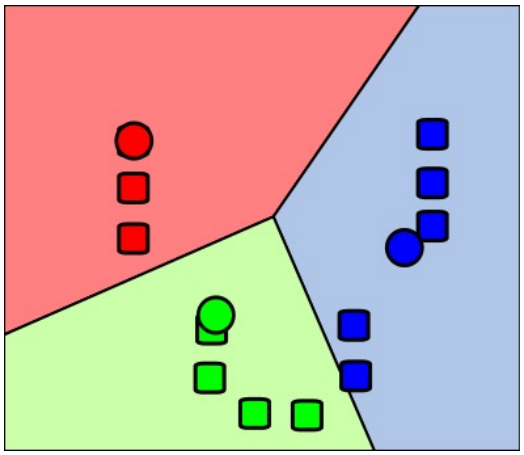
1. Select initial centroids at random



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.

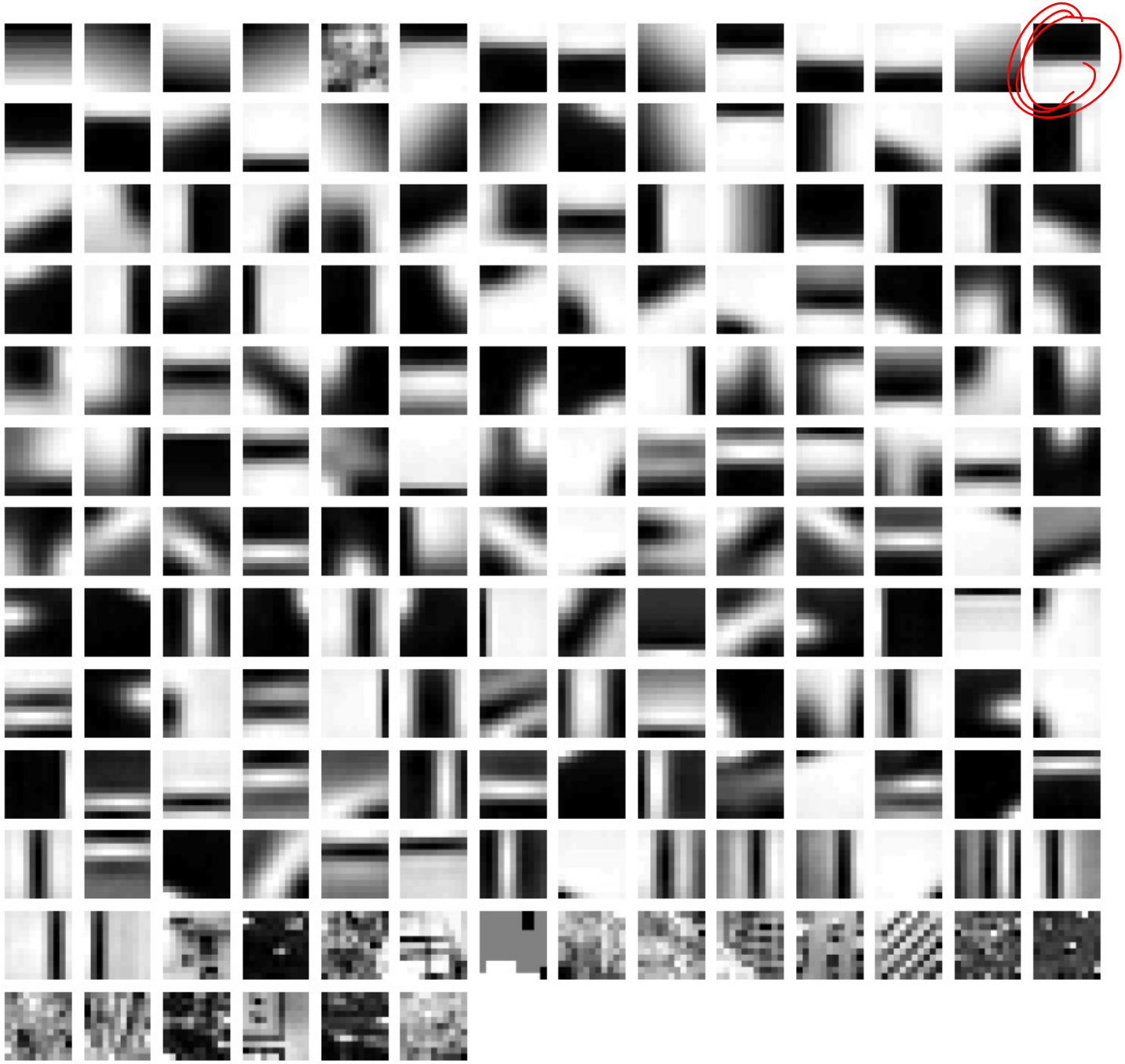
Repeat previous 2 steps until no change

D : $k \times 128$

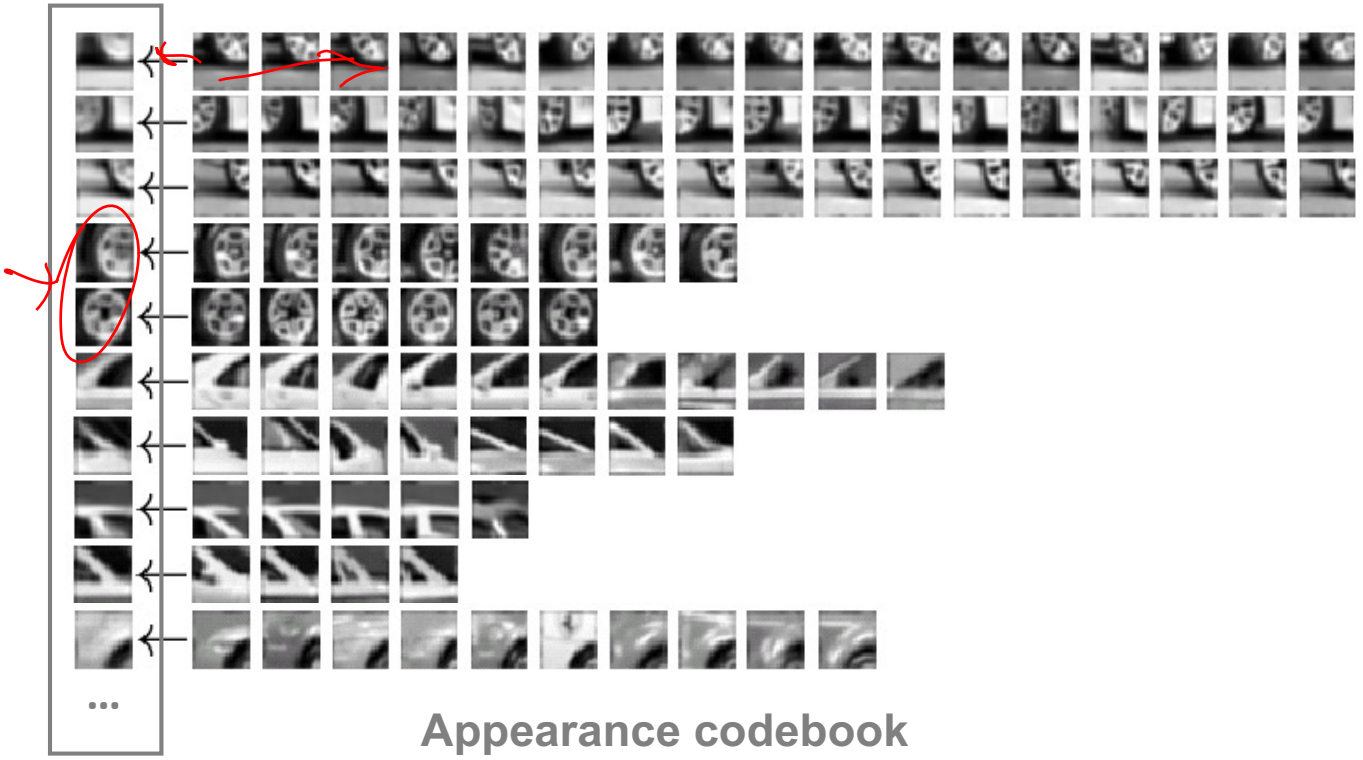
*From what **data** should I learn the dictionary?*

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

Example visual dictionary



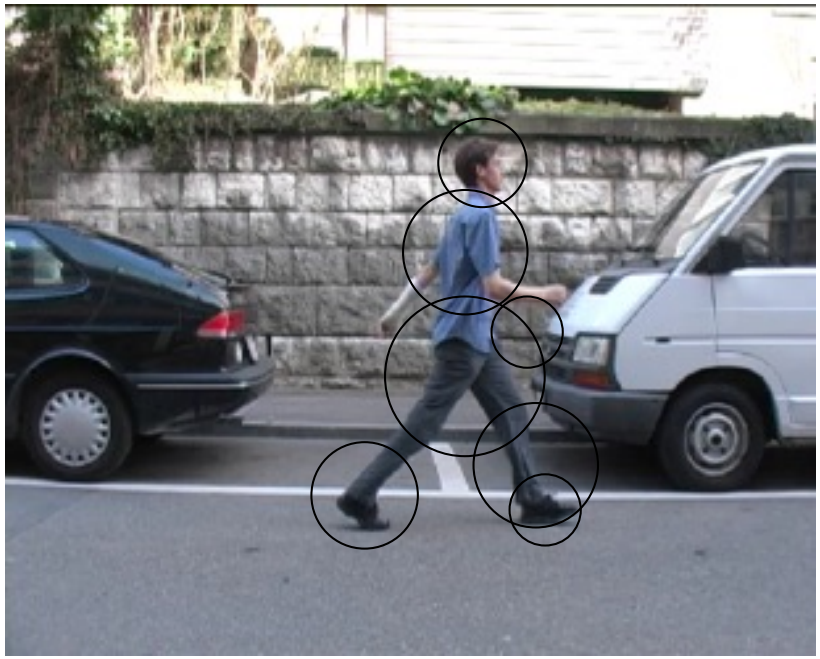
Example dictionary

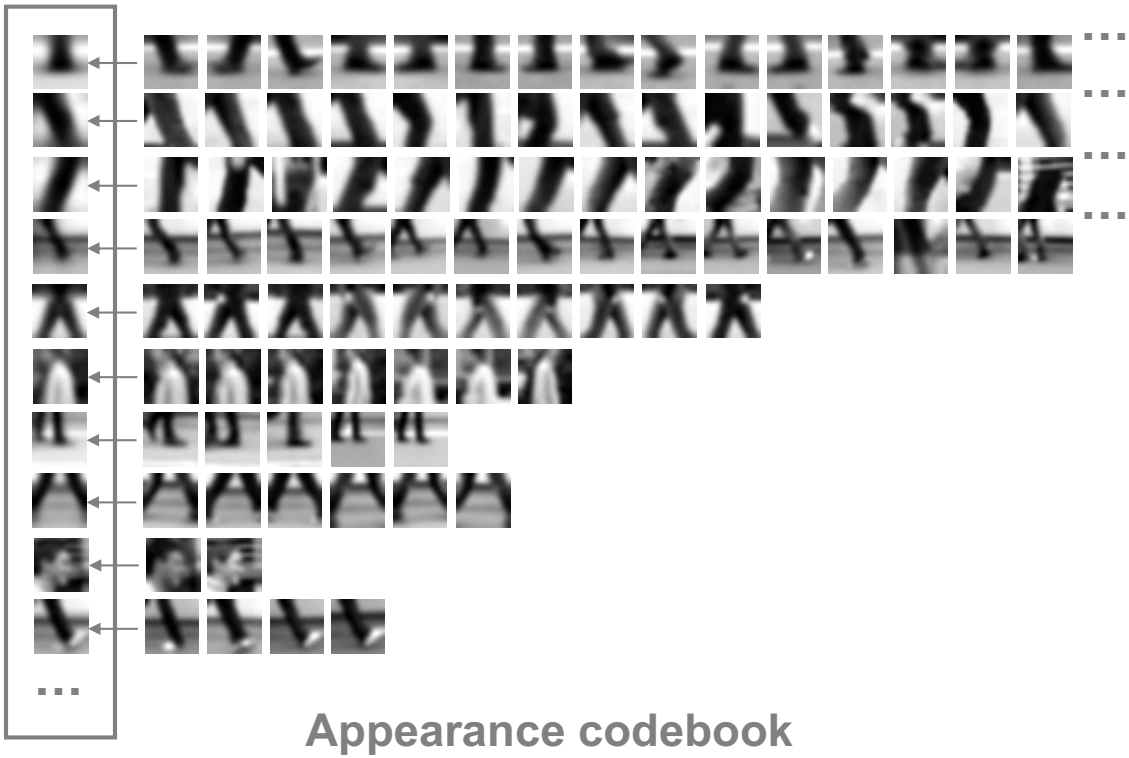


Appearance codebook

Source: B. Leibe

Another dictionary





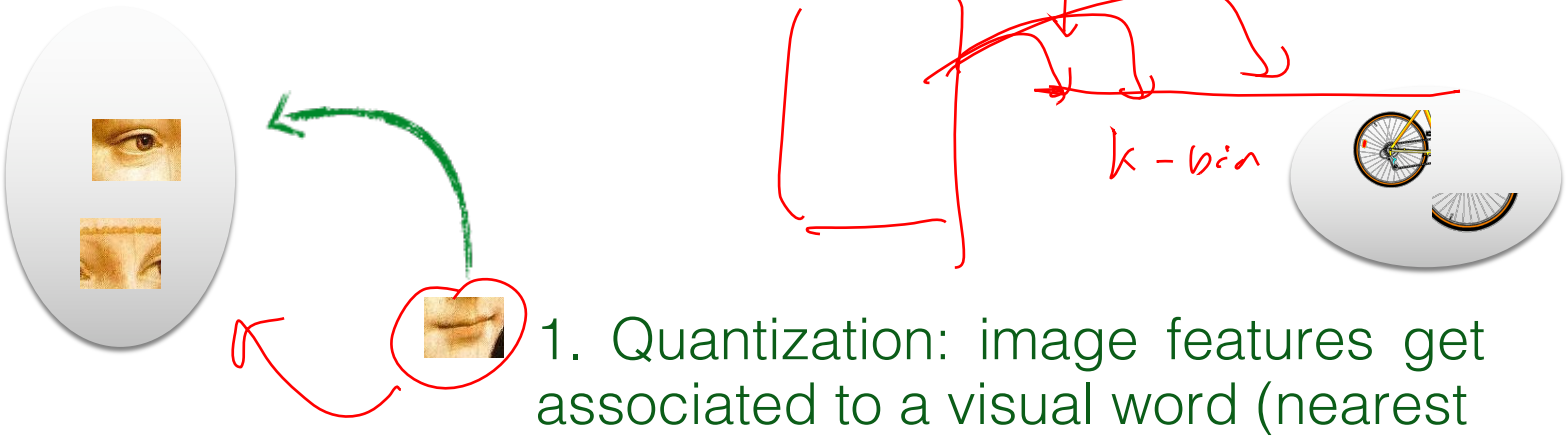
Appearance codebook

Source: B. Leibe

Dictionary Learning:
Learn Visual Words using clustering

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

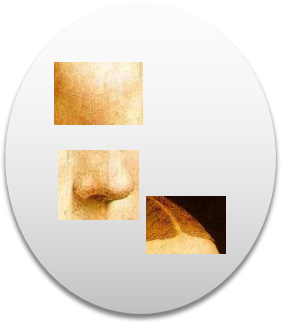
Classify:
Train and test data using BOWs



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

Encode:

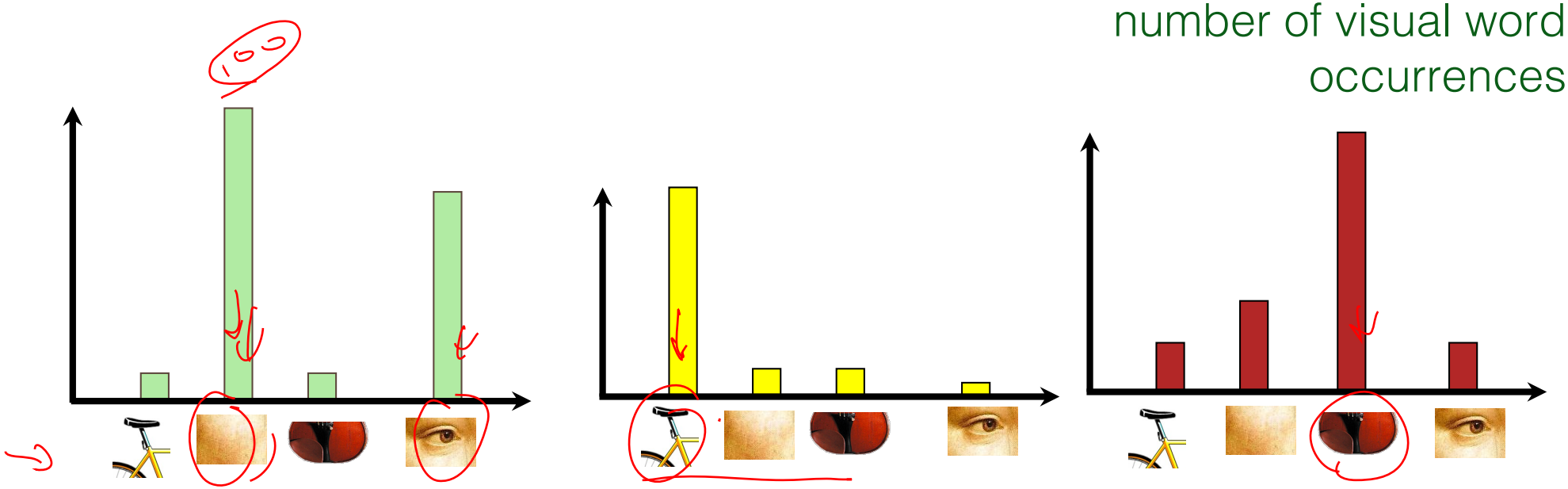
build Bags-of-Words (BOW) vectors for each image

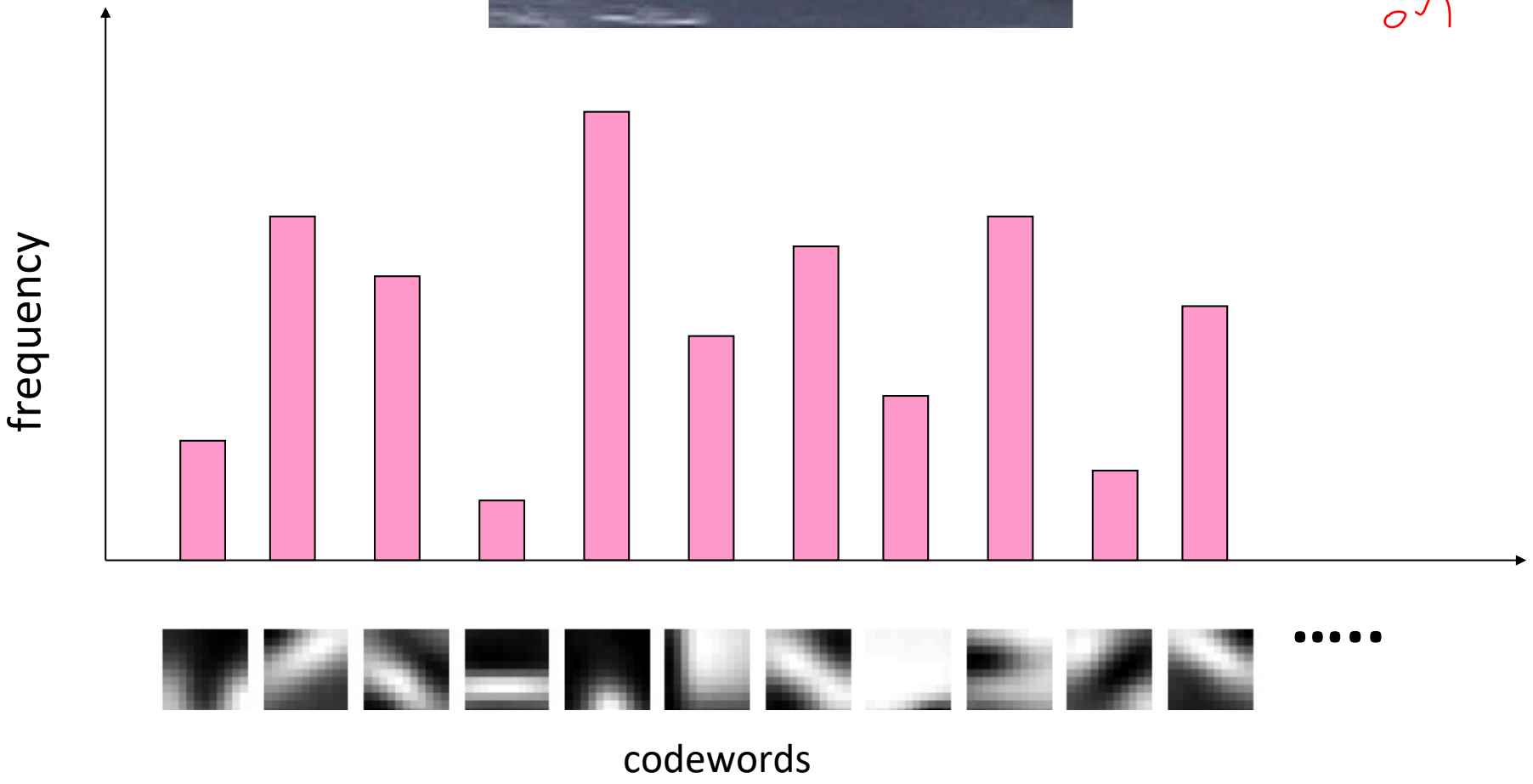


Encode:

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

2. Histogram: count the number of visual word occurrences

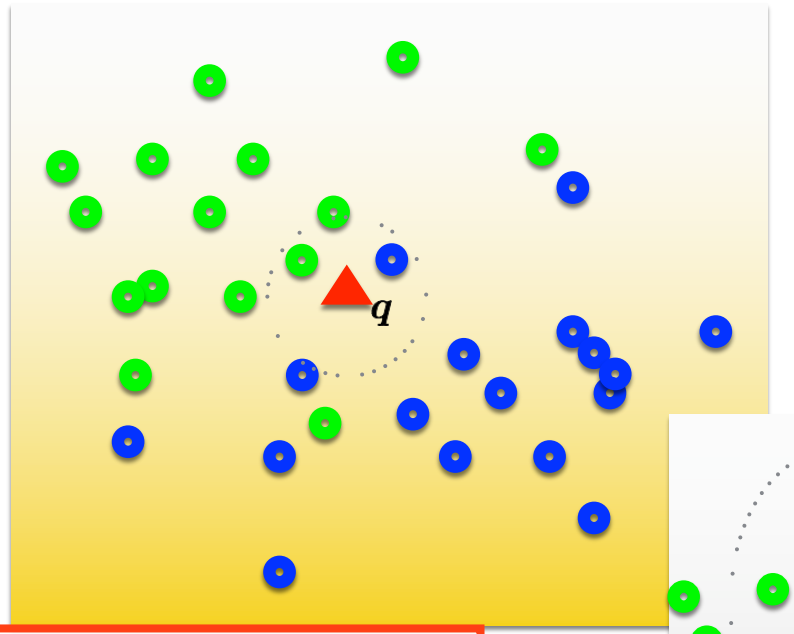



frequency
codewords

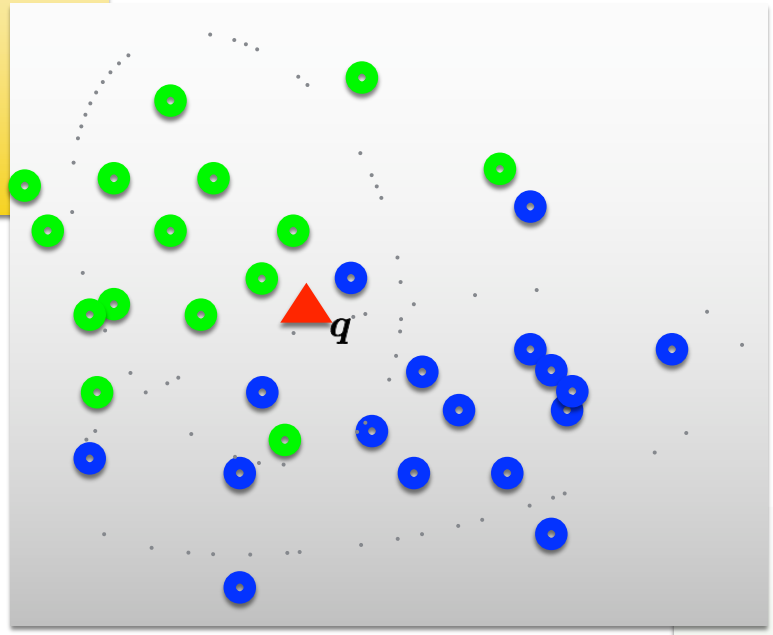
Dictionary Learning:
Learn Visual Words using clustering

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

Classify:
Train and test data using BOWs

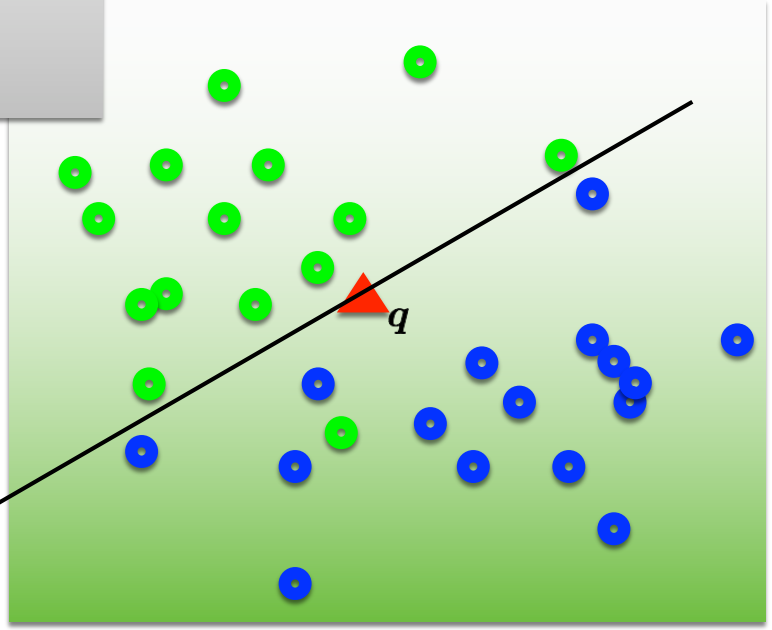


K nearest neighbors



Naïve Bayes

Support Vector Machine



K nearest neighbors

Nearest neighbors

deer bird plane cat car

Training data with labels

?

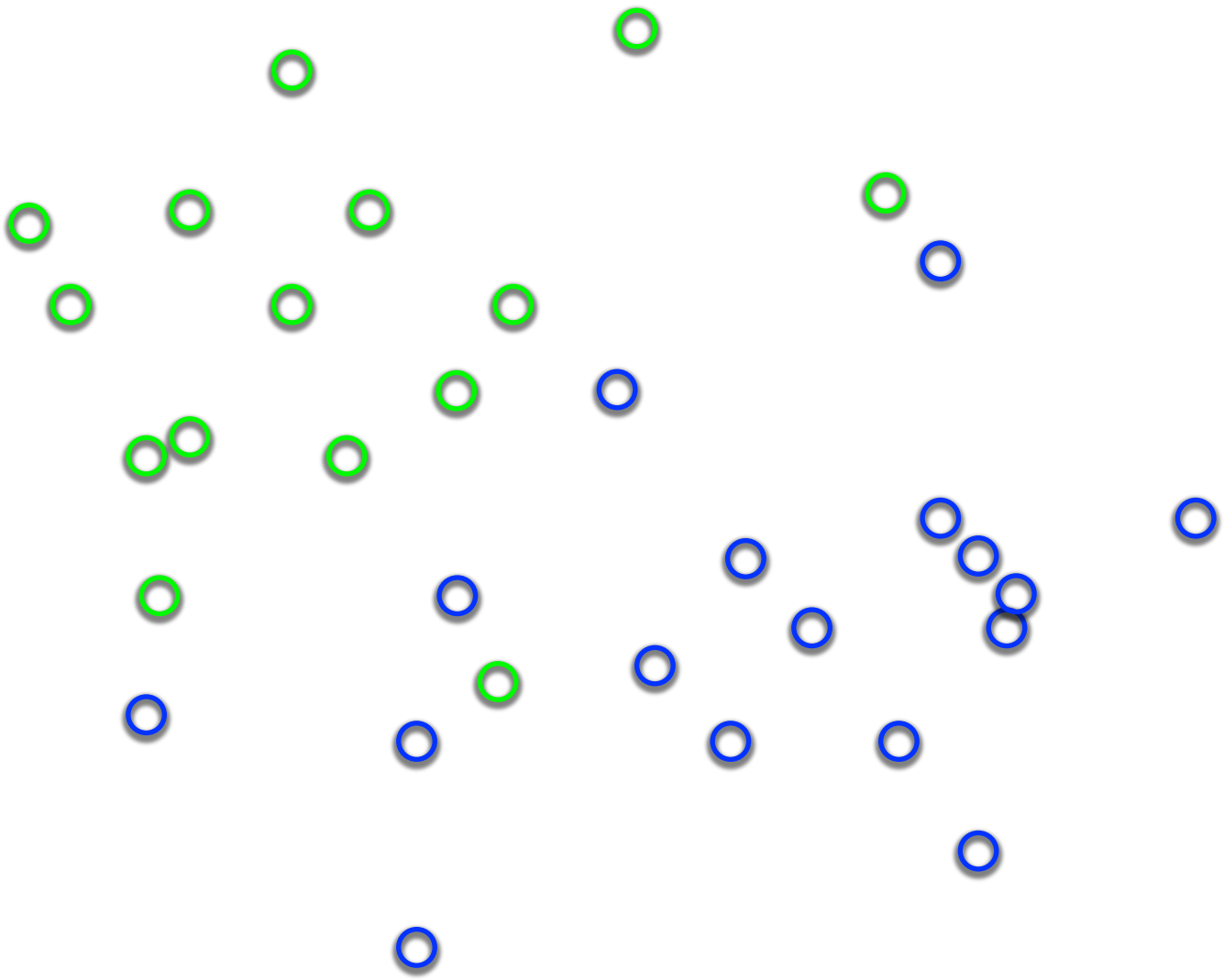
query data

Distance Metric

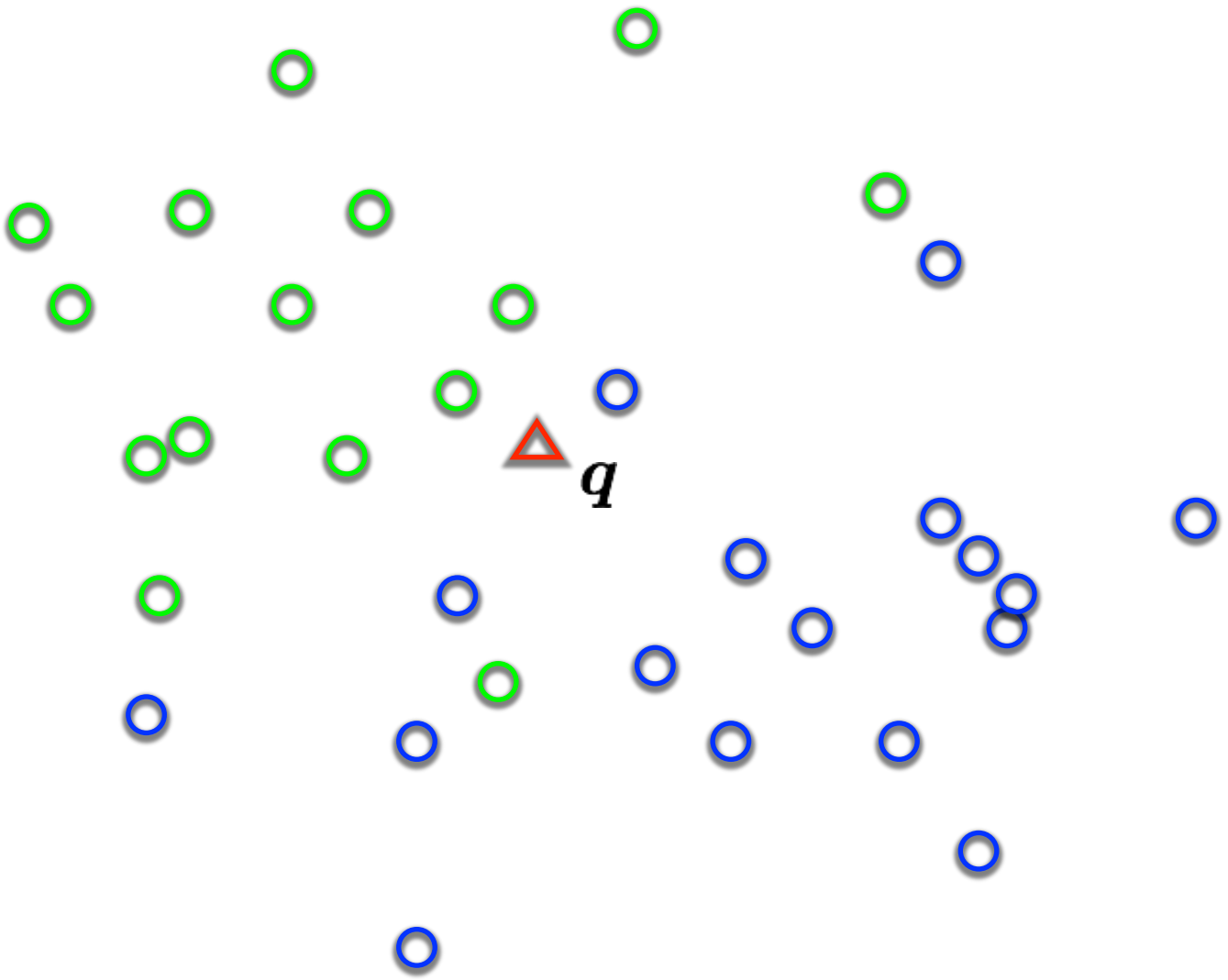
K=1

$$\left| \quad , \quad \right| \rightarrow \mathbb{R}$$

Distribution of data from two classes

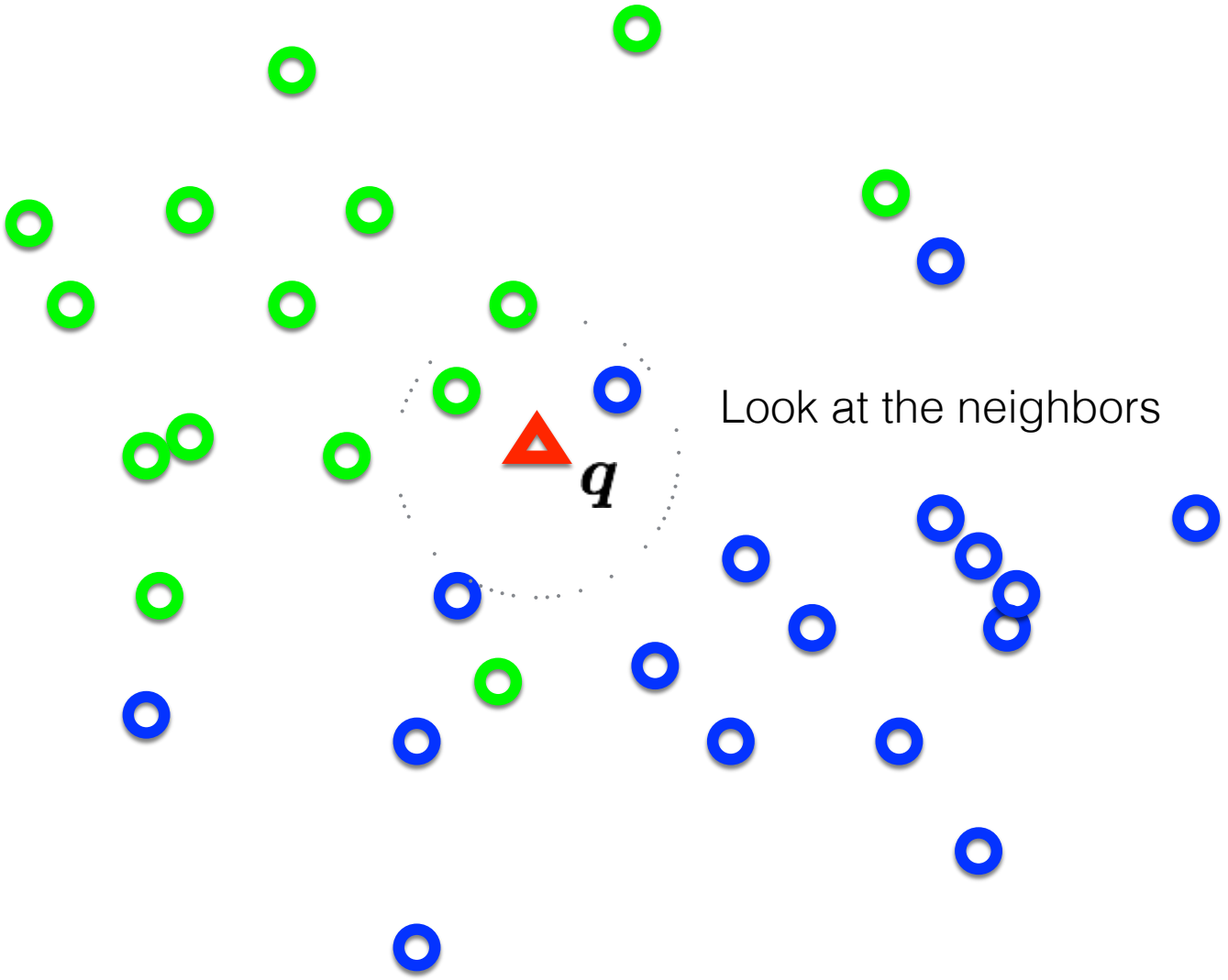


Distribution of data from two classes

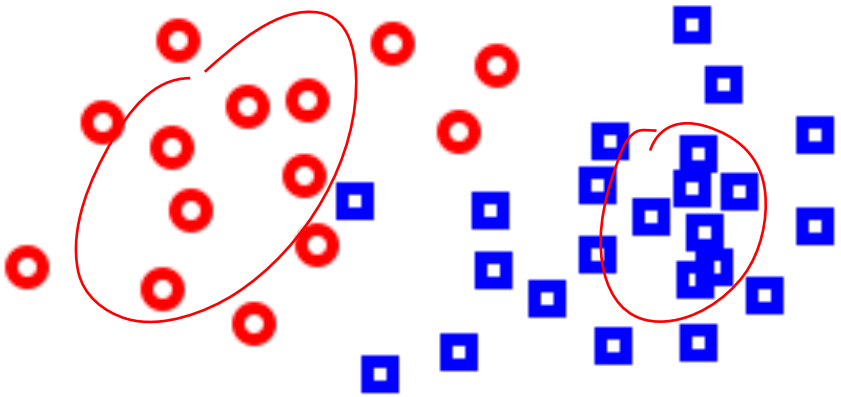


Which class does q belong to?

Distribution of data from two classes



K-Nearest Neighbor (KNN) Classifier

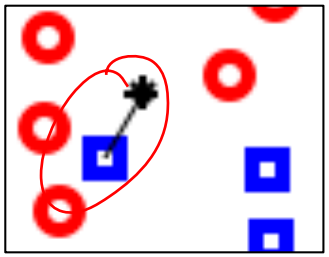


Non-parametric pattern classification approach

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

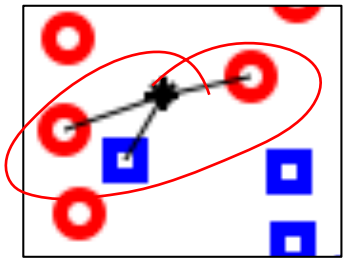
For a given query point q, assign the class of the nearest neighbor

k = 1



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

k = 3



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

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

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

$$D(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2} \sum_n \frac{(x_n - y_n)^2}{(x_n + y_n)}$$ 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

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

- Two most commonly used special cases of p-norm

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

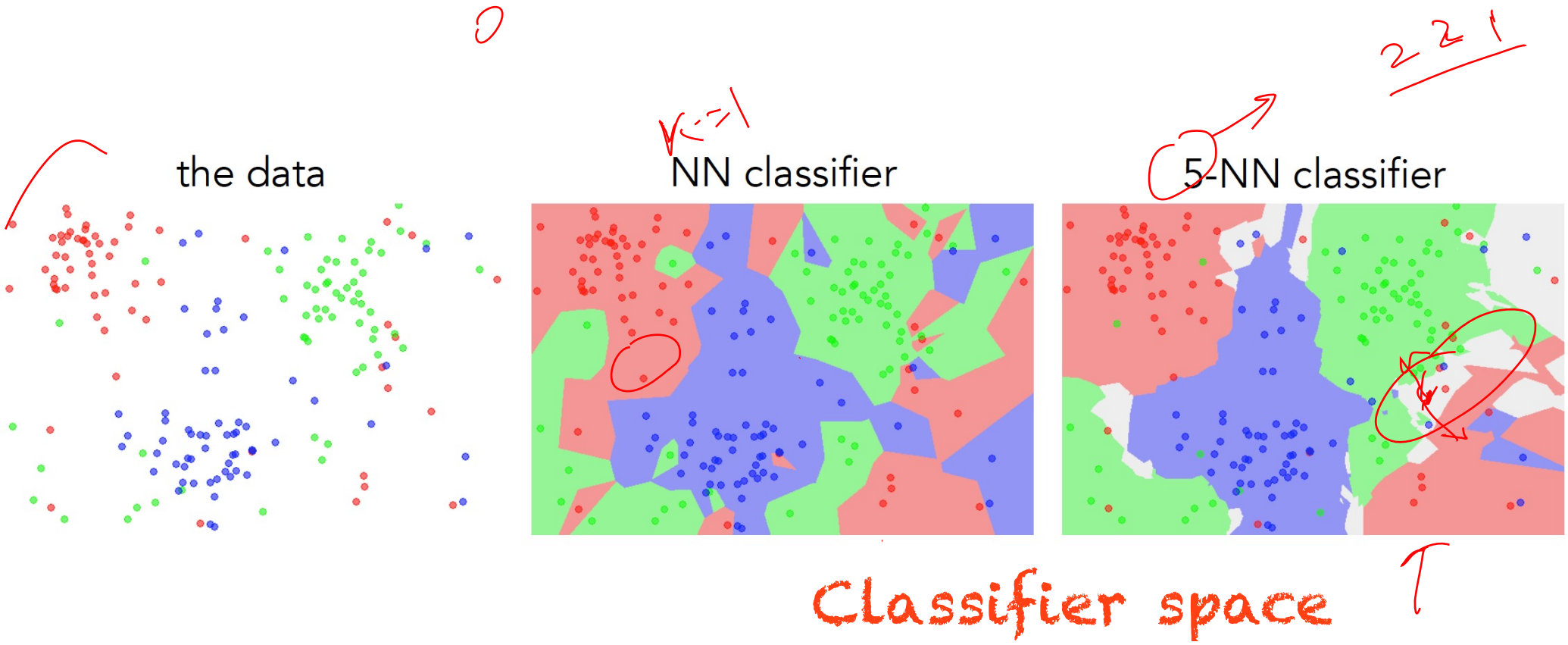
CIFAR-10 and NN results

Example dataset: **CIFAR-10**
10 labels
50,000 training images
10,000 test images.

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



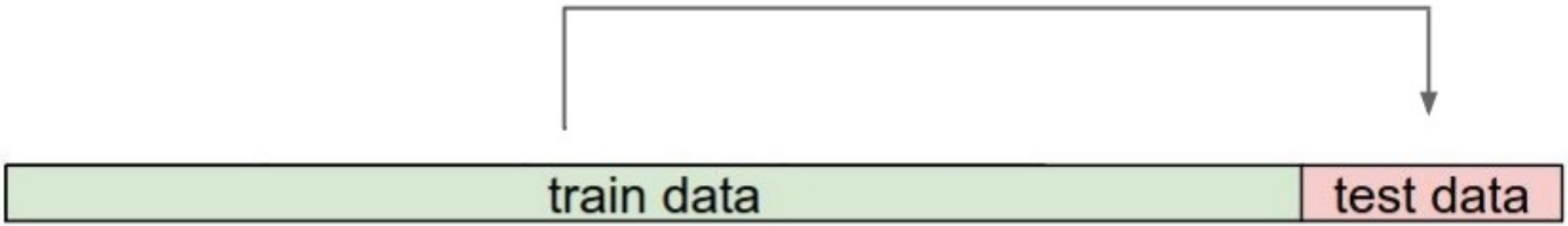
Hyperparameters

- What is the best distance to use?
- What is the best value of k to use?

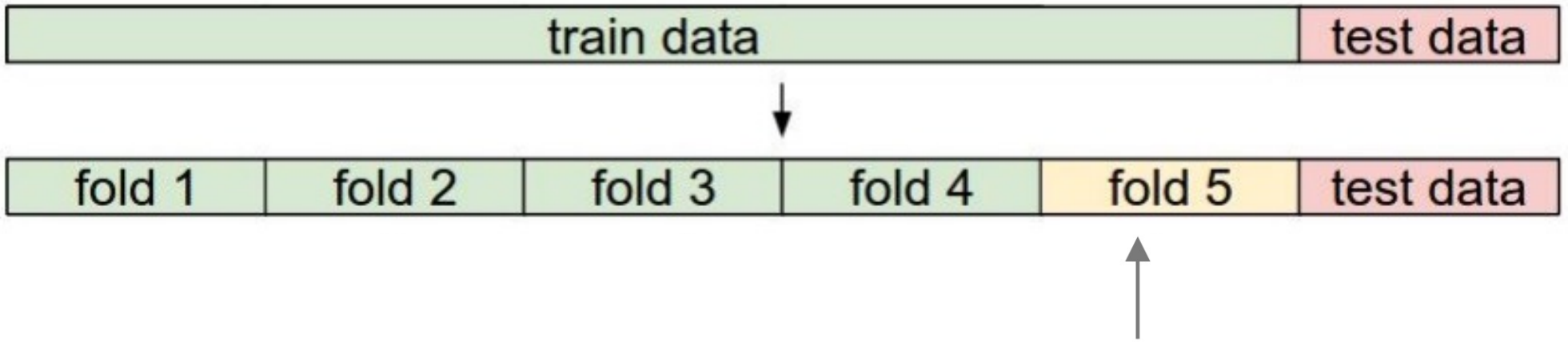
- i.e., how do we set the hyperparameters?

- Very problem-dependent
- Must try them all and see what works best

Try out what hyperparameters work best on test set.



Validation

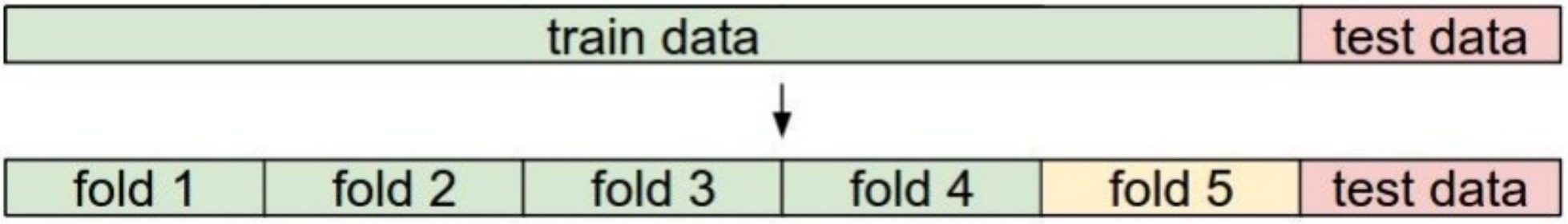


Validation data

use to tune hyperparameters

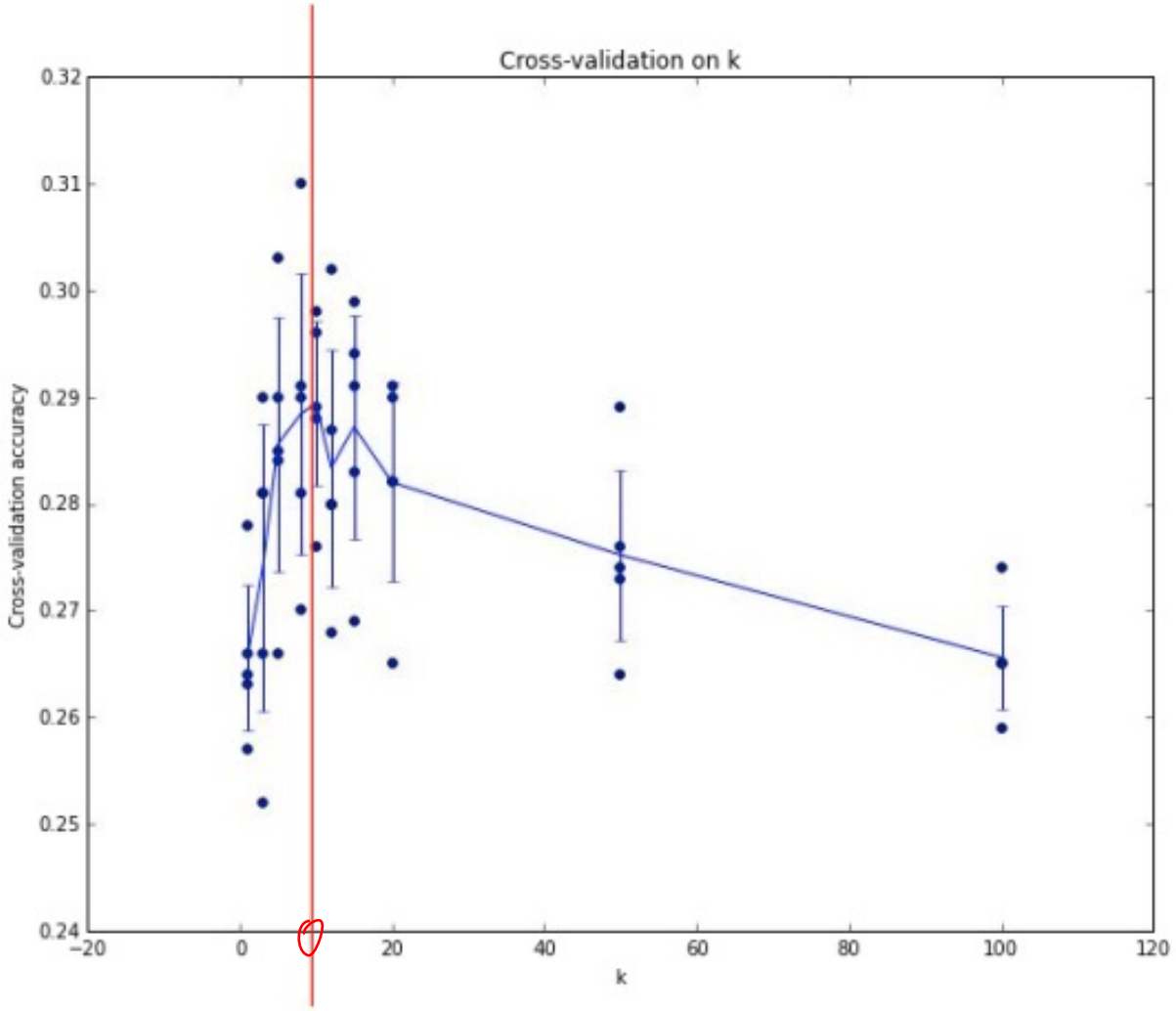
evaluate on test set ONCE at the end

Cross-validation



Cross-validation

cycle through the choice of which fold is the validation fold, average results.



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 ~= 7 works best
for this data)

How to pick hyperparameters?

- Methodology
  - Train, validate, test

- Train for original model
- Validate to find hyperparameters
- Test to understand generalizability

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

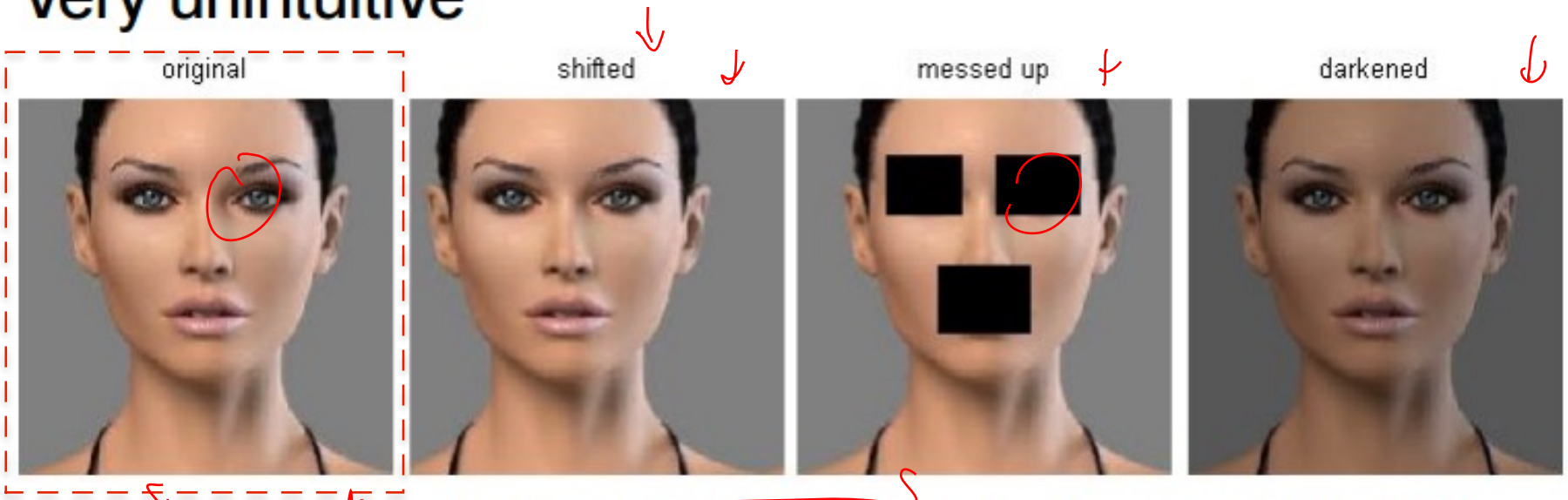
- Training: O(1)
- Testing: O(MN)

why?

- 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)

Support Vector Machine

Image Classification

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

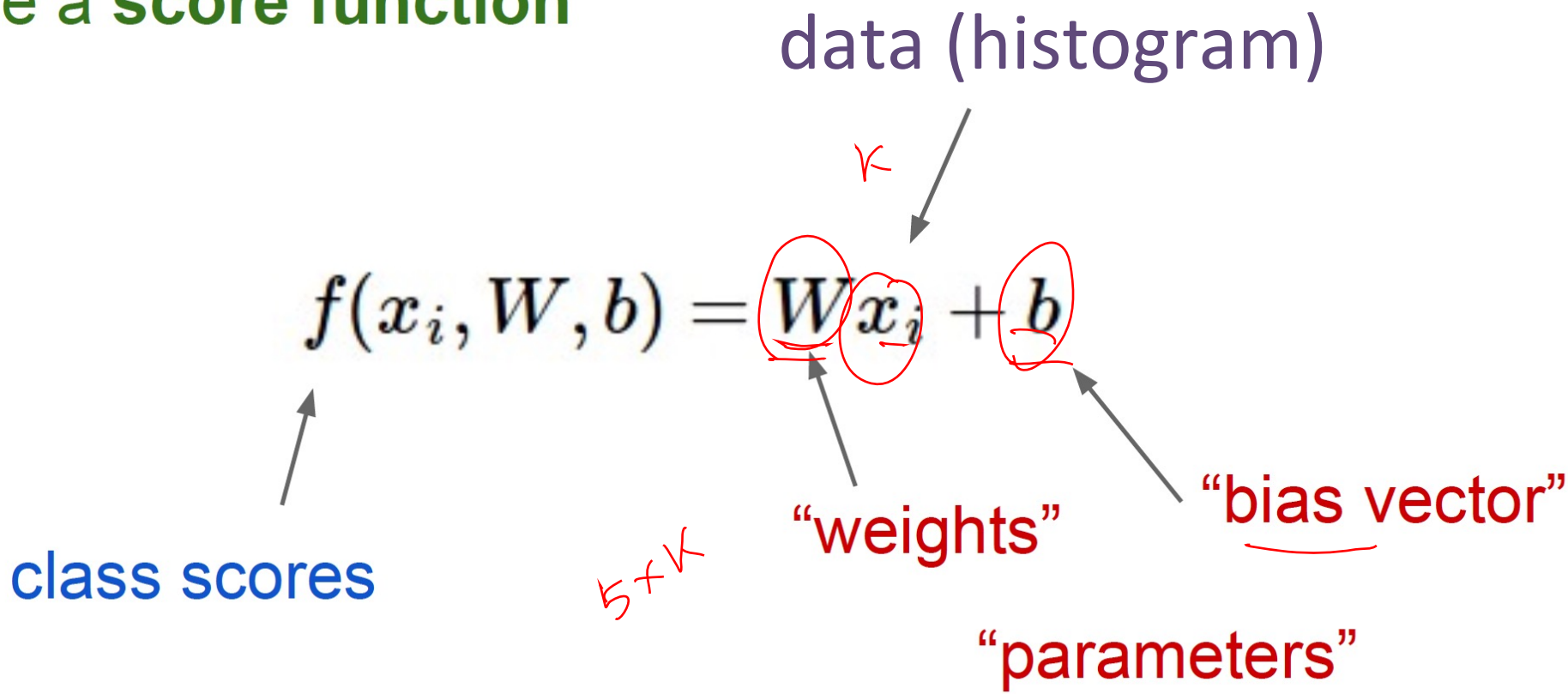
cat

Score function

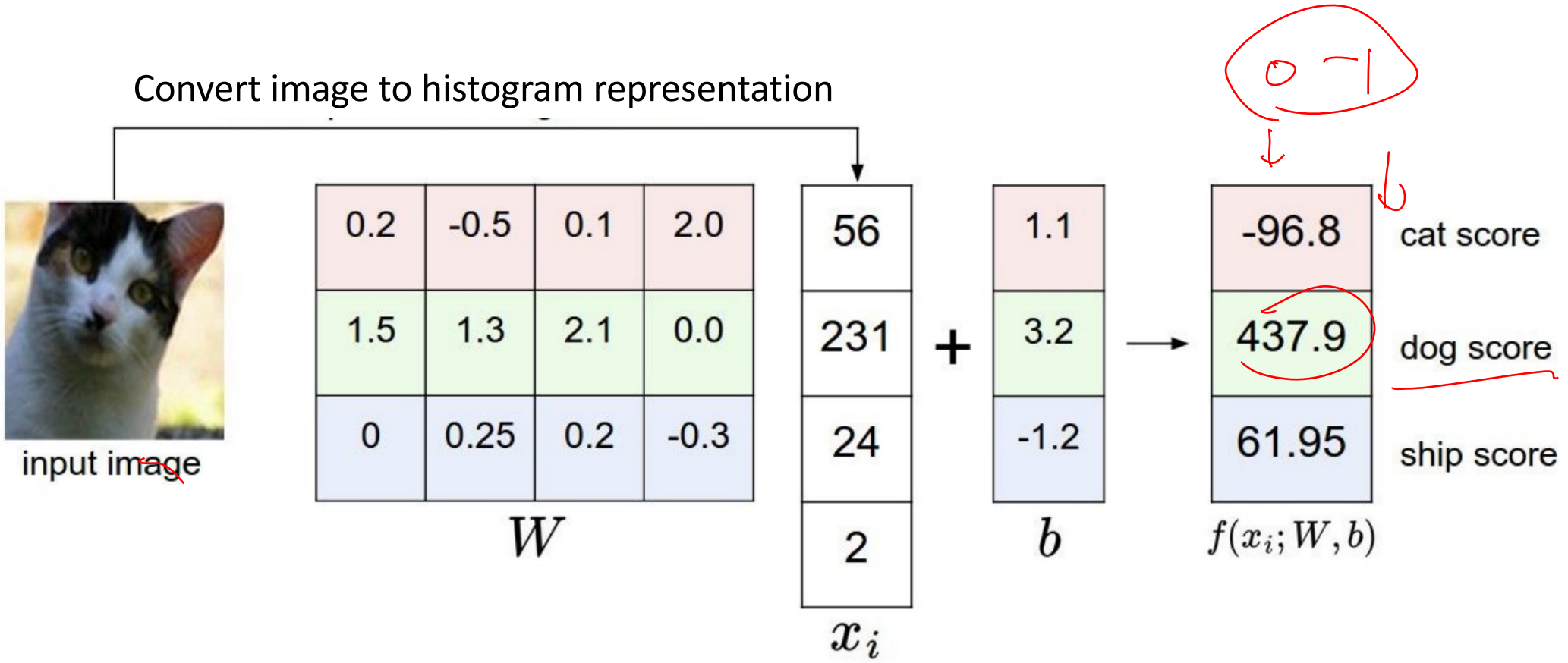
class scores

Linear Classifier

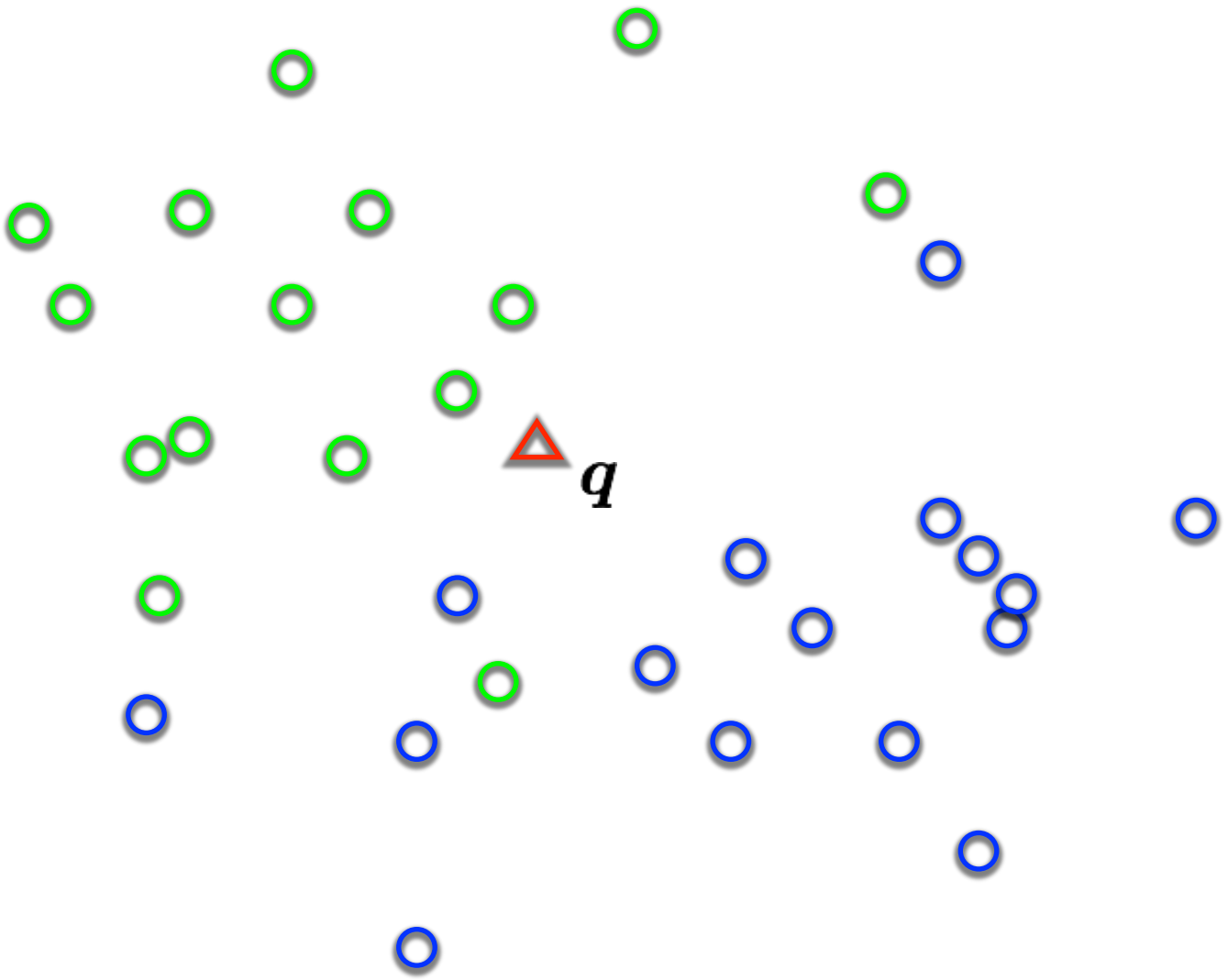
define a **score function**



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

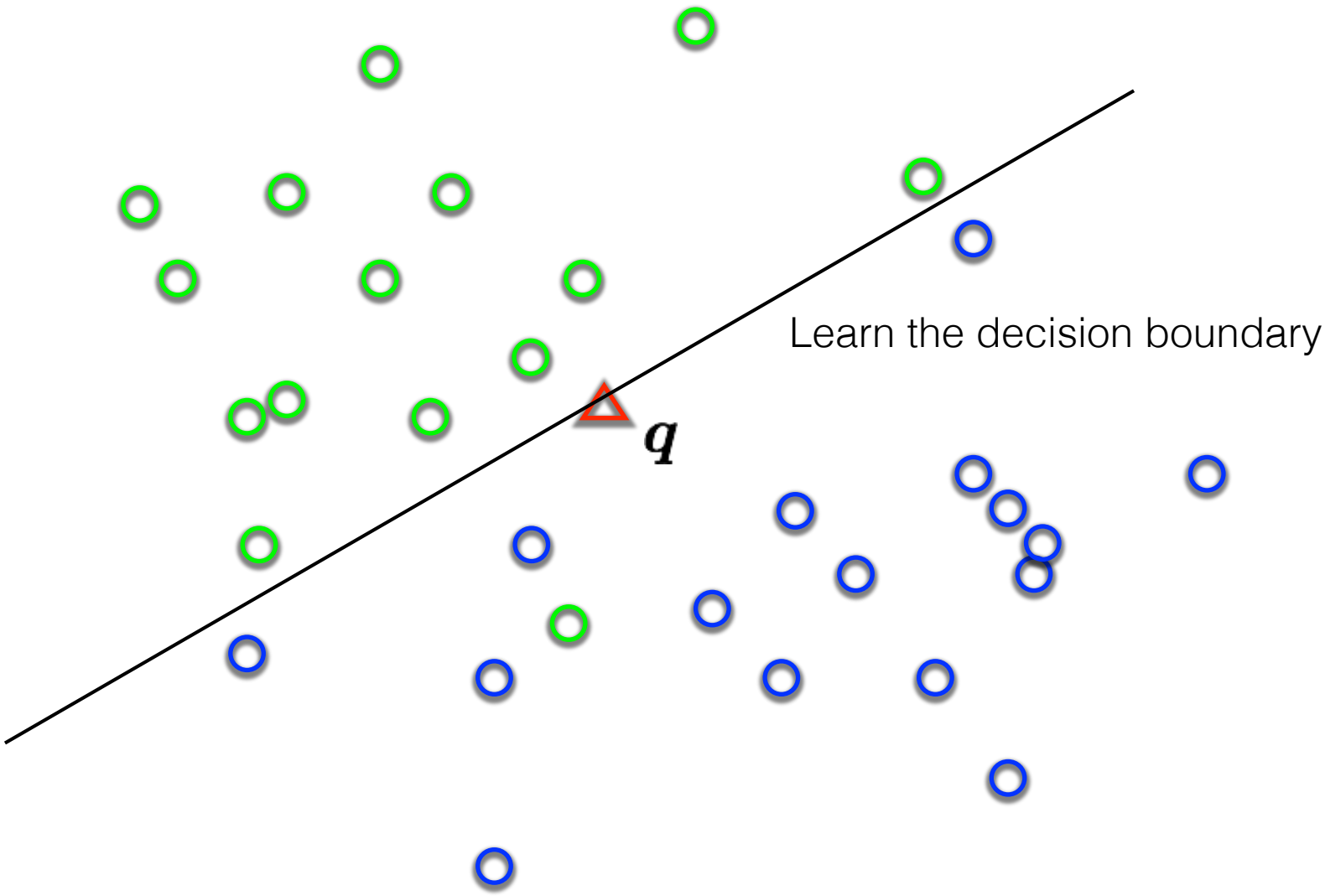


Distribution of data from two classes



Which class does q belong to ?

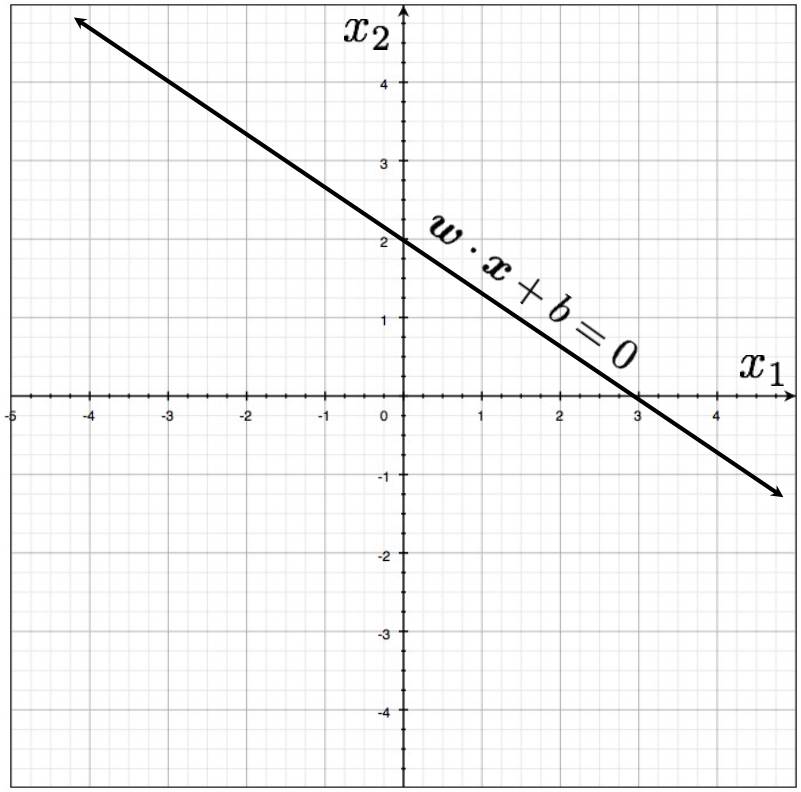
Distribution of data from two classes



First we need to understand hyperplanes…

Hyperplanes (lines) in 2D

$$w_1 x_1 + w_2 x_2 + b = 0$$



a line can be written as
dot product plus a bias

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

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

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

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

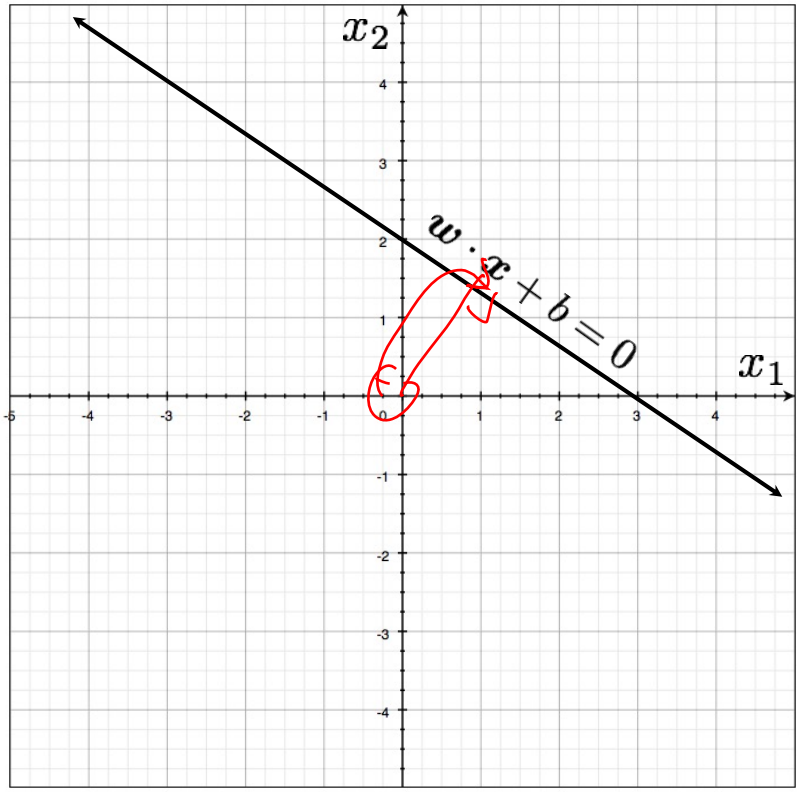
$$\boldsymbol{w} \in \mathcal{R}^3$$

(w_1, w_2, b)

Hyperplanes (lines) in 2D

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

$$w_1 x_1 + w_2 x_2 + b = 0$$

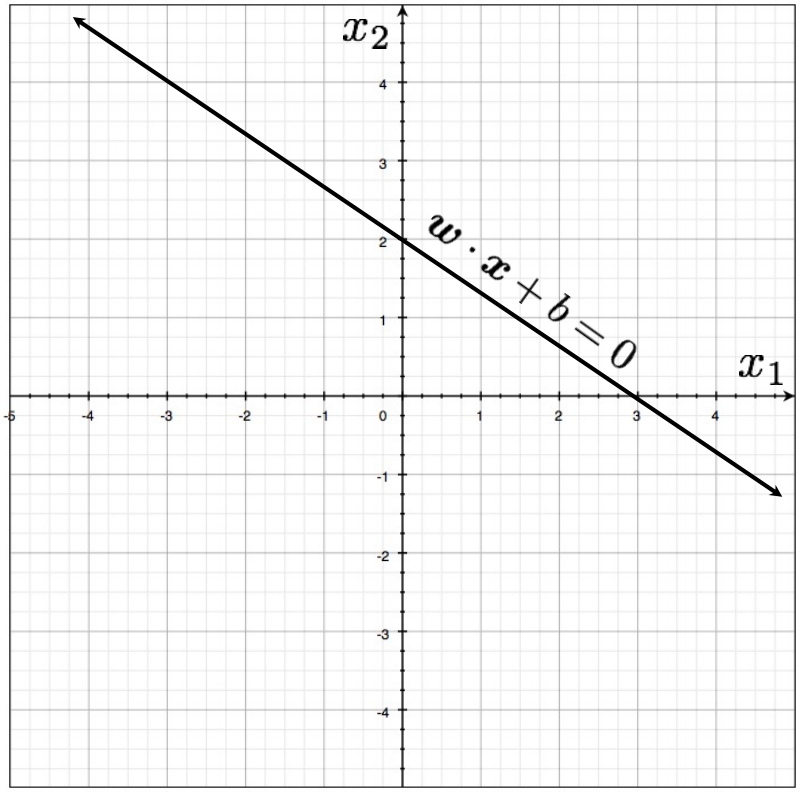


$\frac{b}{\|w\|}$

Hyperplanes (lines) in 2D

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

$$w_1 x_1 + w_2 x_2 + b = 0$$



Important property:

Free to choose any normalization of w

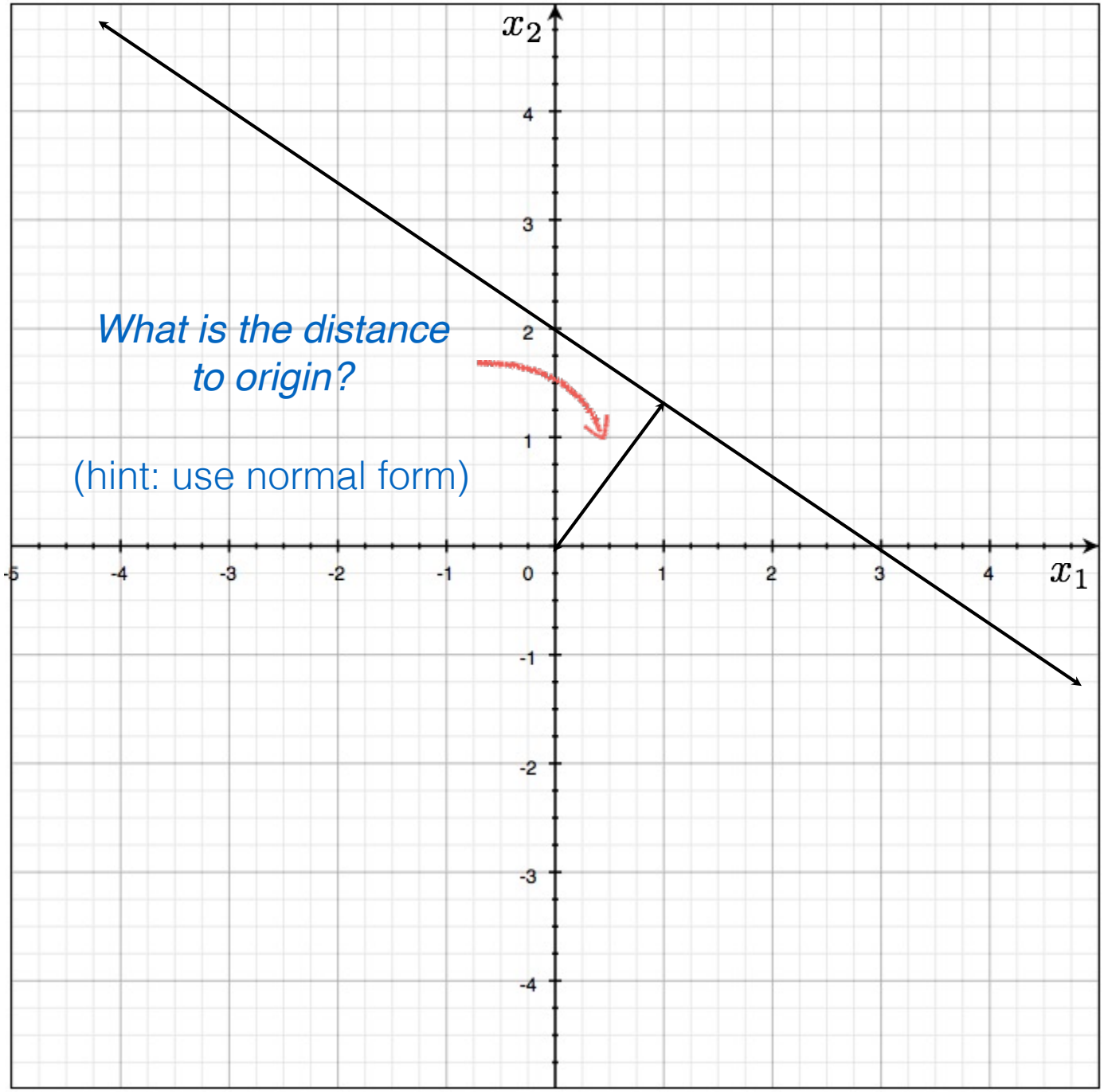
The line

$$w_1 x_1 + w_2 x_2 + b = 0$$

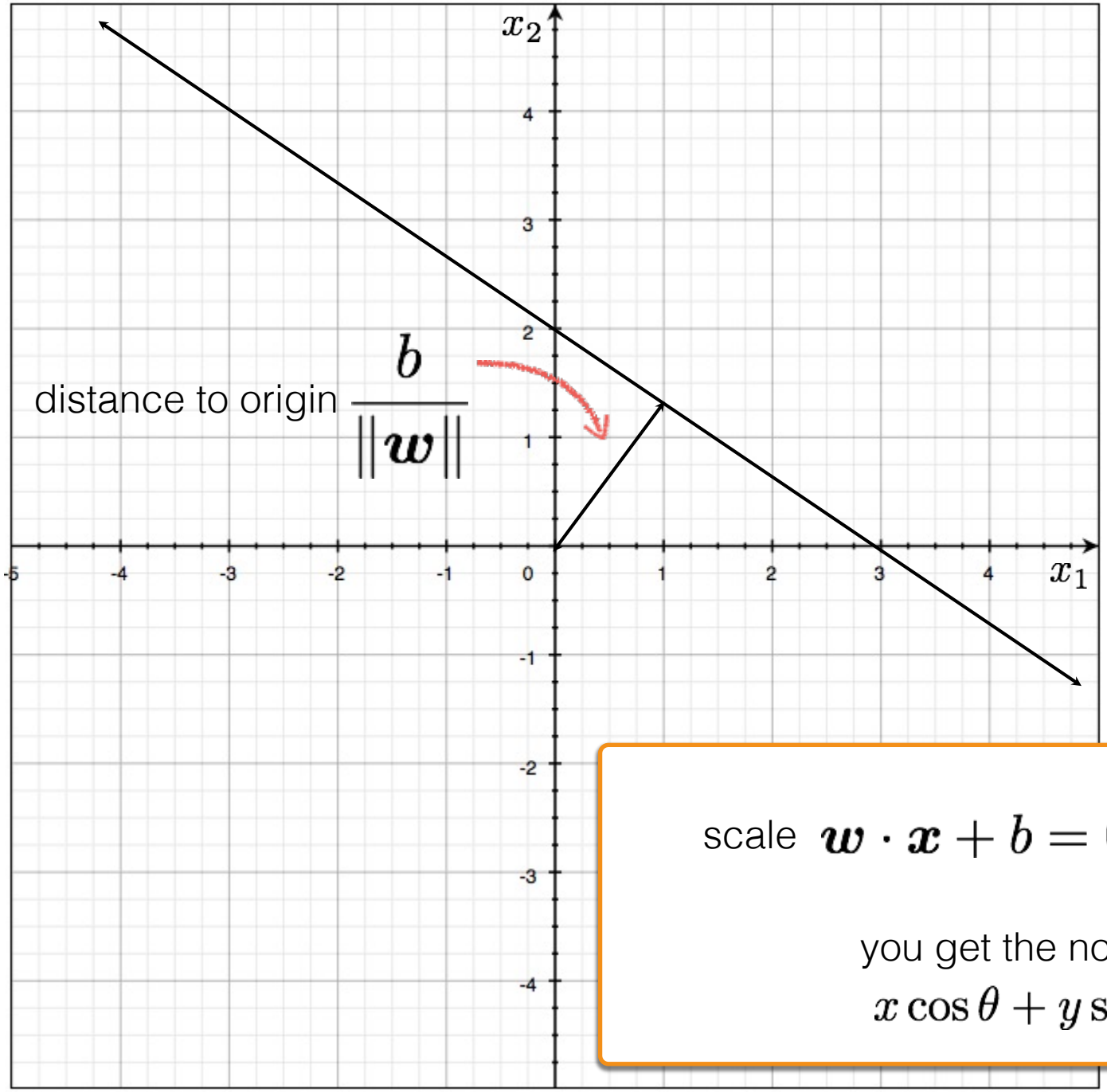
and the line

$$\lambda(w_1 x_1 + w_2 x_2 + b) = 0$$

define the same line



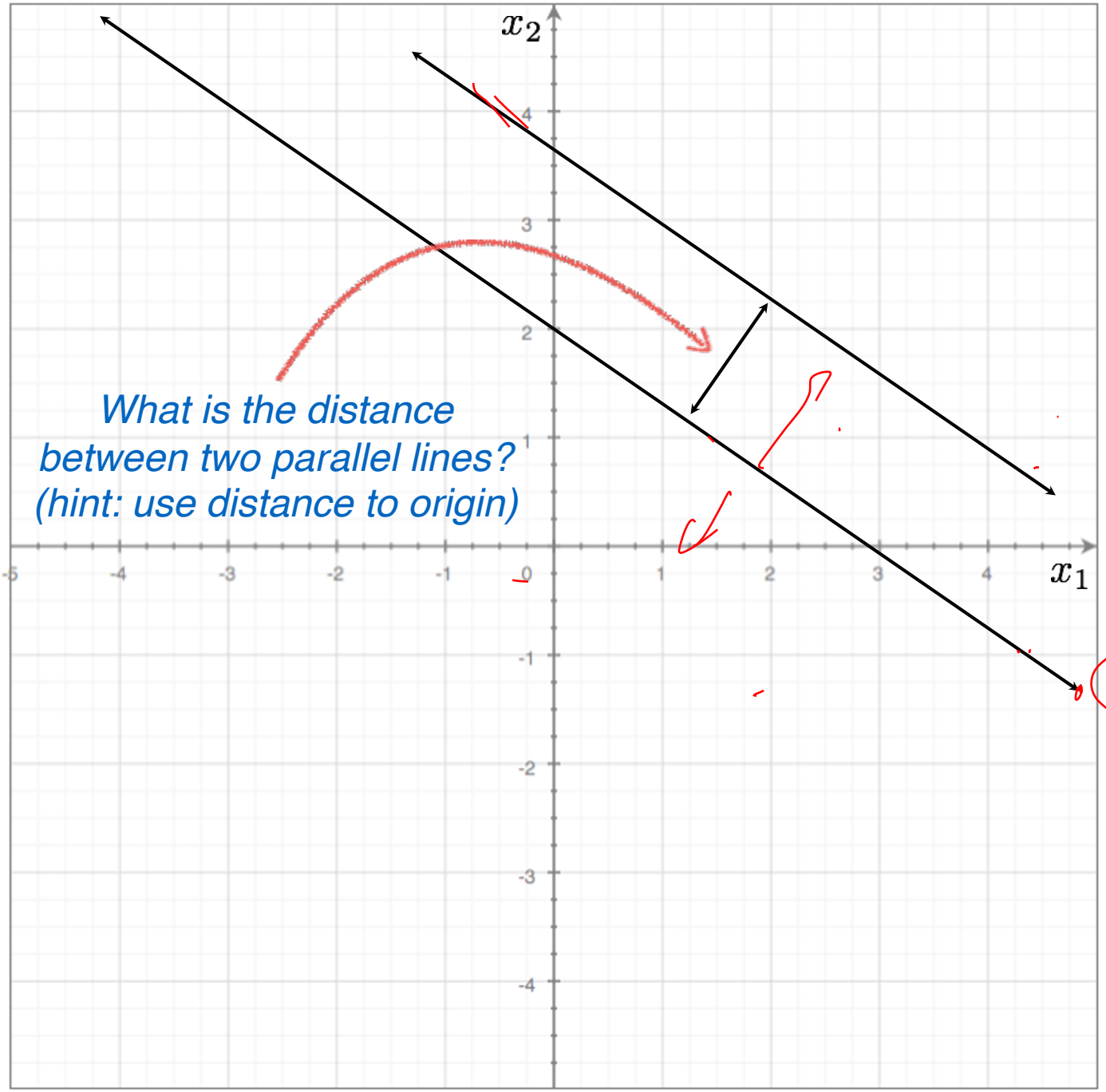
$$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$$

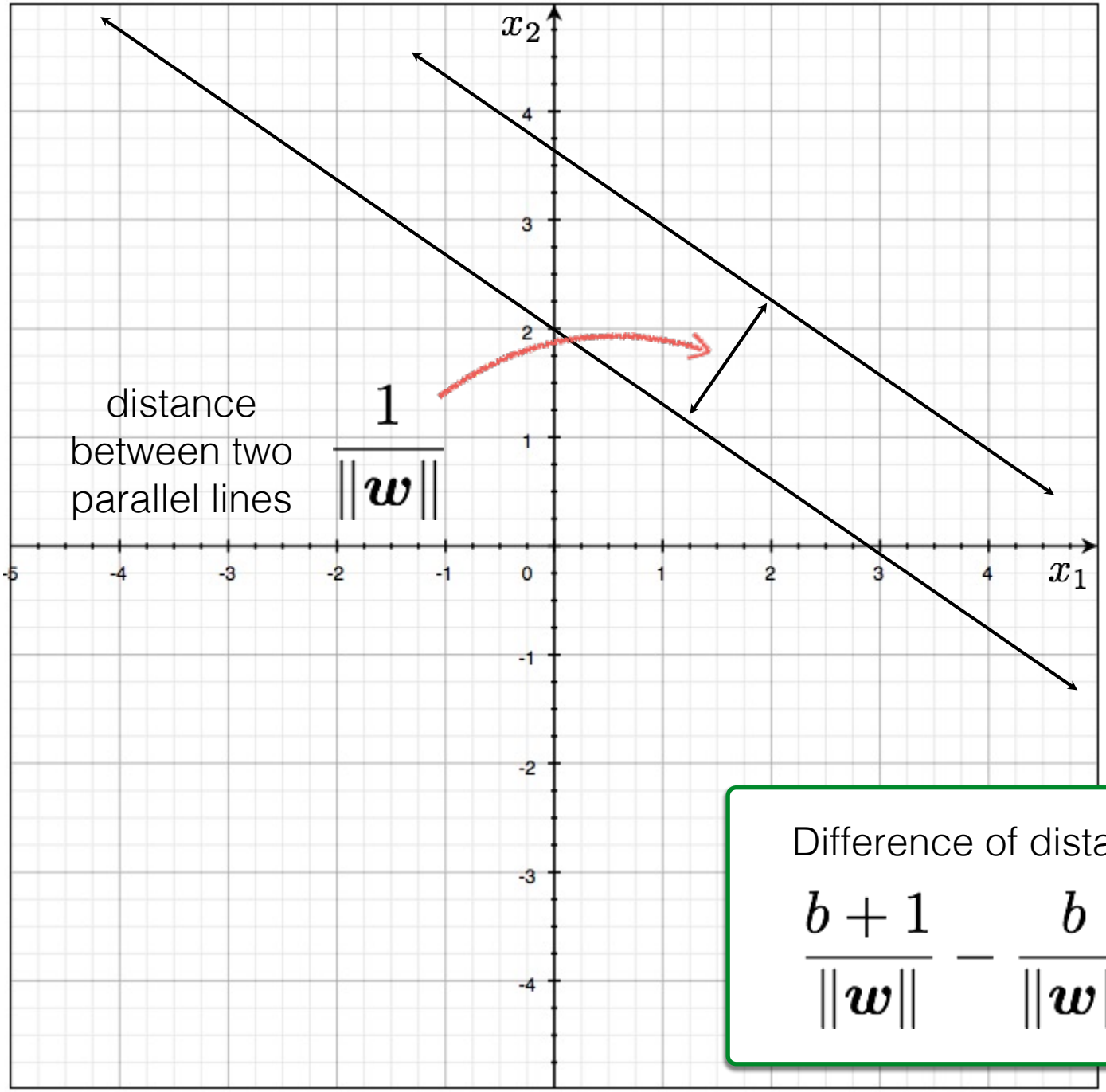


scale $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$ by $\frac{1}{\|\boldsymbol{w}\|}$

you get the normal form

$$x \cos\theta + y \sin\theta = \rho$$


x_2
x_1
What is the distance between two parallel lines? (hint: use distance to origin)
$\boldsymbol{w} \cdot \boldsymbol{x} + b = -1$
$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$



distance between two parallel lines $\frac{1}{\|\boldsymbol{w}\|}$

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

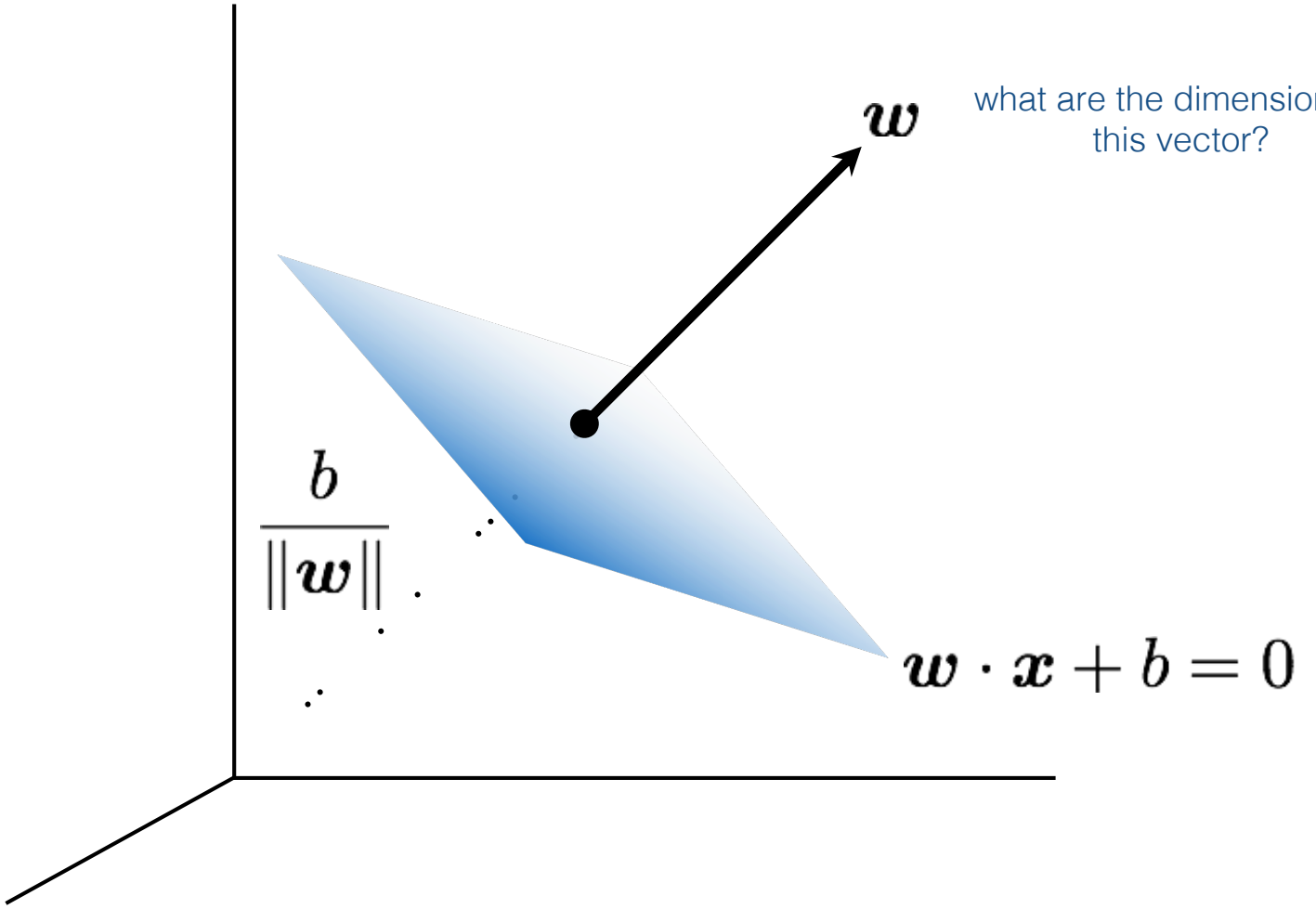
$$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$$

Difference of distance to origin

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

Now we can go to 3D …

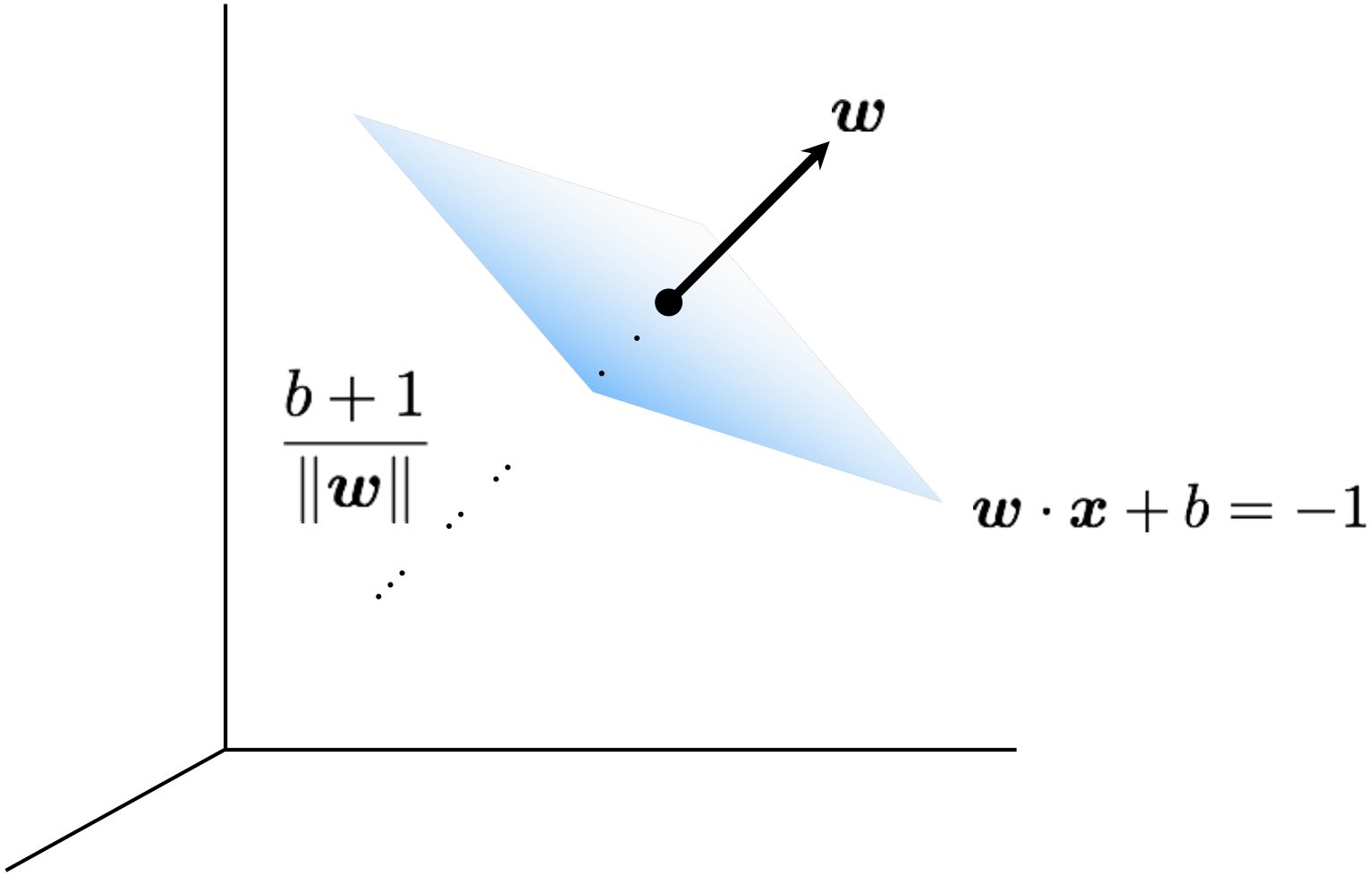
Hyperplanes (planes) in 3D



what are the dimensions of this vector?

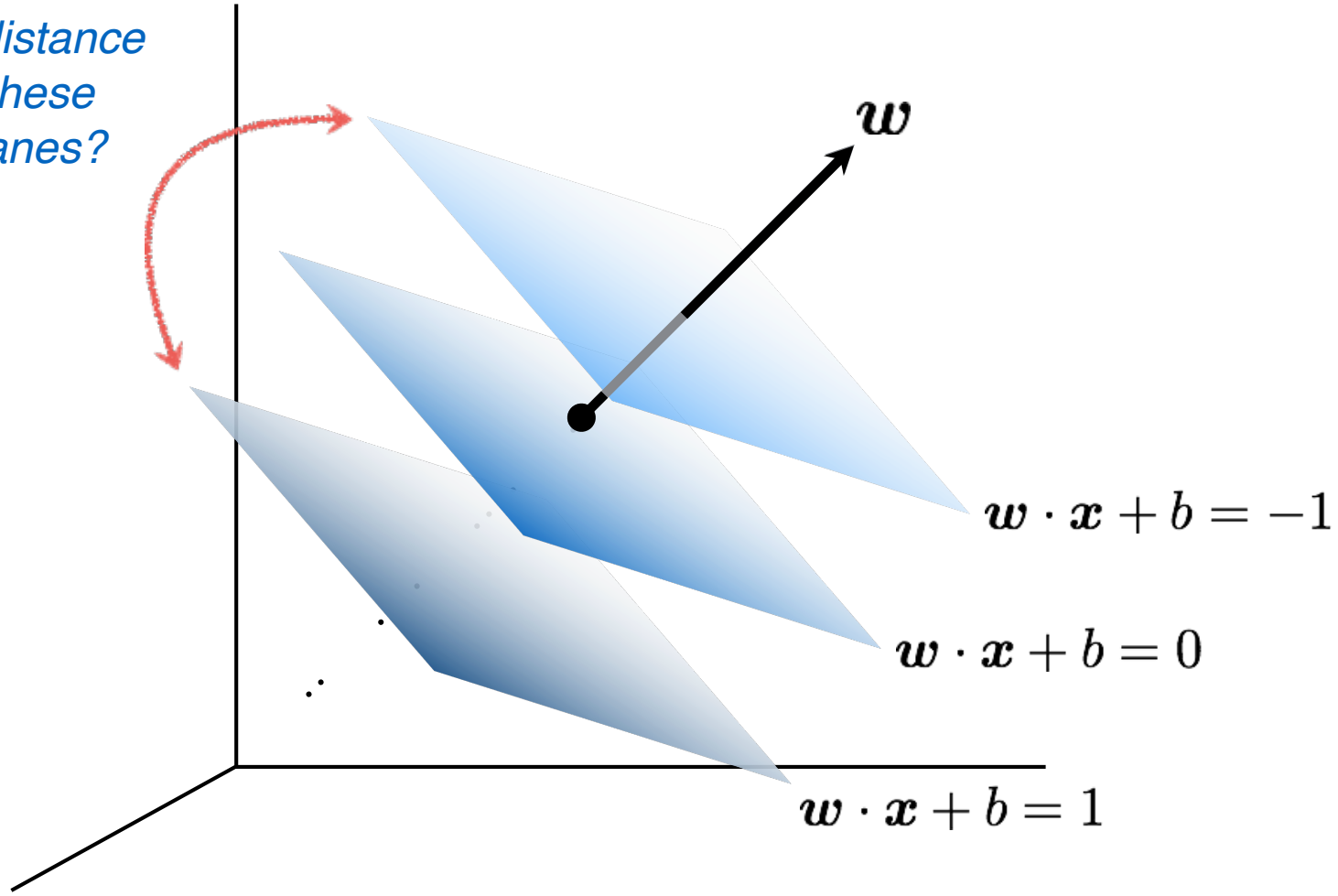
*What happens if you change **b**?*

Hyperplanes (planes) in 3D

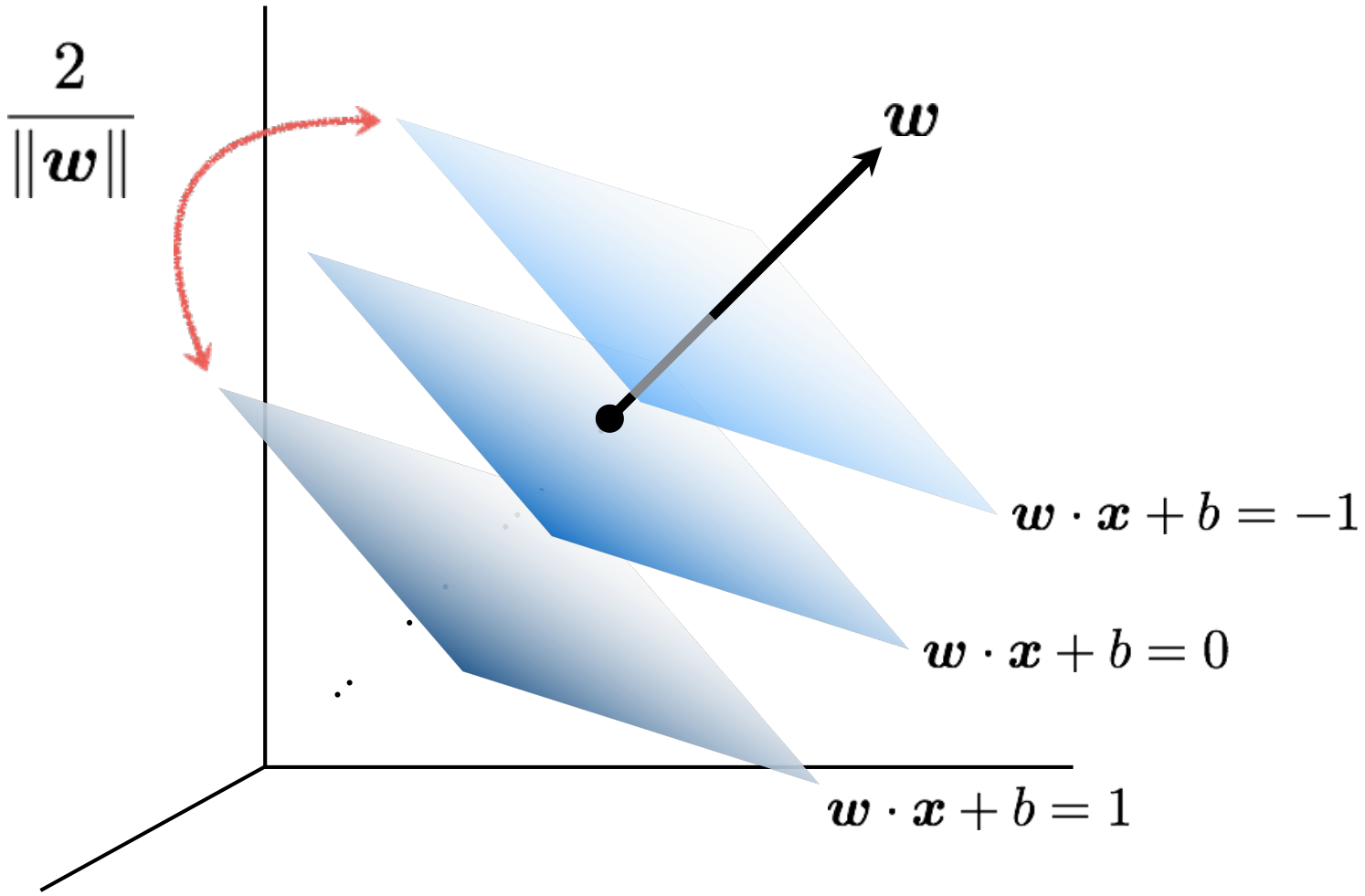


Hyperplanes (planes) in 3D

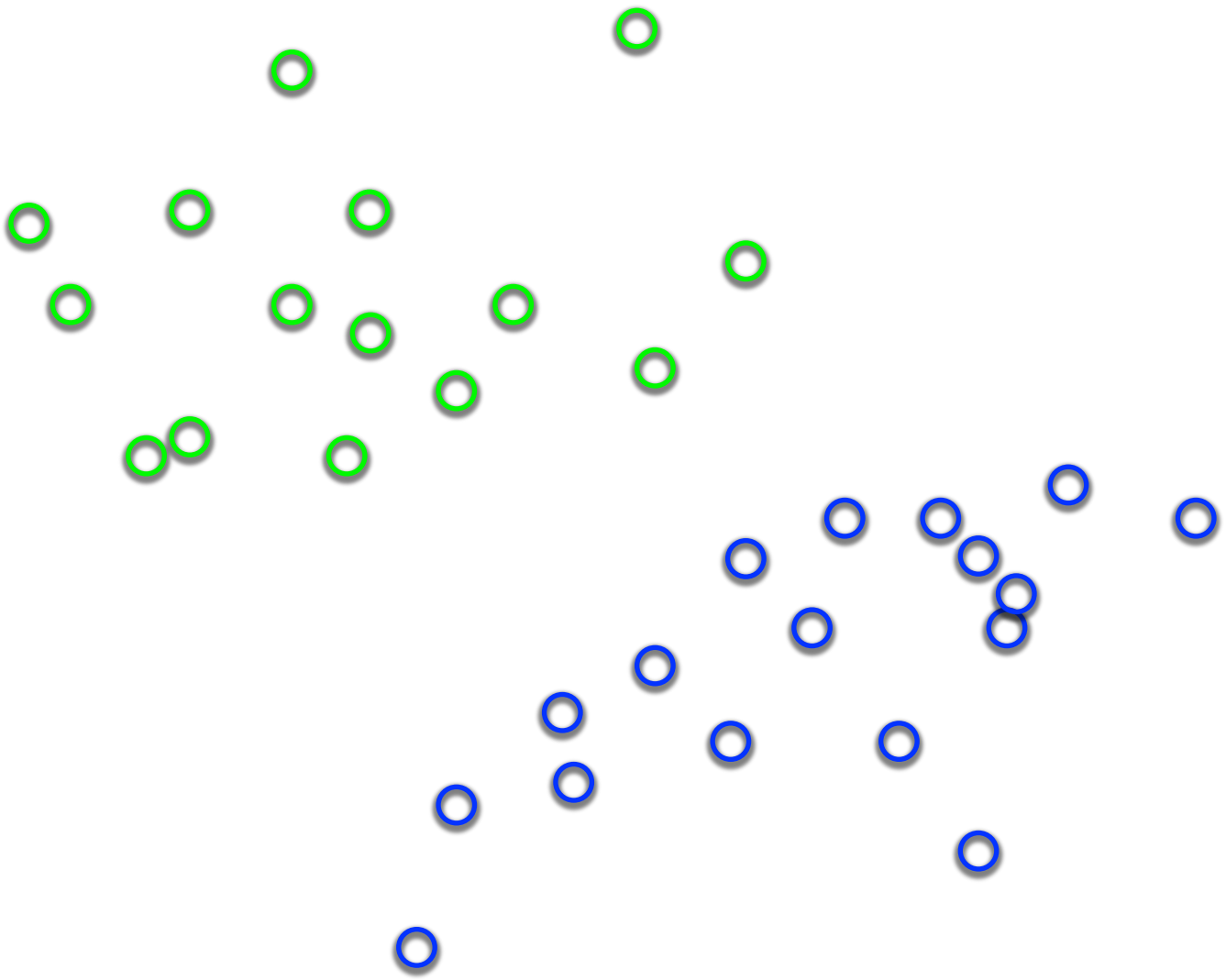
What's the distance between these parallel planes?



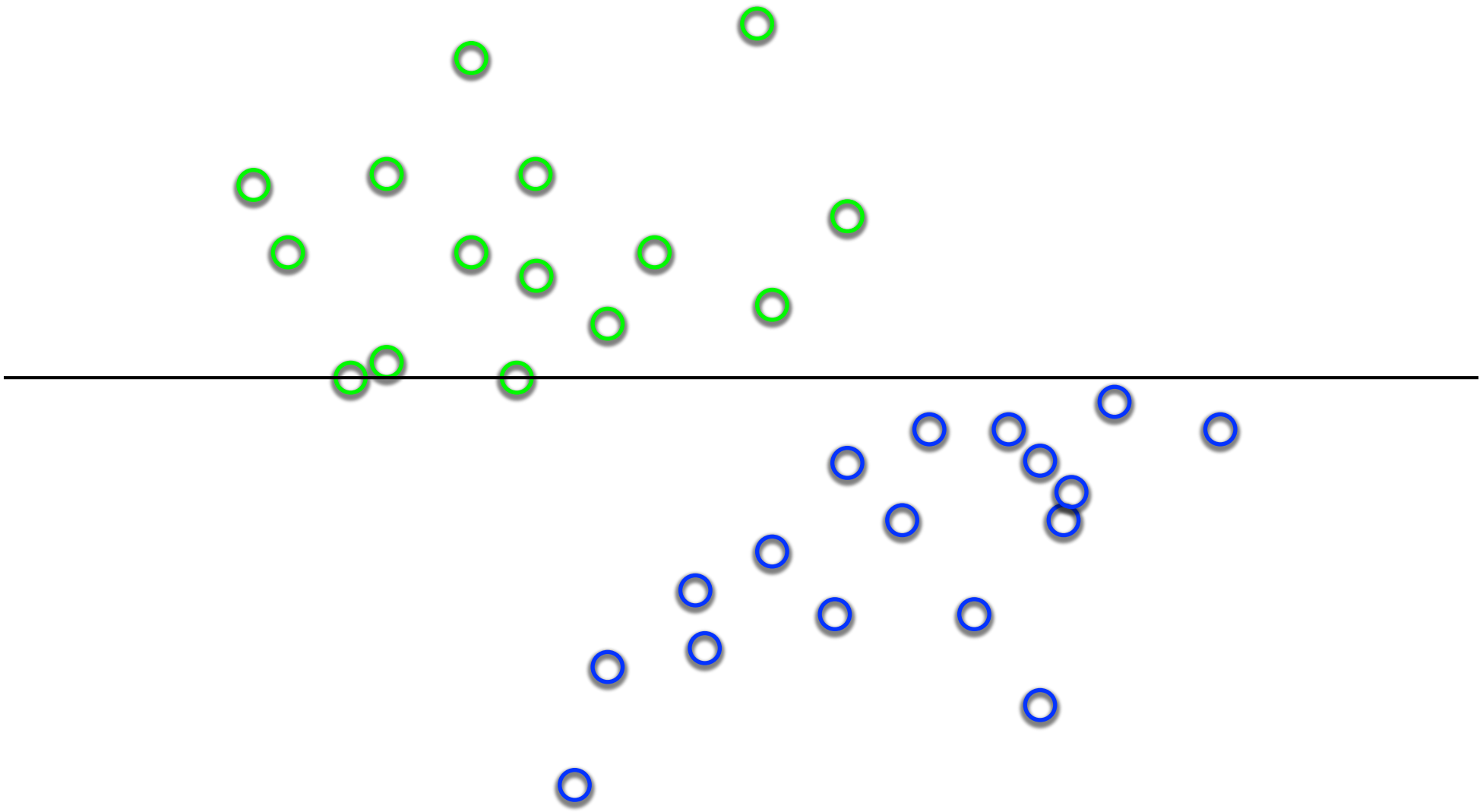
Hyperplanes (planes) in 3D



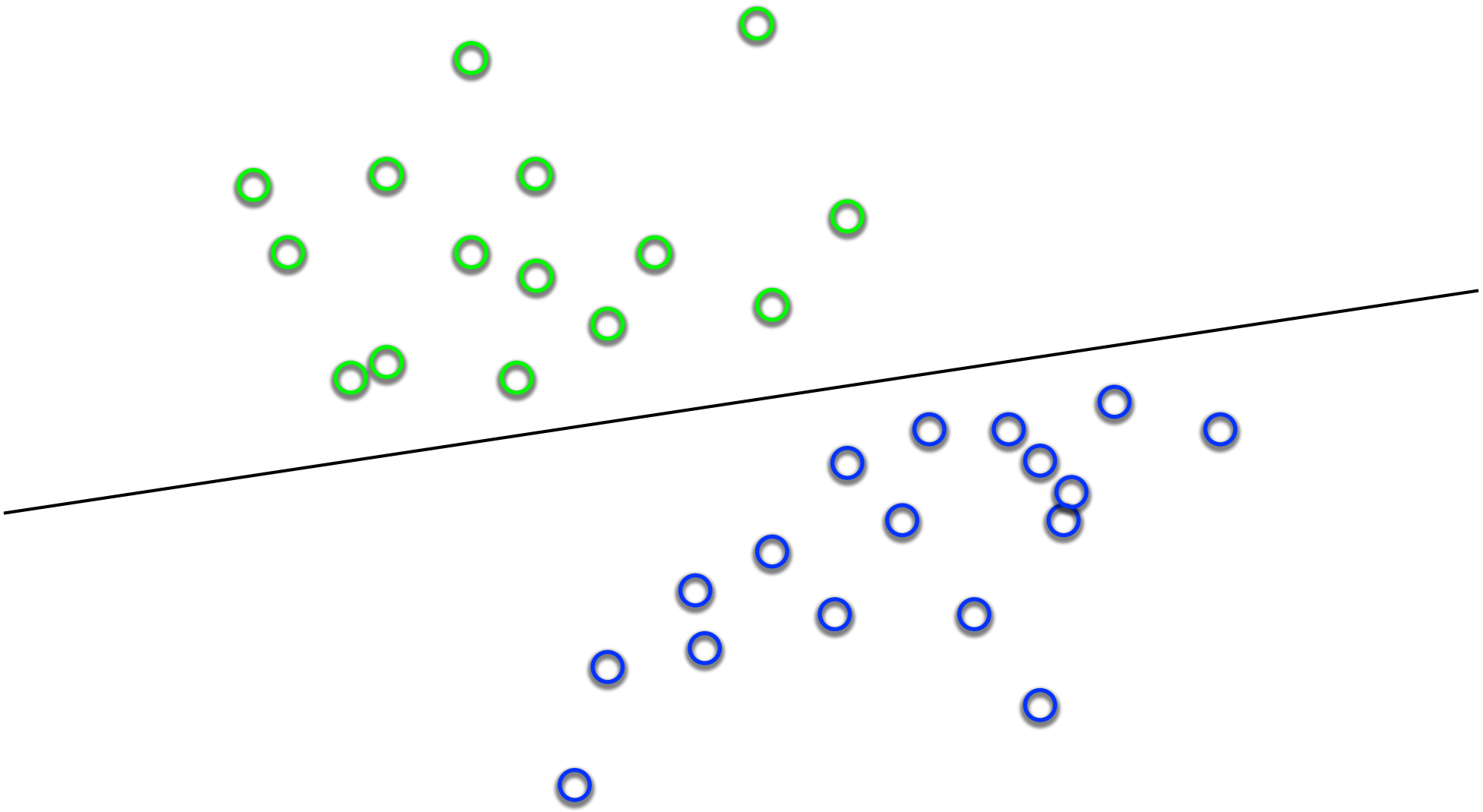
What's the best **w**?



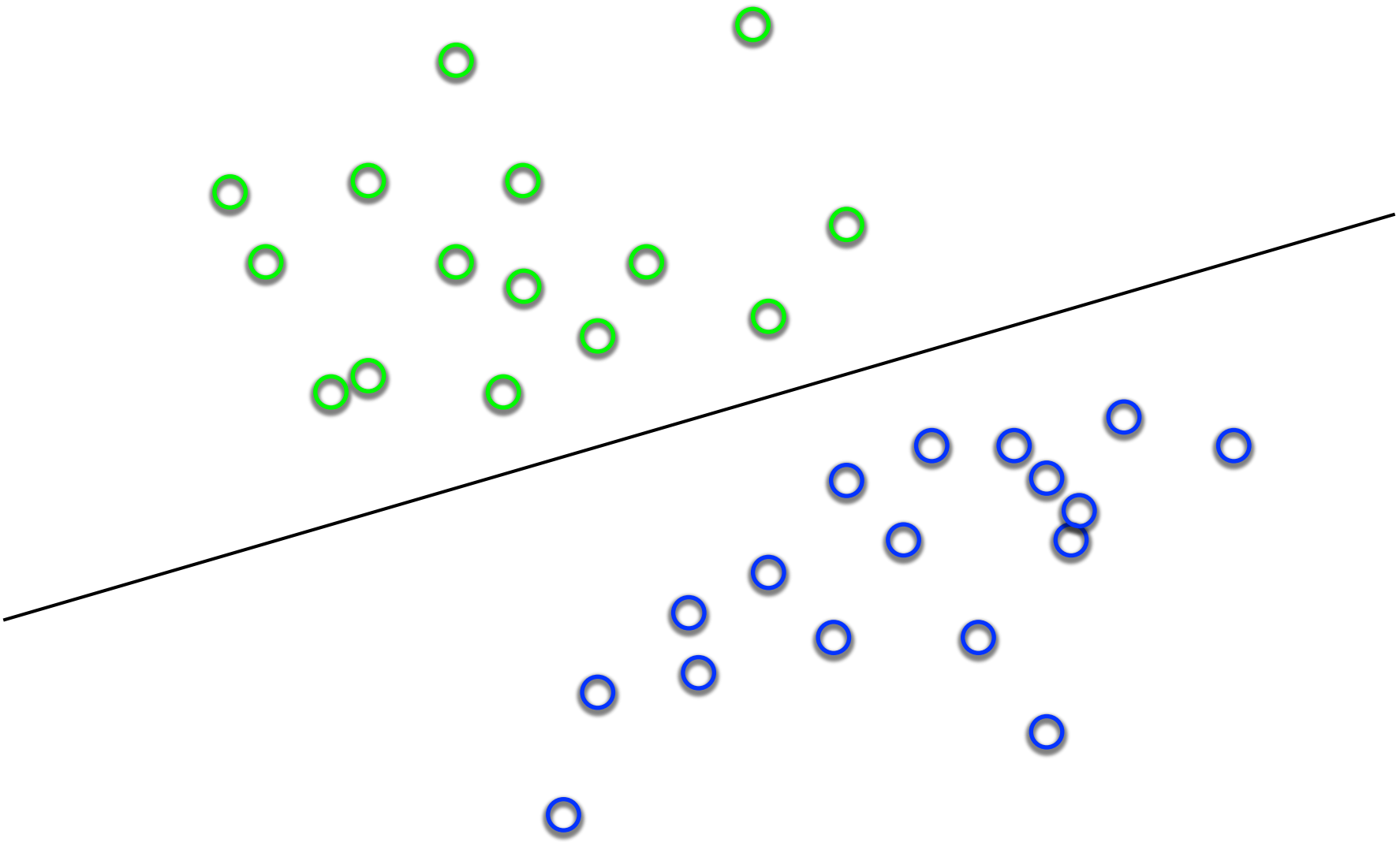
What's the best **w**?



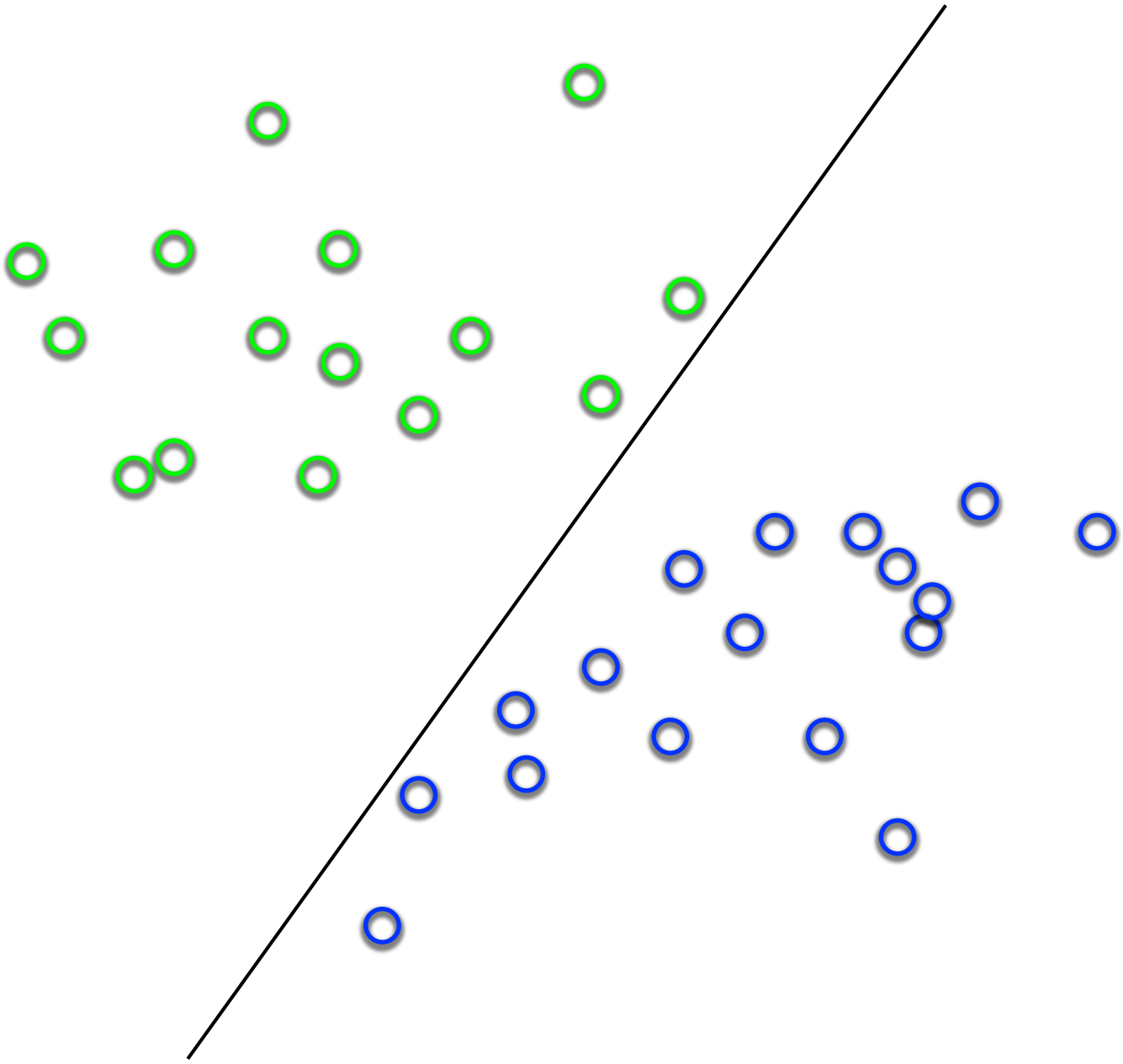
What's the best **w**?



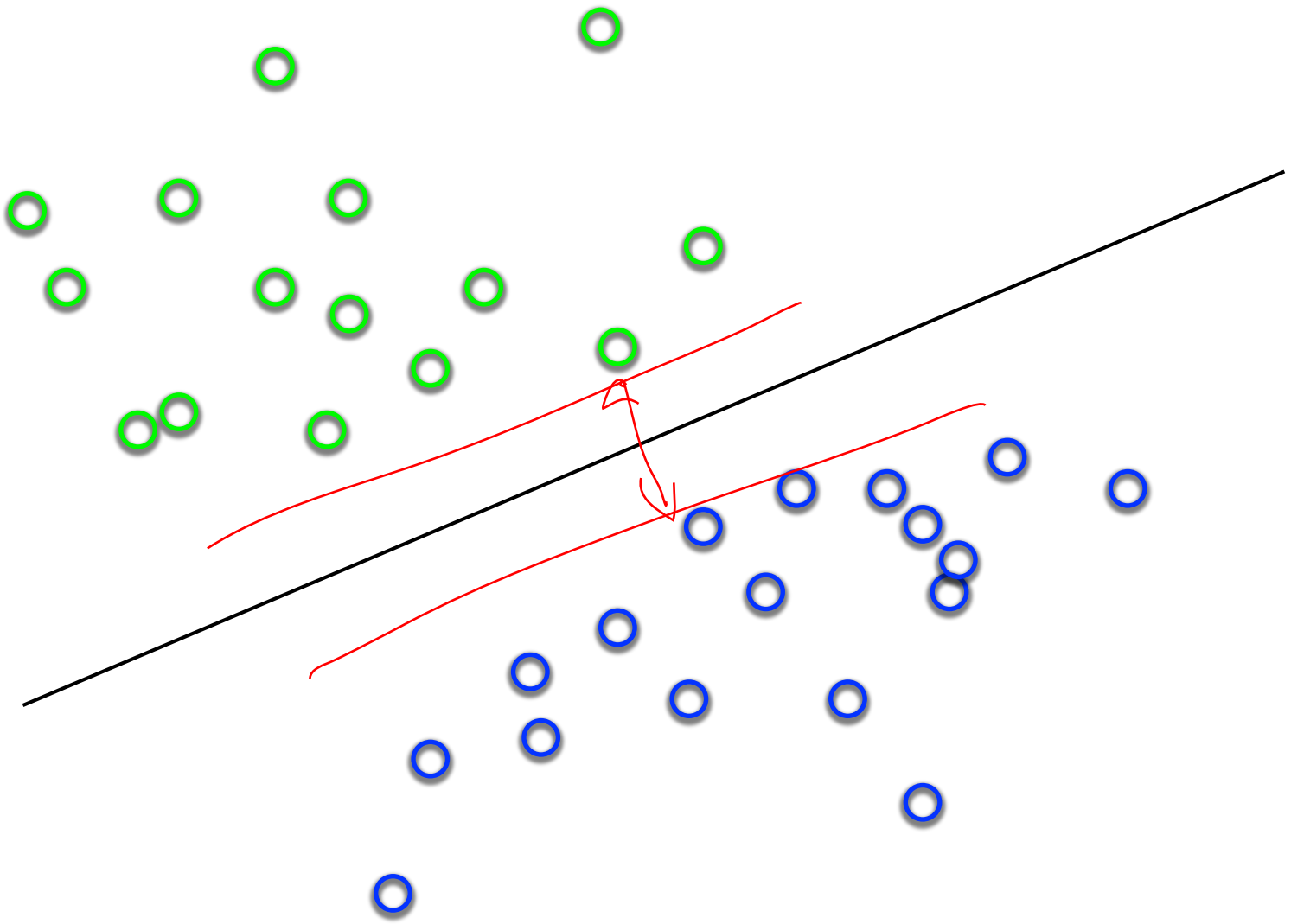
What's the best **w**?



What's the best **w**?

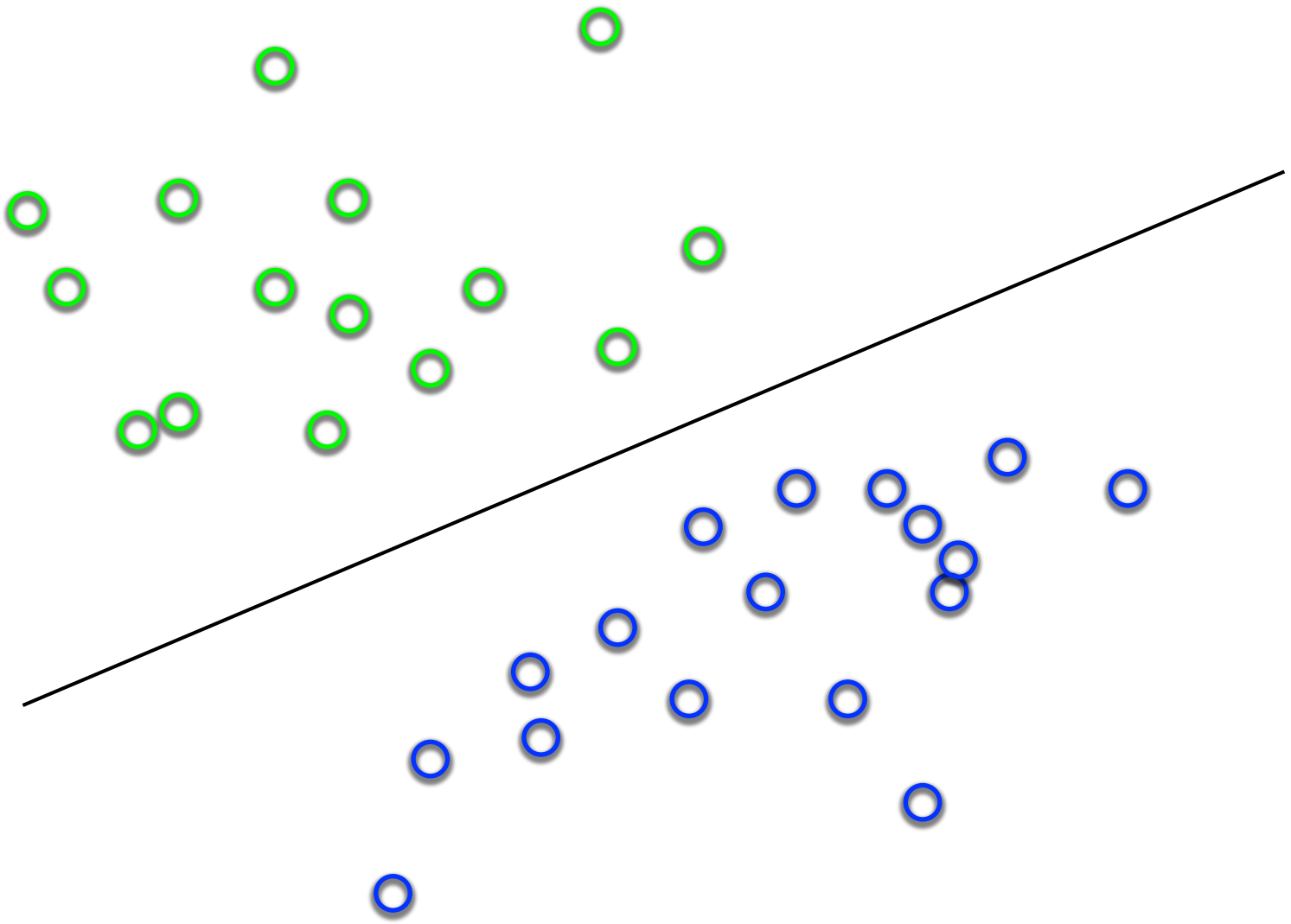


What's the best **w**?



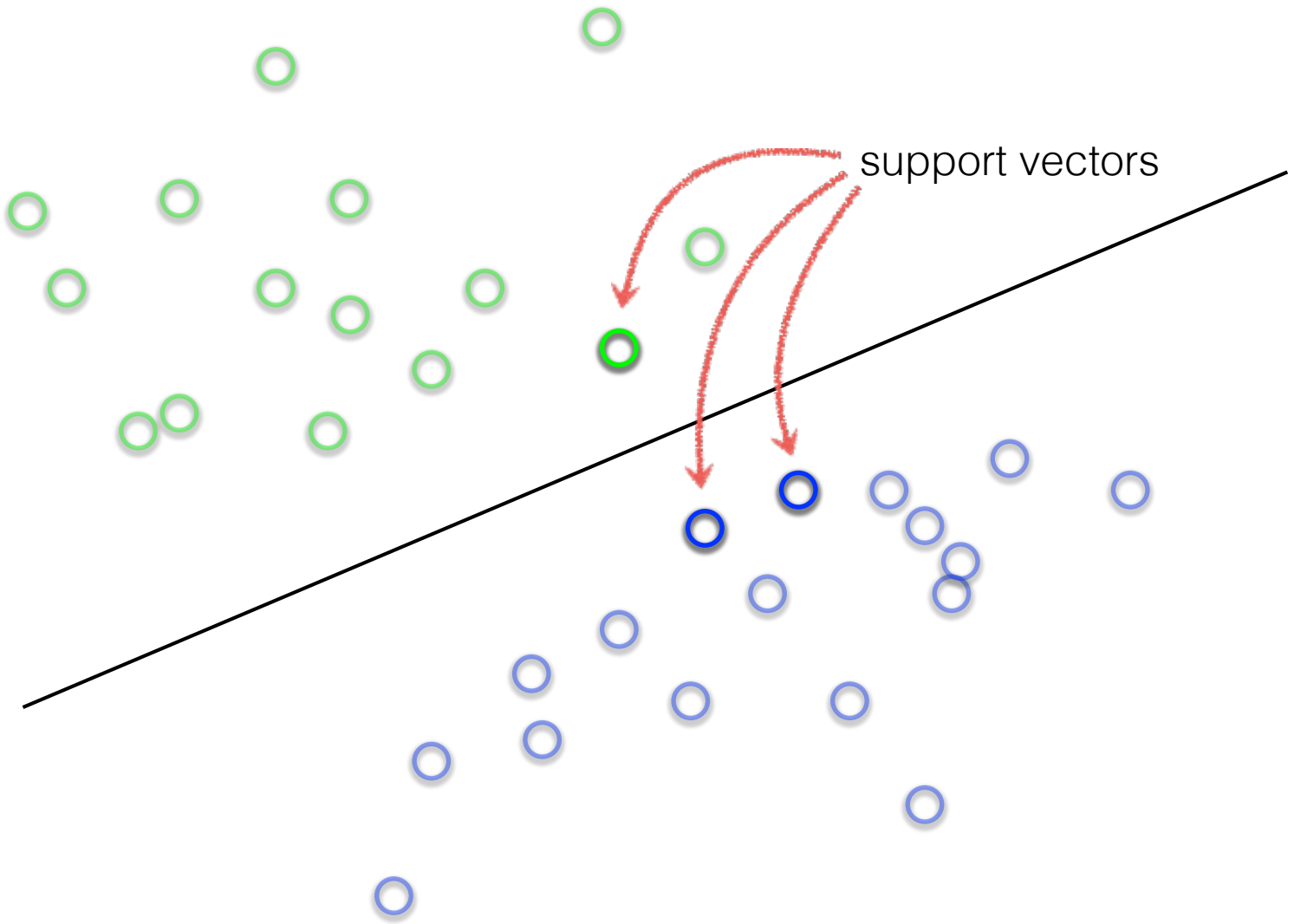
Intuitively, the line that is the farthest from all interior points

What's the best **w**?



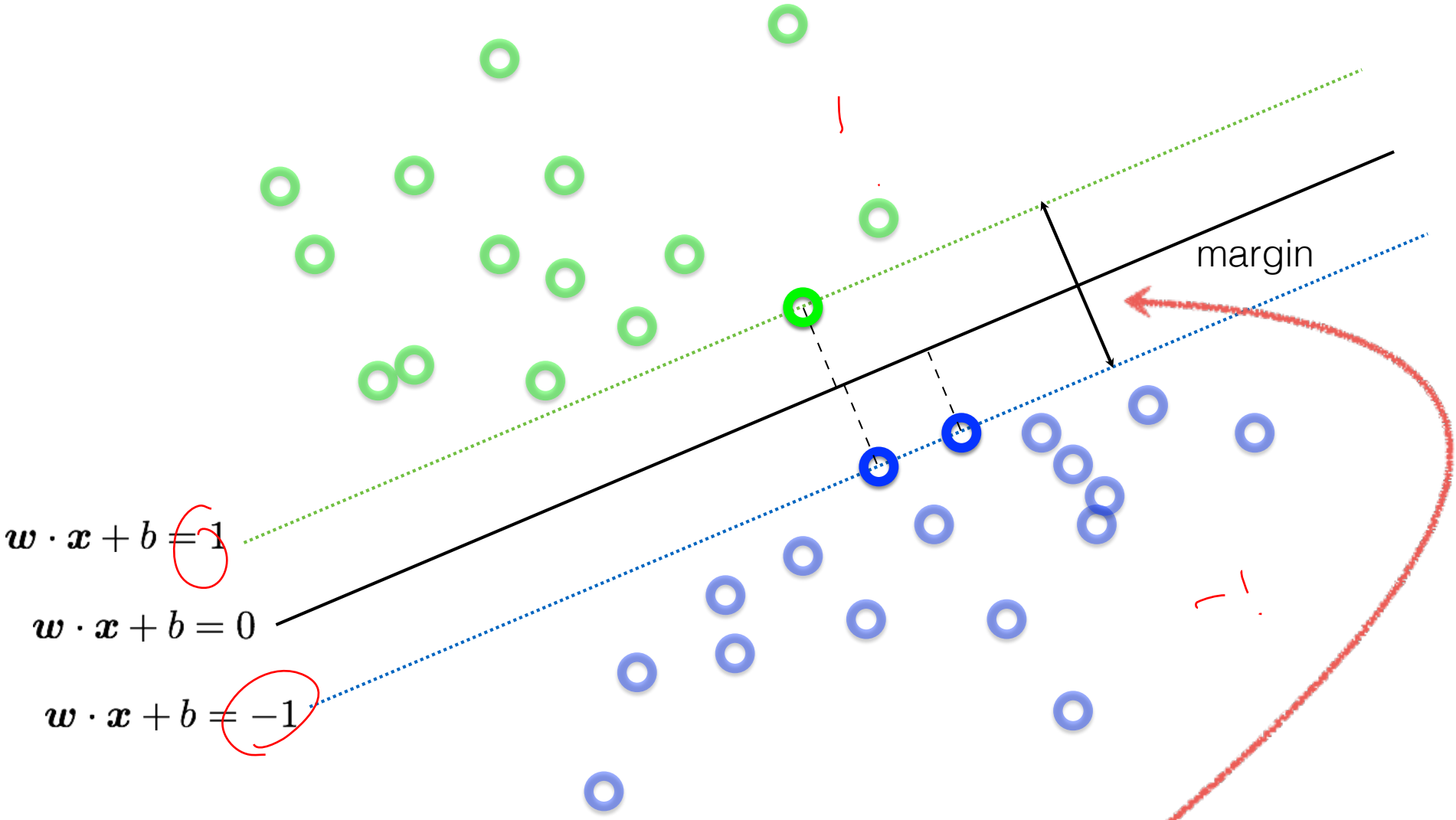
Maximum Margin solution:
most stable to perturbations of data

What's the best **w**?



Want a hyperplane that is far away from 'inner points'

Find hyperplane **w** such that …



the gap between parallel hyperplanes $\frac{2}{\|\boldsymbol{w}\|}$ is maximized

Can be formulated as a maximization problem

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

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

What does this constraint mean?

label of the data point

Why is it +1 and -1?

Can be formulated as a maximization problem

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

$$\text{subject to } \boldsymbol{w} \cdot \boldsymbol{x}_i + b \begin{array}{l} \geq +1 \text{ if } y_i = +1 \\ \leq -1 \text{ if } y_i = -1 \end{array} \text{ for } i = 1, \ldots, N$$

Equivalently,

Where did the 2 go?

$$\min_{\boldsymbol{w}} \|\boldsymbol{w}\|$$

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

What happened to the labels?

‘Primal formulation’ of a linear SVM

$$\min_{\boldsymbol{w}} \|\boldsymbol{w}\|$$

Objective Function

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

Constraints

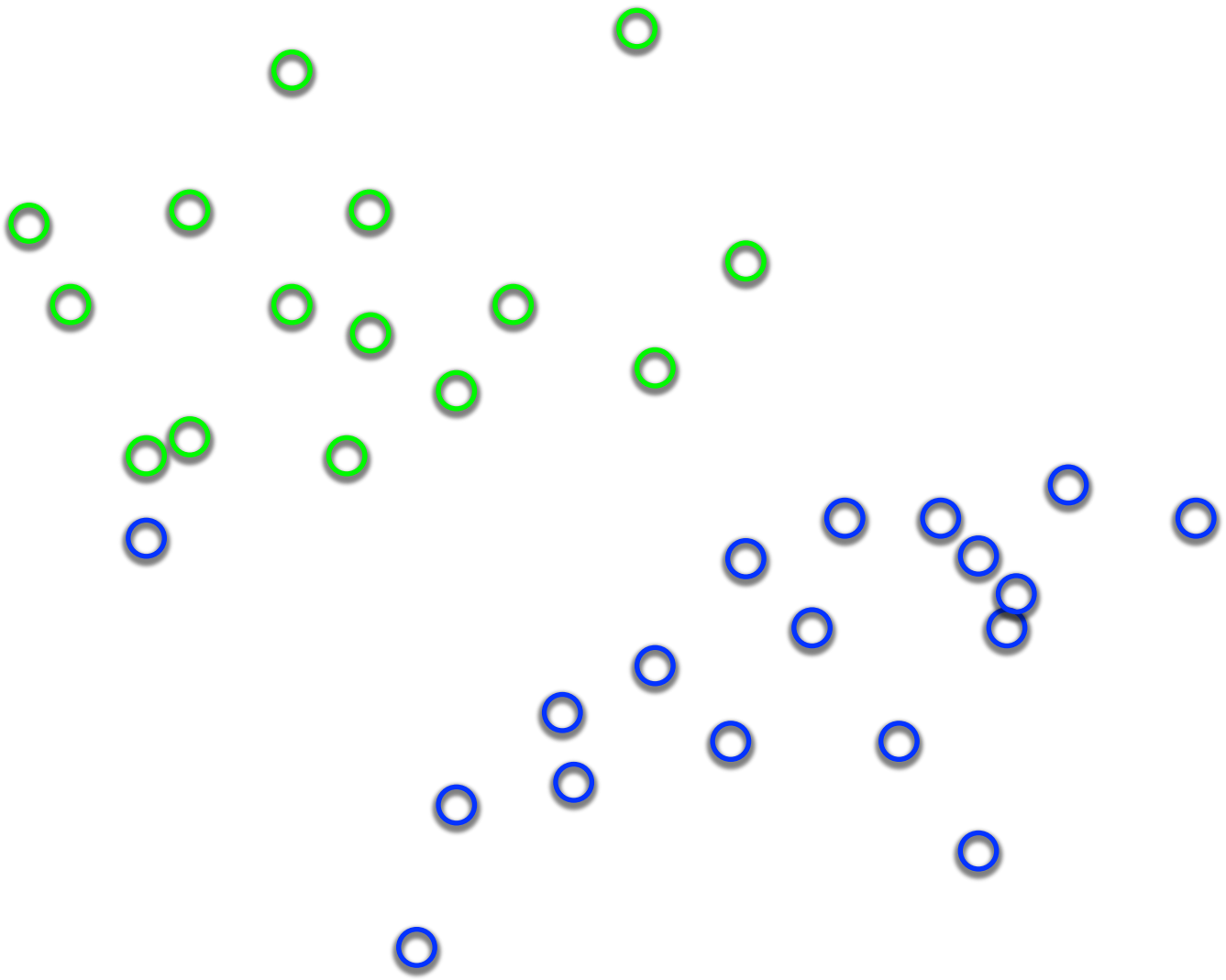
This is a convex quadratic programming (QP) problem

(a unique solution exists)

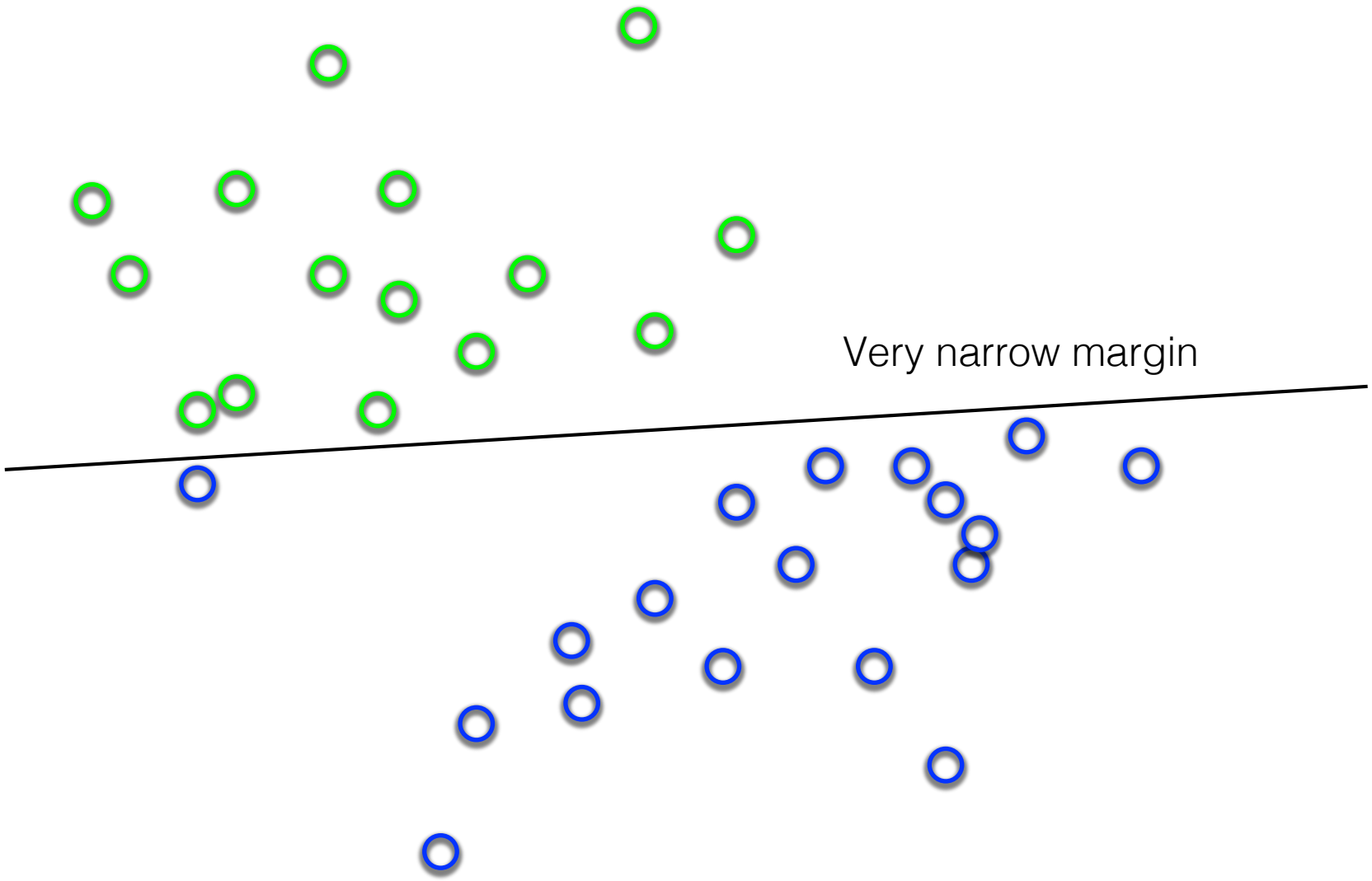
https://scikit-learn.org/stable/modules/svm.html

‘soft’ margin

What's the best **w**?

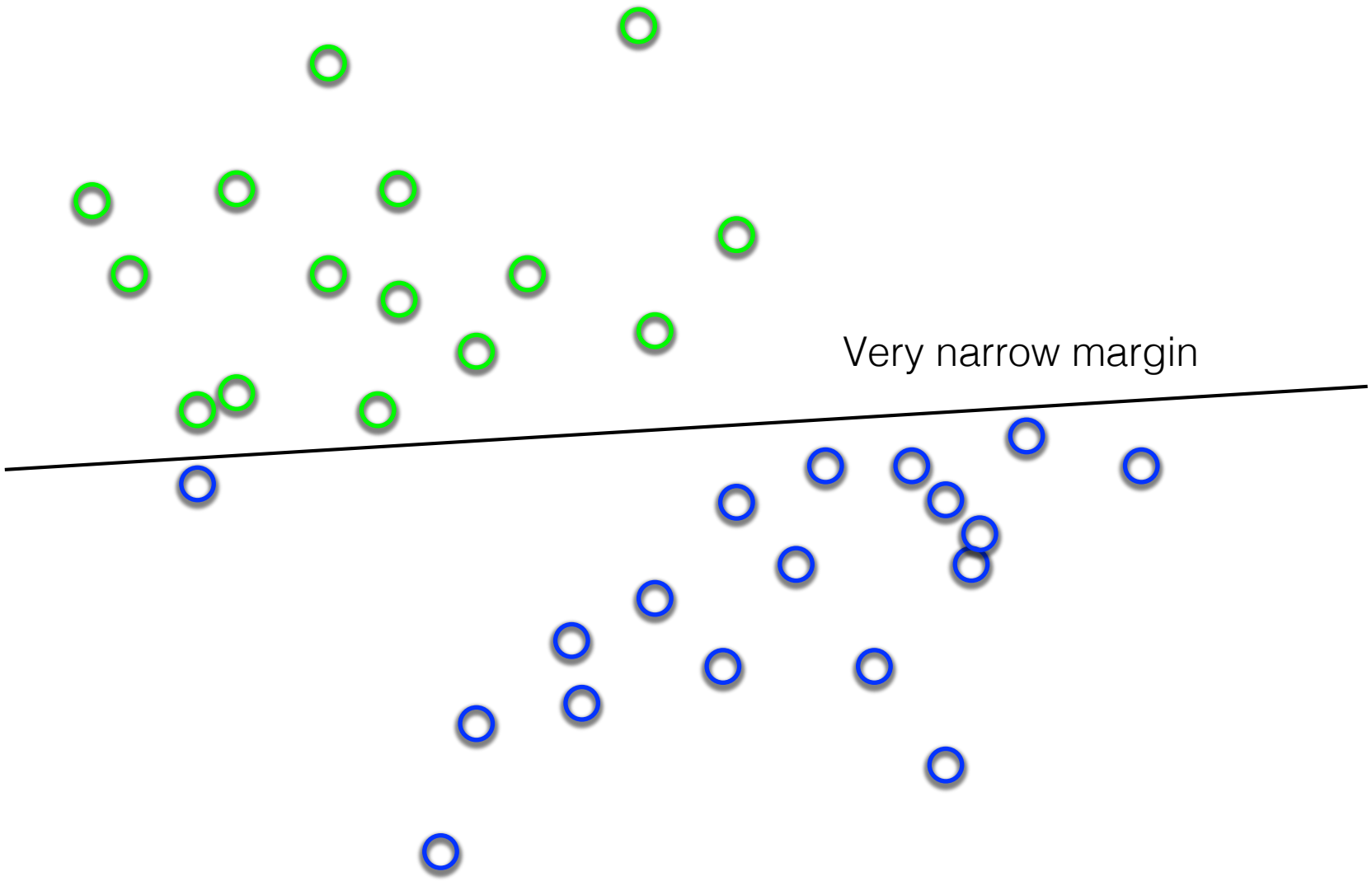


What's the best **w**?



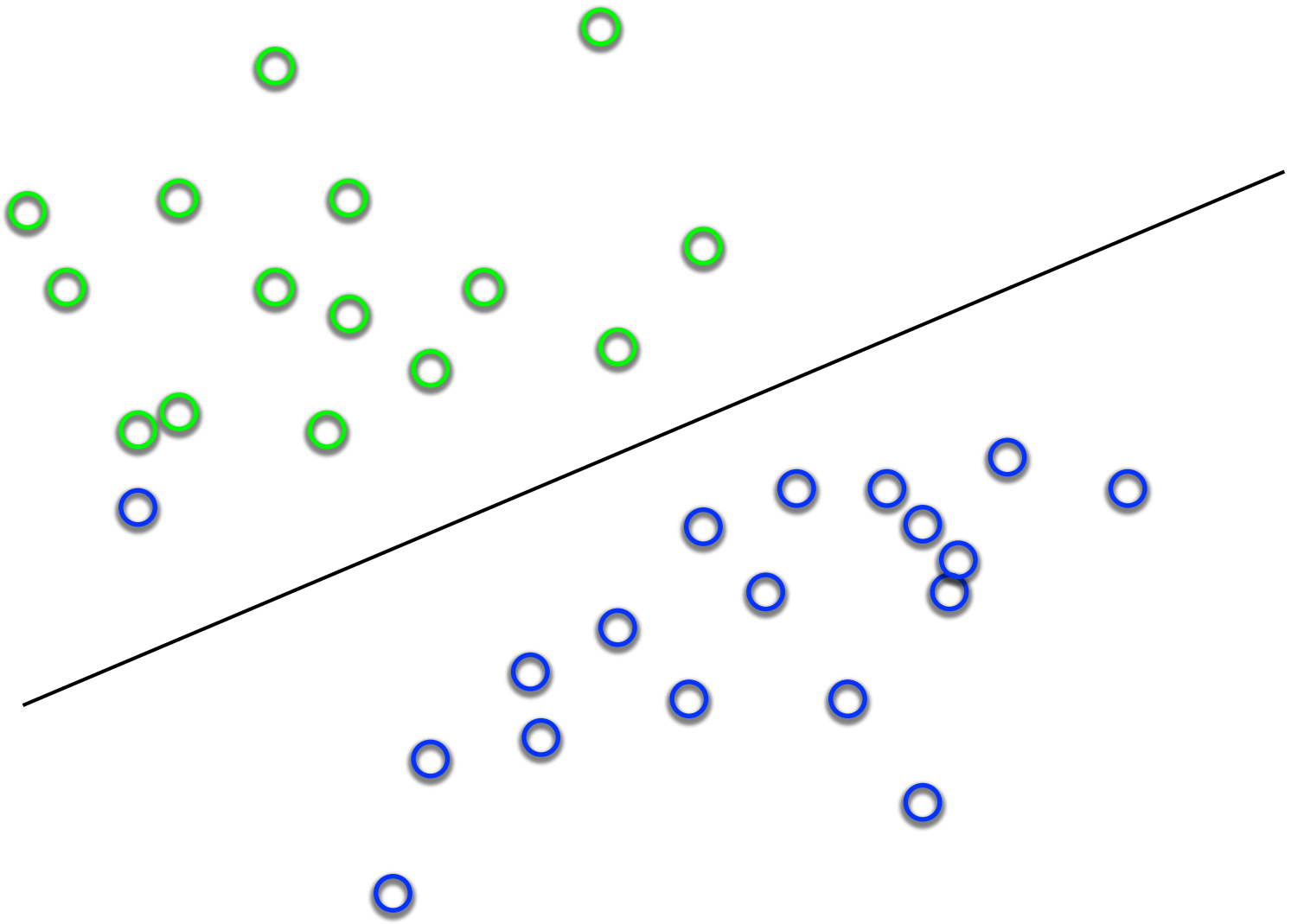
Separating cats and dogs

What's the best **w**?



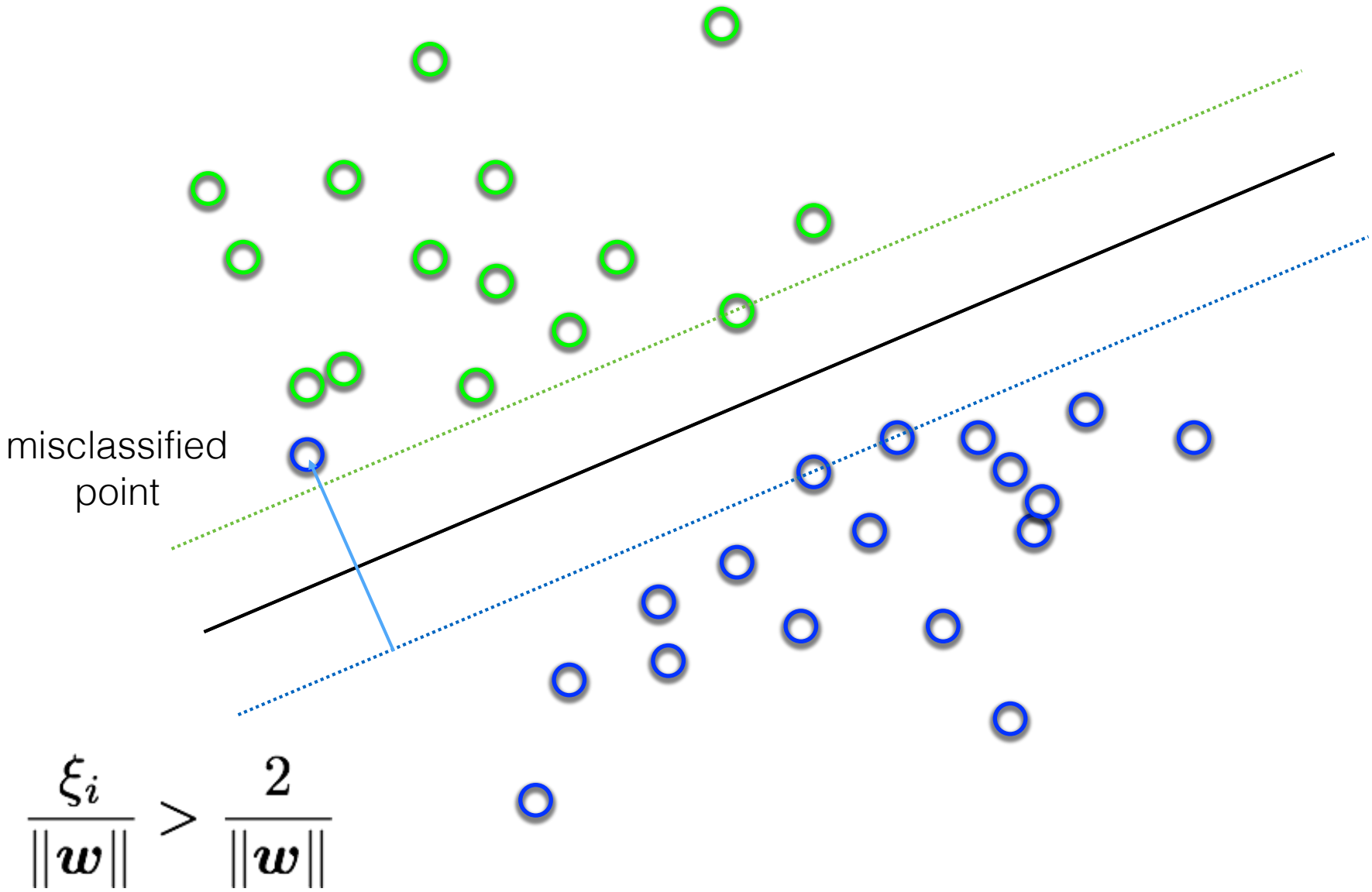
Intuitively, we should allow for some misclassification if we can get more robust classification

What's the best **w**?



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

Adding slack variables $\xi_i \geq 0$



$$\frac{\xi_i}{\|\boldsymbol{w}\|} > \frac{2}{\|\boldsymbol{w}\|}$$

‘soft’ margin

objective

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

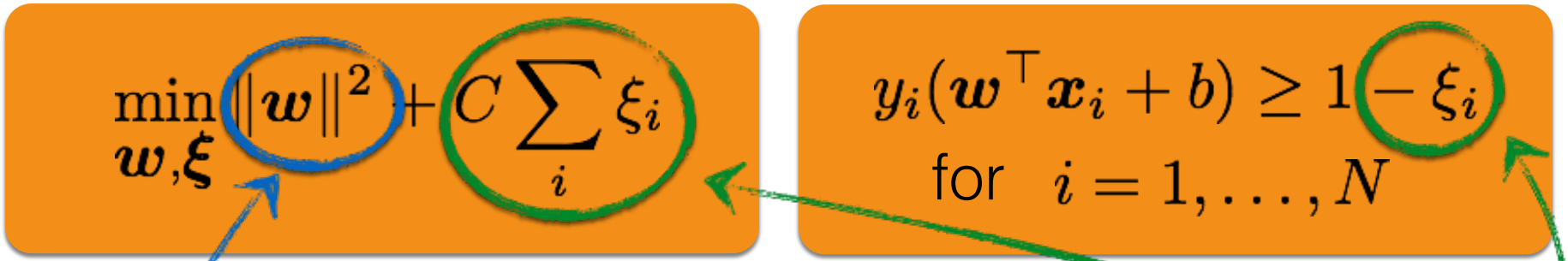
subject to

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

‘soft’ margin

objective

subject to



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

‘soft’ margin

objective

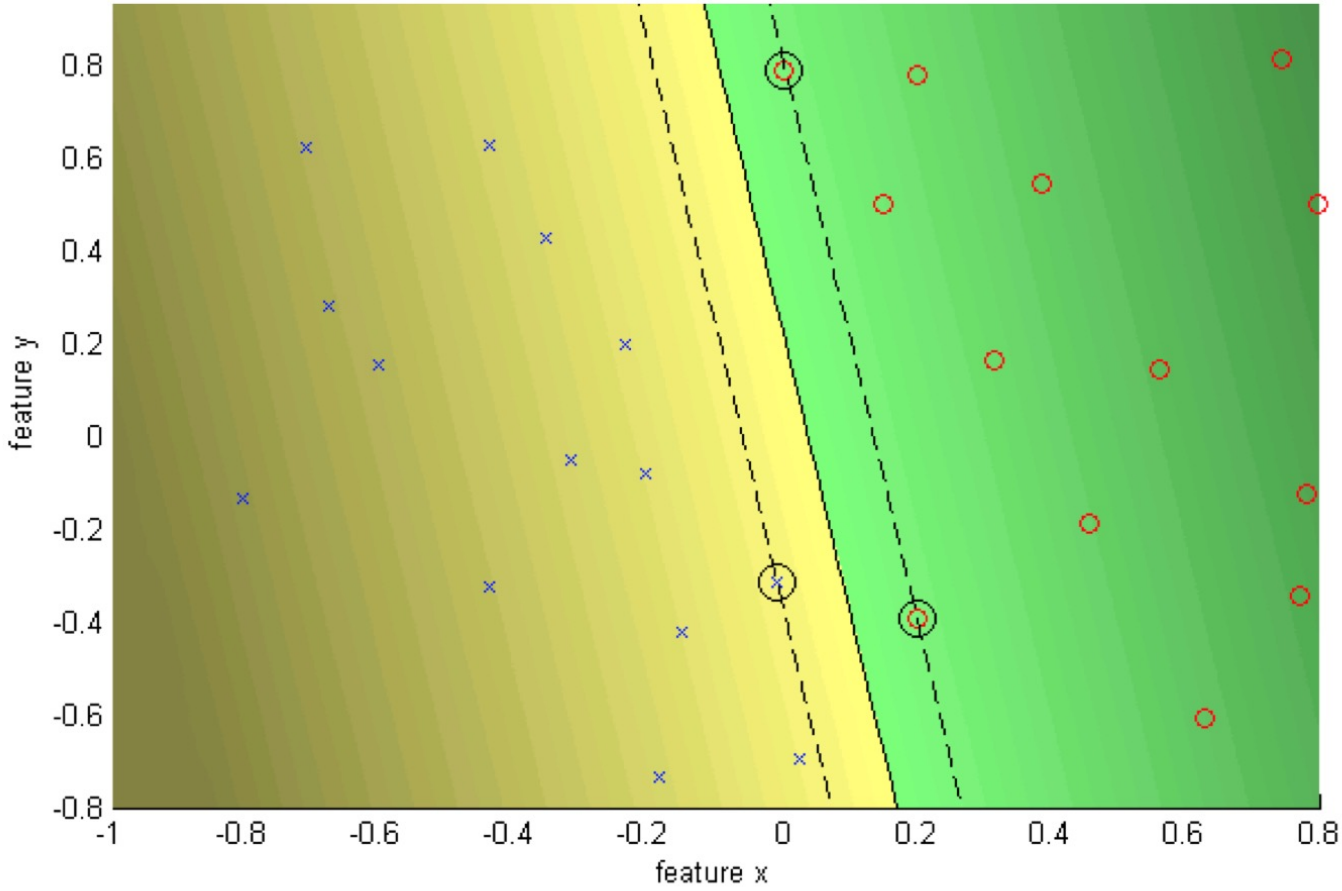
$$\min_{\boldsymbol{w},\boldsymbol{\xi}} \|\boldsymbol{w}\|^2 + C\sum_i \xi_i$$

subject to

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

- 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)

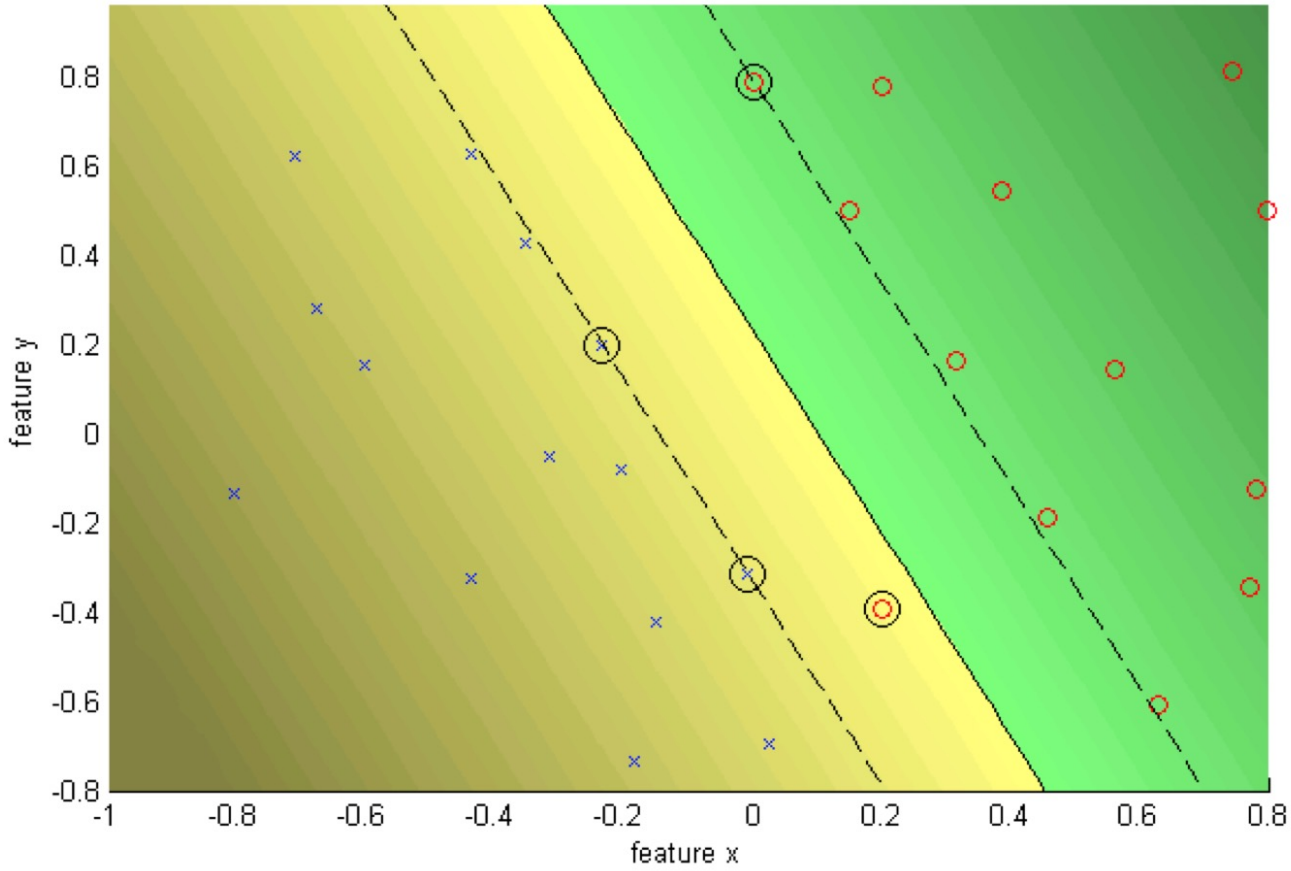
C = Infinity hard margin



Comment Window

SVM (L1) by Sequential Minimal Optimizer
Kernel: linear (-), C: Inf
Kernel evaluations: 971
Number of Support Vectors: 3
Margin: 0.0966
Training error: 0.00%

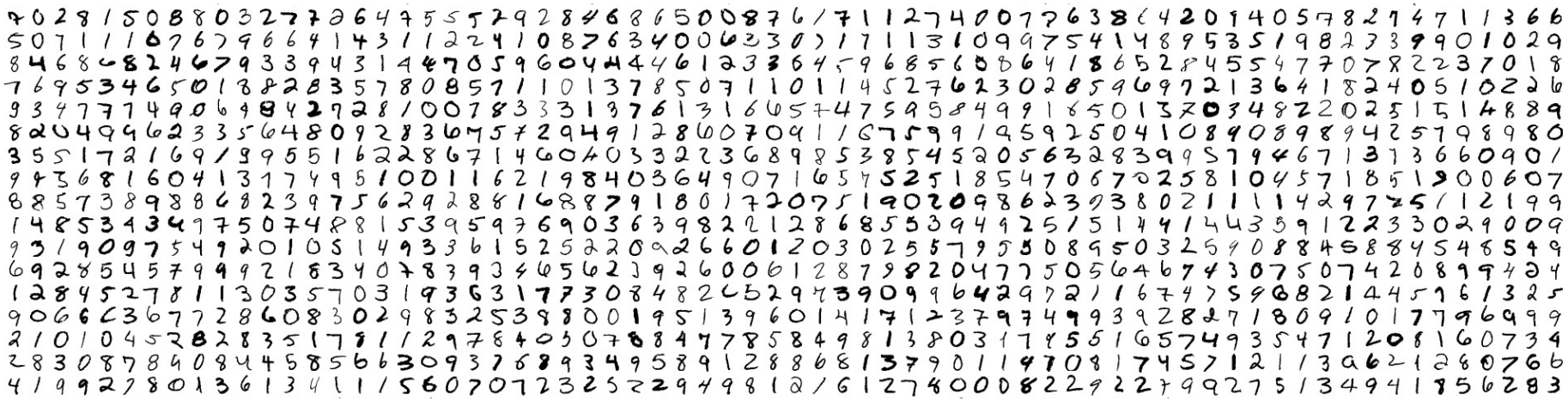
C = 10 soft margin



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%

Nearest Neighbor is competitive



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