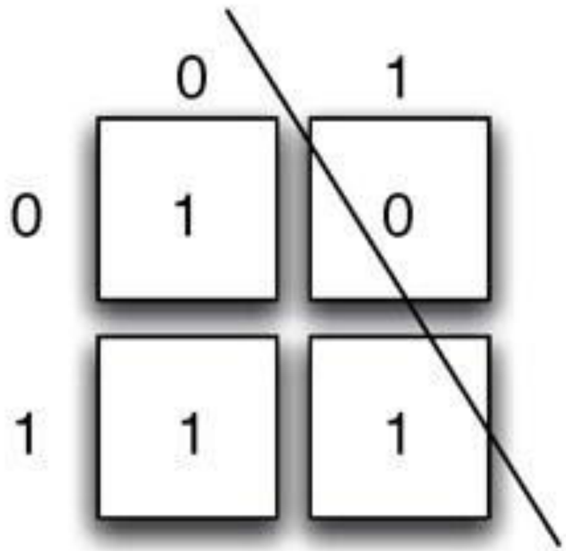
Support Vector Machines. Logistic Regression.

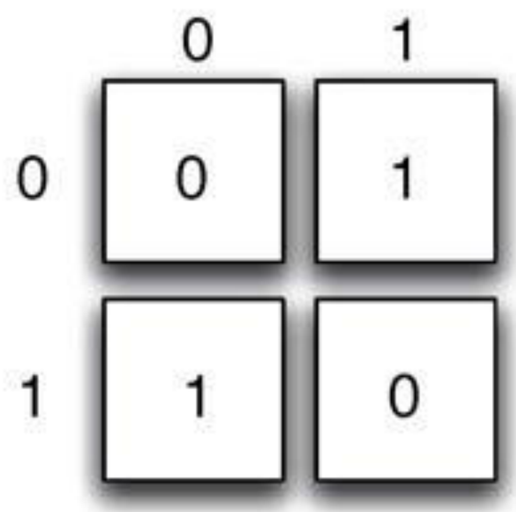
Radu Ionescu, Prof. PhD.
raducu.ionescu@gmail.com

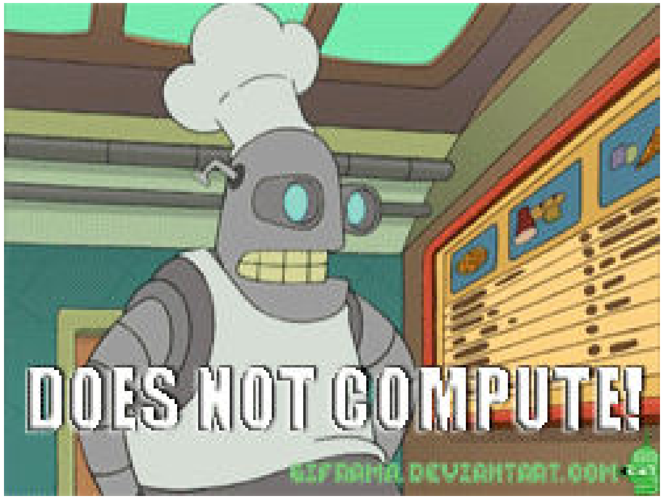
Faculty of Mathematics and Computer Science
University of Bucharest

XOR (Minsky & Papert, 1969)

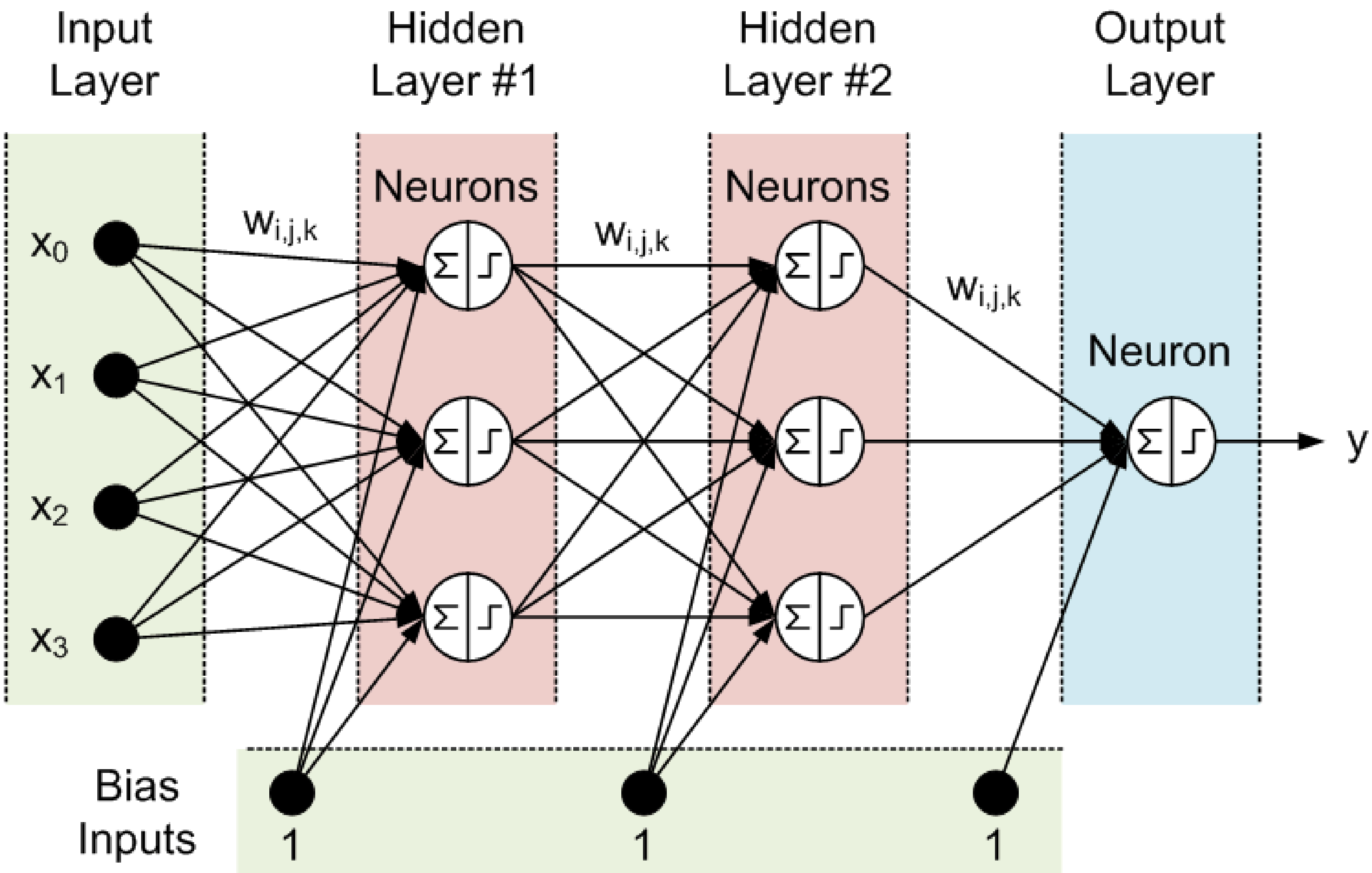
- A linear classification method cannot solve the XOR problem







Solution 1: Neural Networks



Solution 2: Kernel Methods

- Kernel methods are based on two steps:
  - ➢ 1. Embed data in a higher-dimensional Hilbert space
  - ➢ 2. Search for linear relations in the embedding space
- The embedding can be performed implicitly, by specifying the scalar product among data samples
  - ➢ Steps 1 and 2 can be comprised in one step!

Primal Form

Features: f_1, f_2, f_3, f_4, f_5, f_6, f_7

Train samples: x_1, x_2, x_3, x_4

	f_1	f_2	f_3	f_4	f_5	f_6	f_7
x_1	4	0	2	5	3	0	1
x_2	0	0	1	3	4	0	2
x_3	2	1	0	0	1	2	5
x_4	1	3	0	1	0	1	2

= **X**

l_1	1
l_2	1
l_3	-1
l_4	-1

= **L**

Linear classifier: C = (w_1, w_2, w_3, w_4, w_5, w_6, w_7, b) such that sign(X * W' + b) = L

Test samples: y_1, y_2, y_3

	f_1	f_2	f_3	f_4	f_5	f_6	f_7
y_1	1	0	2	4	2	0	2
y_2	1	2	0	1	2	2	1
y_3	3	1	0	0	4	1	1

= **Y**

p_1	?
p_2	?
p_3	?

= **P**

Apply C to obtain predictions: P = sign(Y * W' + b)

Dual form

Kernel type: **linear**

Train samples:
x_1, x_2, x_3, x_4

$$\begin{array}{c|cccc|} & x_1 & x_2 & x_3 & x_4 \\ \hline x_1 & 55 & 31 & 16 & 11 \\ x_2 & 31 & 30 & 14 & 7 \\ x_3 & 16 & 14 & 35 & 17 \\ x_4 & 11 & 7 & 17 & 16 \\ \hline \end{array} = \mathbf{X * X' = K_X}$$

$$\begin{array}{c|c|} l_1 & 1 \\ l_2 & 1 \\ l_3 & -1 \\ l_4 & -1 \\ \end{array} = \mathbf{L}$$

Linear classifier: C = $(\alpha_1, \alpha_2, \alpha_3, \alpha_4, b)$ such that $\text{sign}(K_X * \alpha' + b) = L$

Test samples:
y_1, y_2, y_3

$$\begin{array}{c|cccc|} & x_1 & x_2 & x_3 & x_4 \\ \hline y_1 & 36 & 26 & 14 & 9 \\ y_2 & 16 & 13 & 15 & 12 \\ y_3 & 25 & 18 & 18 & 9 \\ \hline \end{array} = \mathbf{Y * X' = K_Y}$$

$$\begin{array}{c|c|} p_1 & ? \\ p_2 & ? \\ p_3 & ? \\ \end{array} = \mathbf{P}$$

Apply C to obtain predictions: $P = \text{sign}(K_Y * \alpha' + b)$

Data normalization

- In primal form:

$$x \longmapsto \phi(x) \longmapsto \frac{\phi(x)}{\|\phi(x)\|}$$

- In dual form:

$$\hat{k}(x_i, x_j) = \frac{k(x_i, x_j)}{\sqrt{k(x_i, x_i) \cdot k(x_j, x_j)}}$$

- Directly on the kernel matrix:

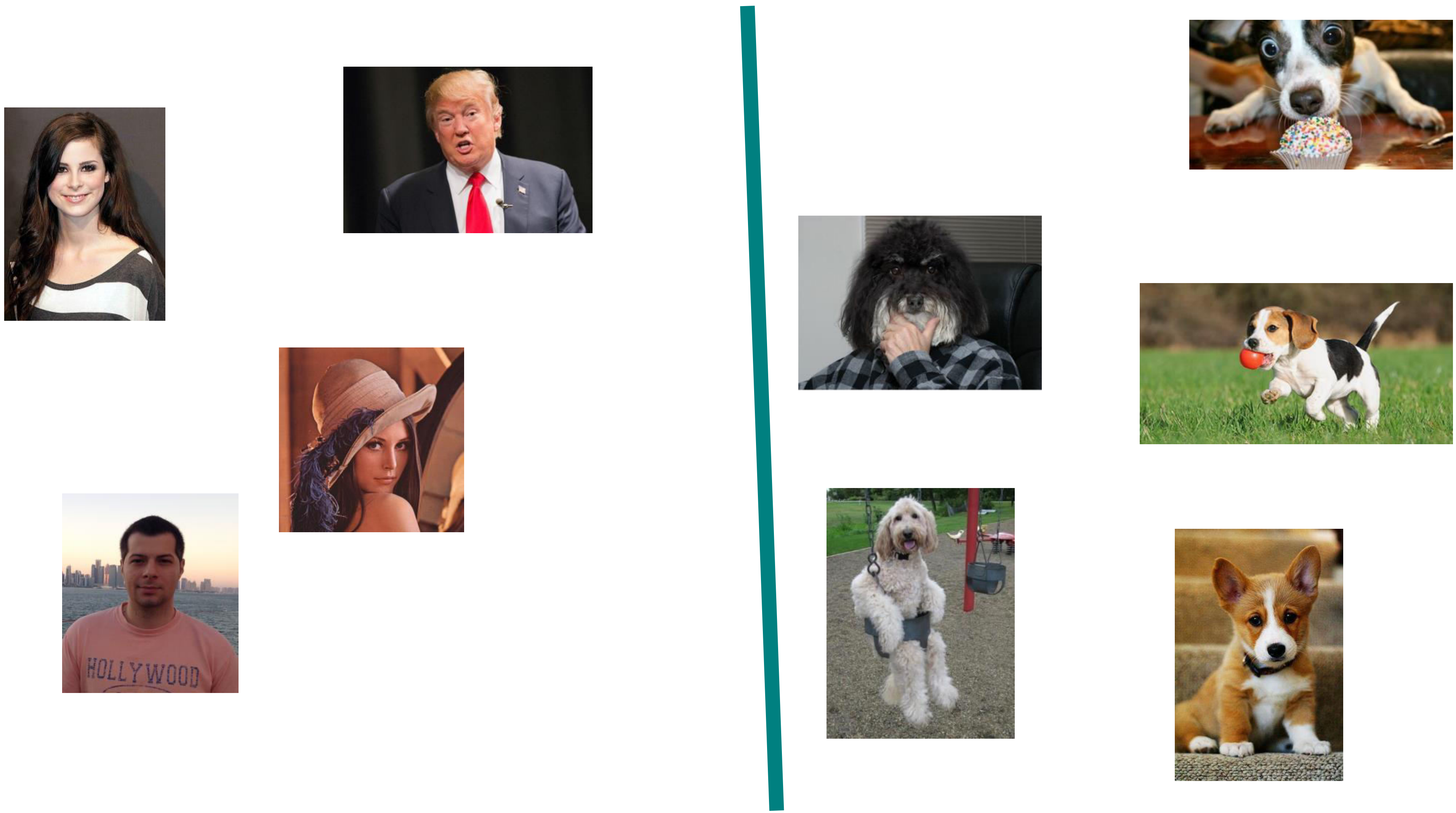
$$\hat{K}_{ij} = \frac{K_{ij}}{\sqrt{K_{ii} \cdot K_{jj}}}$$

Data normalization (Python)

```
% X - data (one sample per row)

% L2 norm in primal form:
norms = np.linalg.norm(X, axis = 1, keepdims = True)
X = X / norms

% L2 norm in dual form:
K = np.matmul(X, X.T)
KNorm = np.sqrt(np.diag(K))
KNorm = KNorm[np.newaxis]
K = K / np.matmul(KNorm.T, KNorm)
```

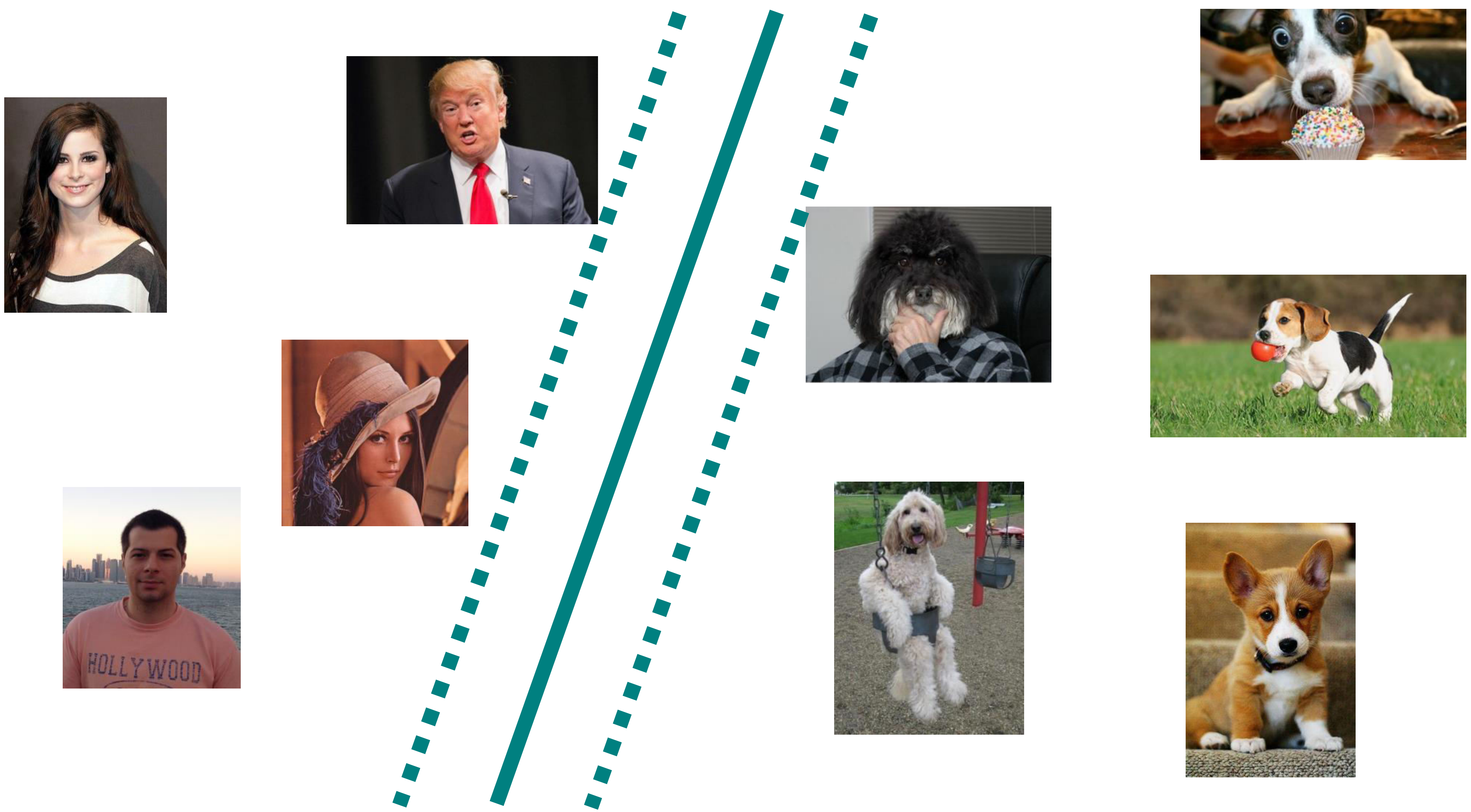
How do we separate these points optimally?

How do we separate these points optimally?



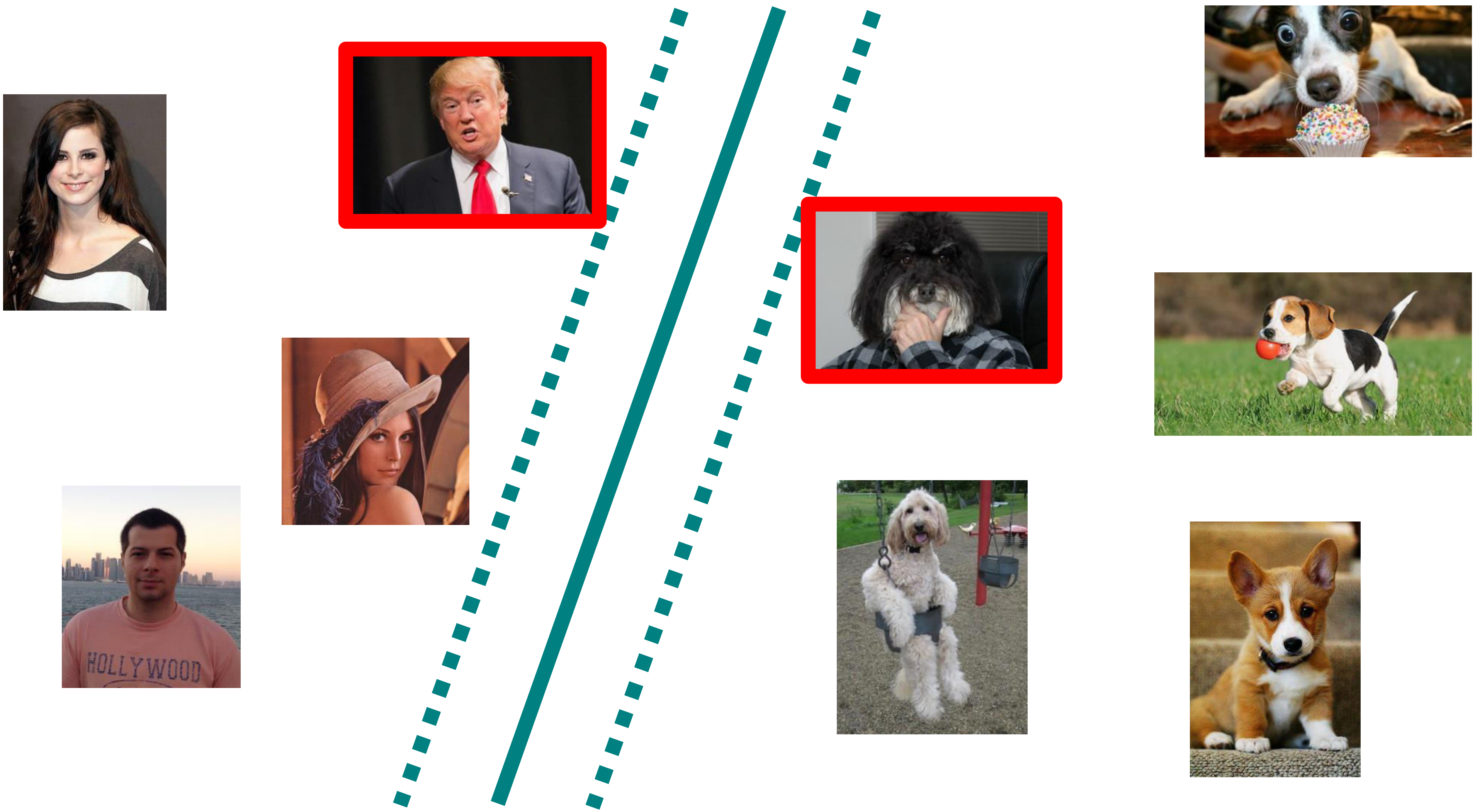
How do we separate these points optimally?

Pick the maximum margin hyperplane



Pick the maximum margin hyperplane

- Support Vector Machines (SVM)



SVM (Hard Margin)

$$S = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x}_\ell, y_\ell)\}$$

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

$$\max_{\mathbf{w}, b, \gamma} \gamma$$

subject to

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq \gamma$$

$$i = 1, \ldots, \ell$$

$$\|\mathbf{w}\|^2 = 1$$

$\Longleftrightarrow$

$$\min_{\mathbf{w}, b} \|\mathbf{w}\|^2$$

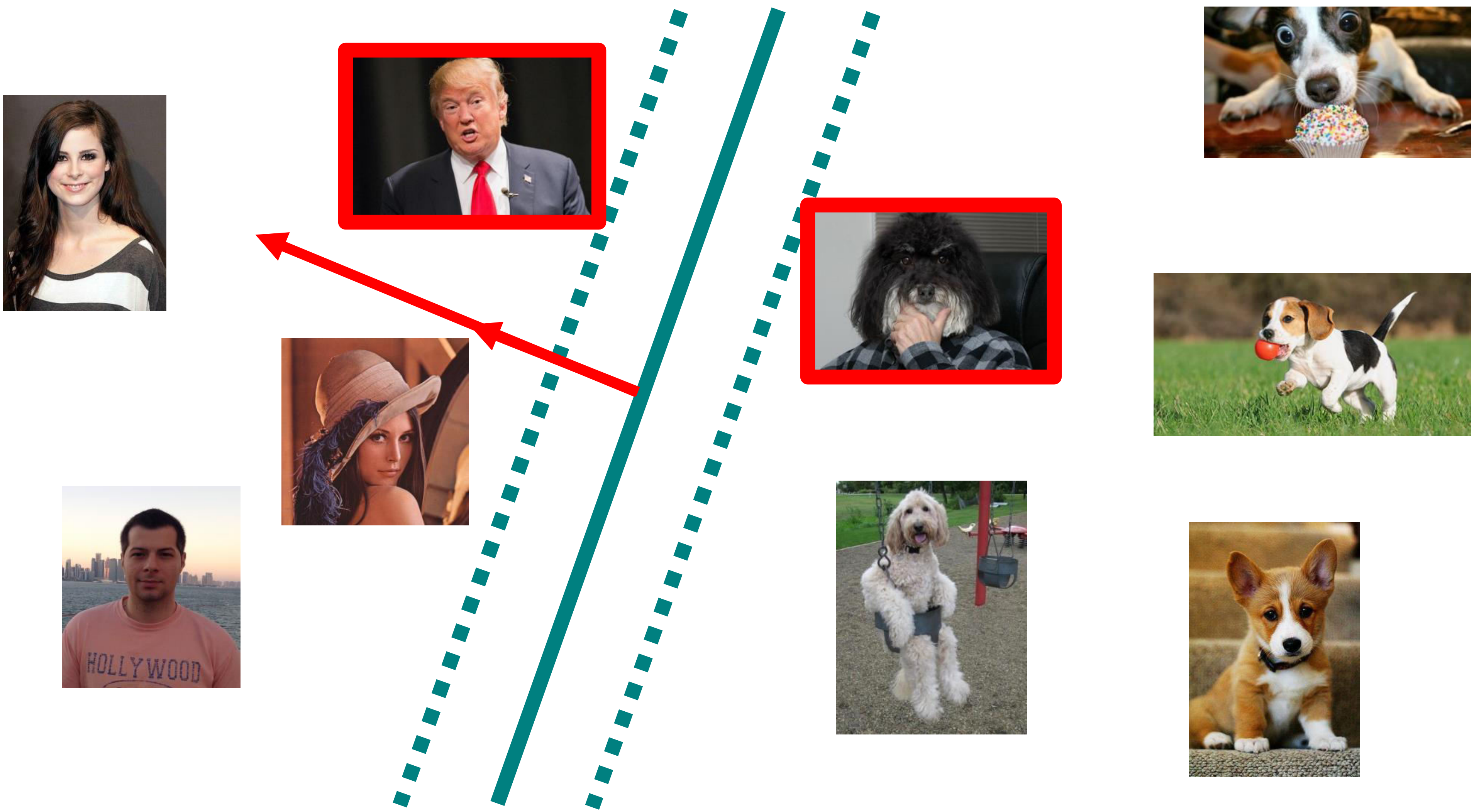
subject to

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$$

$$i = 1, \ldots, \ell$$

Pick the maximum margin hyperplane

- Support Vector Machines (SVM)



SVM Dual (Hard Margin)

- Lagrange multipliers: a way of finding the extremum of a function subject to constraints
- We want to minimize $\|w\|^2$ subject to $y_i(\langle w, x_i \rangle + b) - 1 \geq 0$
- We define a new function and find its minimum instead

$$L(w, b, \alpha) = \|w\|^2 - \sum_i \alpha_i [y_i(\langle w, x_i \rangle + b) - 1], \alpha_i \geq 0$$

$$\frac{\partial L}{\partial w} = w - \sum_i \alpha_i y_i x_i = 0 \Rightarrow w = \sum_i \alpha_i y_i x_i$$

$$\frac{\partial L}{\partial b} = -\sum_i \alpha_i y_i = 0 \Rightarrow \sum_i \alpha_i y_i = 0$$

- α_i are called Lagrange multipliers

SVM Dual (Hard Margin)

$$\left.\begin{aligned} &L = \|w\|^2 - \textstyle\sum_i \alpha_i [y_i(\langle w, x_i \rangle + b) - 1] \\ &w = \textstyle\sum_i \alpha_i y_i x_i \\ &\textstyle\sum_i \alpha_i y_i = 0 \end{aligned}\right\} L = \sum_i \alpha_i - \sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

- Note: The optimization depends only on the scalar product of training sample pairs
- α_i will be non-zero only for support vectors
- Our decision rule (in primal form) was: x is positive if $\langle w, x \rangle + b \geq 0$
- In dual form, it becomes: $\sum_i \alpha_i y_i \langle x_i, x \rangle + b \geq 0$
- Note: The decision also depends on the scalar product
- Note: b can be computed from $\sum_i \alpha_i y_i \langle x_i, x_+ \rangle + b = 1$, for some positive support vector x_+

SVM Dual (Hard Margin)

- The decision rule (in dual form) is:

x is positive if $\sum_i \alpha_i y_i \langle x_i, x \rangle + b \geq 0$

- In order to obtain α_i and b, we have to:

$$min \sum_i \alpha_i - \sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

$$subject\ to\ \alpha_i \geq 0, \forall i$$

SVM (Hard Margin)

Primal form

- Decision rule:

$$\langle w, x \rangle + b \geq 0$$

- Optimization problem:

$$min \|w\|^2$$

$$subject\ to\ y_i(\langle w, x_i \rangle + b) - 1 \geq 0$$

Dual form

- Decision rule:

$$\sum_i \alpha_i y_i \langle x_i, x \rangle + b \geq 0$$

- Optimization problem:

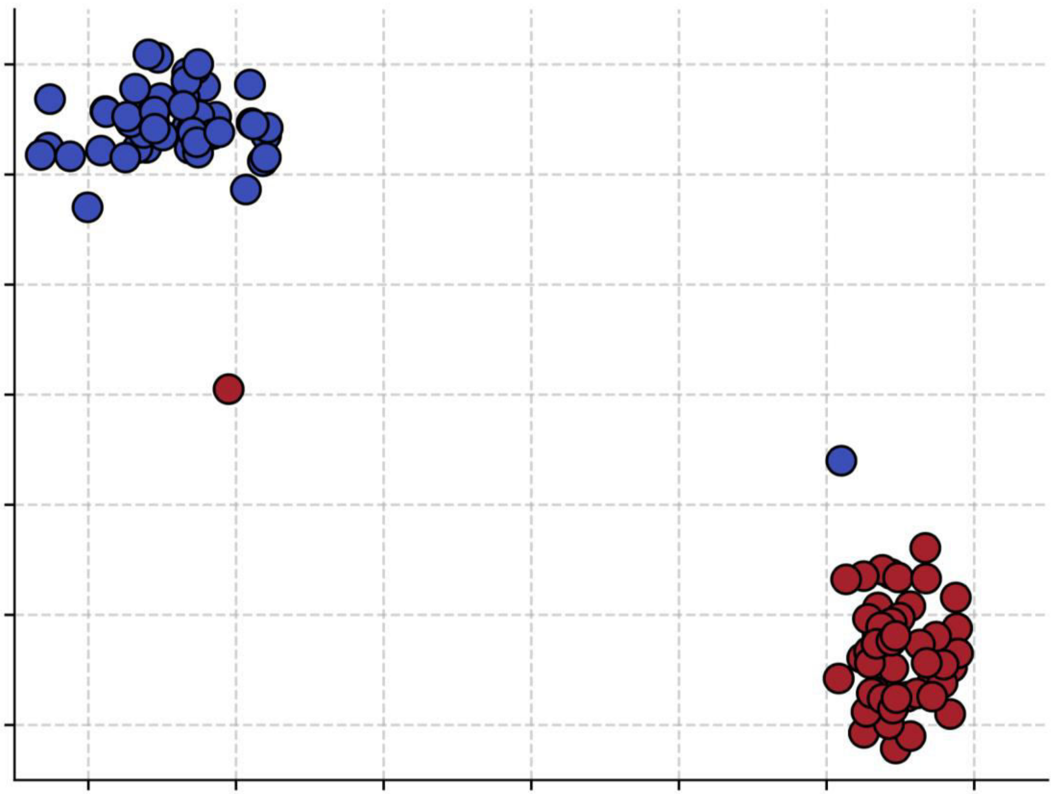
$$min \sum_i \alpha_i - \sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

$$subject\ to\ \alpha_i \geq 0$$

- Primal has as many params as features (n)
- Dual has as many params as samples (l)
- It is more efficient to optimize primal if $l \gg n$

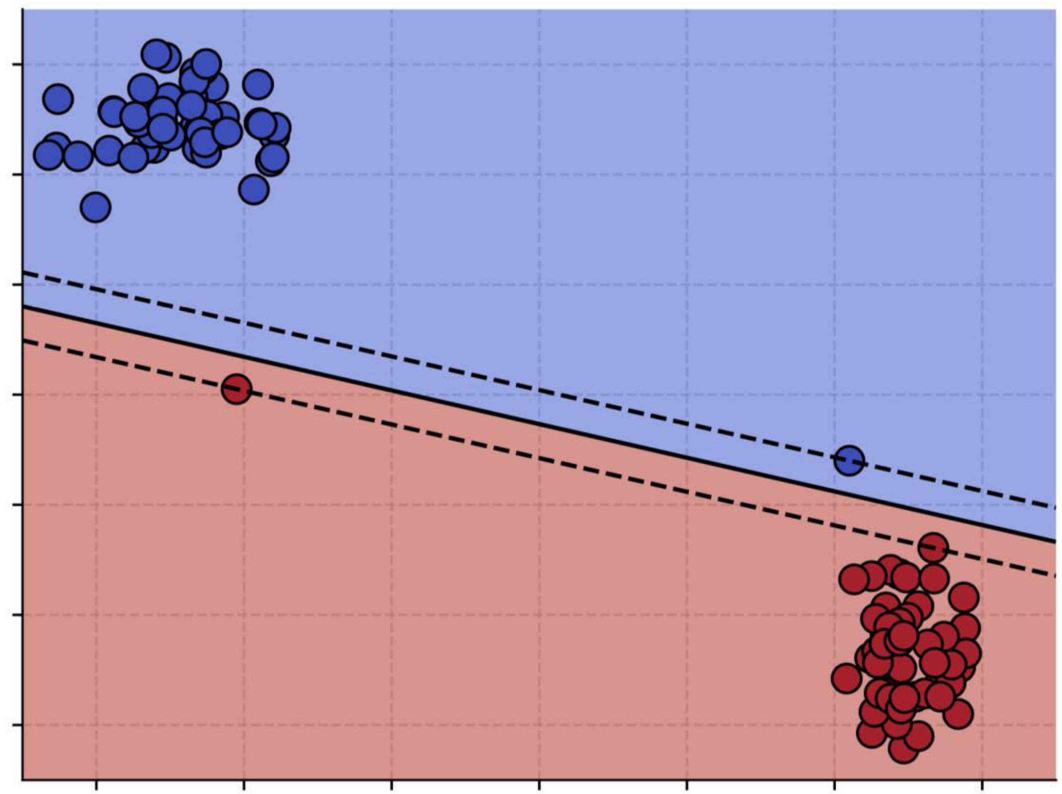
SVM (Soft Margin)

- What happens when we train a hard-margin SVM on this data set?



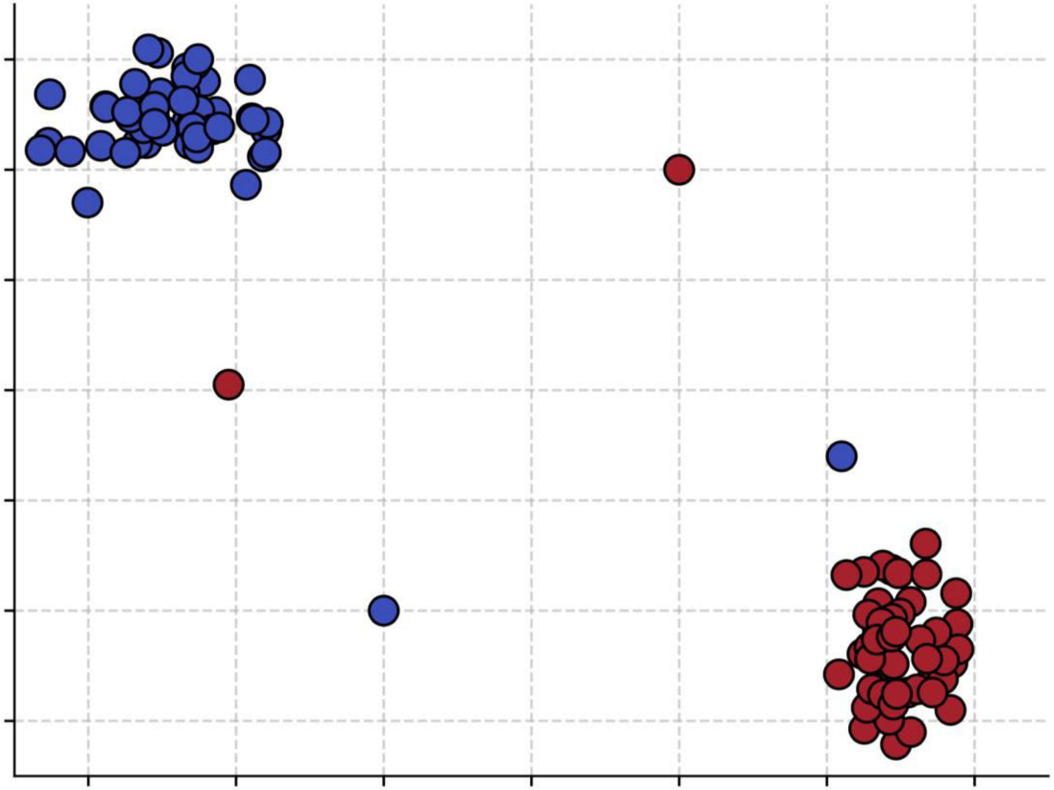
SVM (Soft Margin)

- What happens when we train a hard-margin SVM on this data set?
- It succeeds at separating the training samples, but with a very tight margin



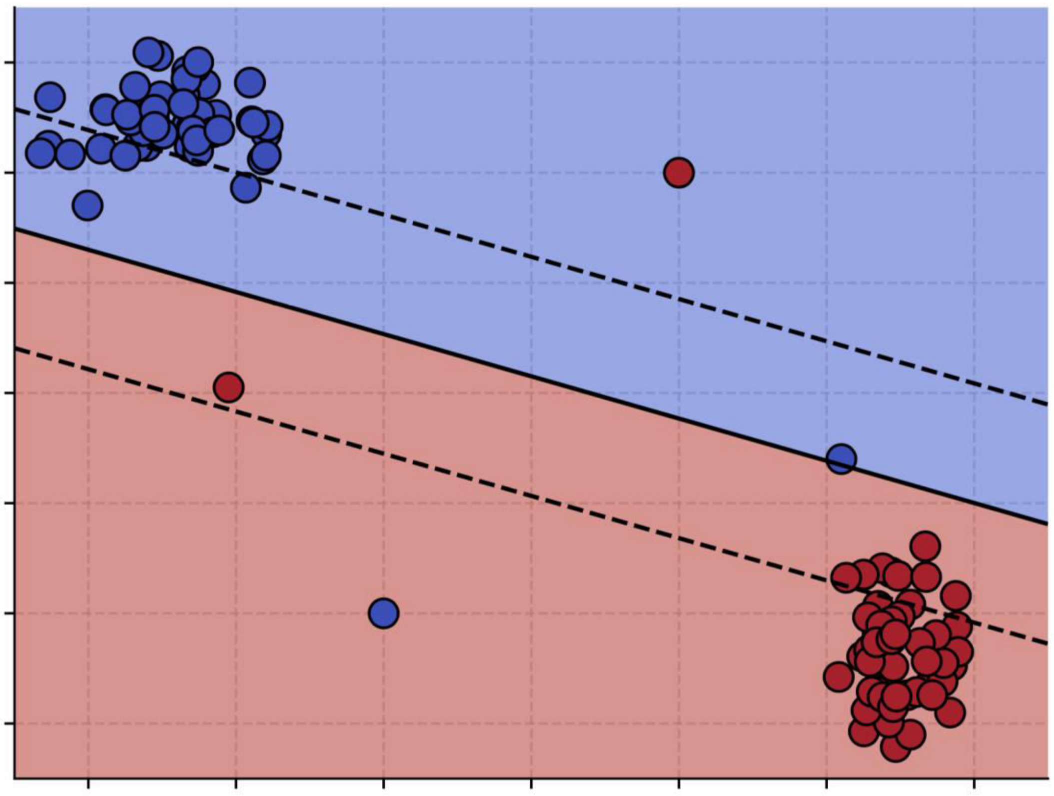
SVM (Soft Margin)

- What happens when we train a hard-margin SVM on this data set?



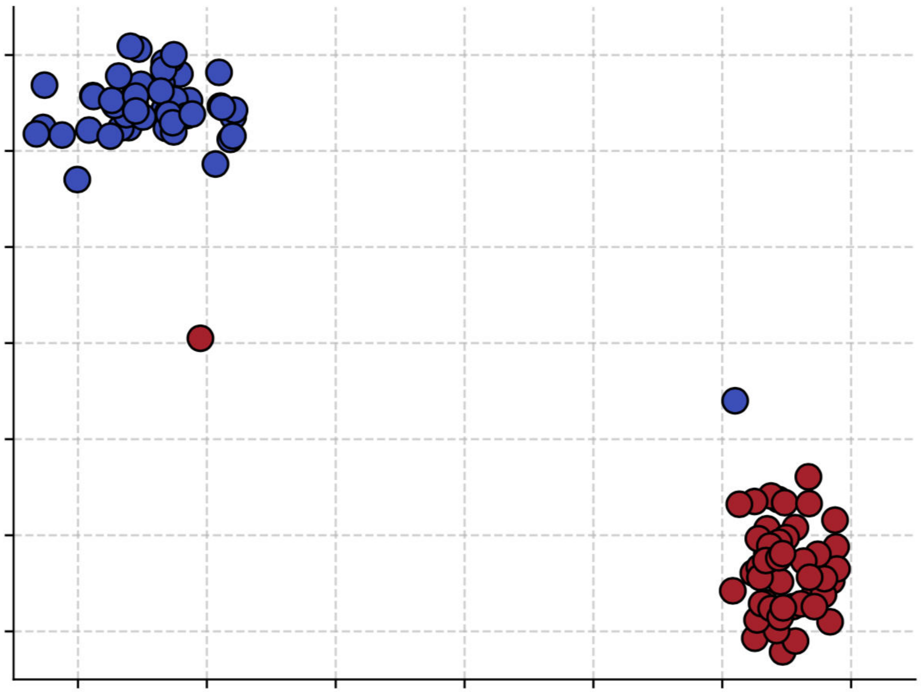
SVM (Soft Margin)

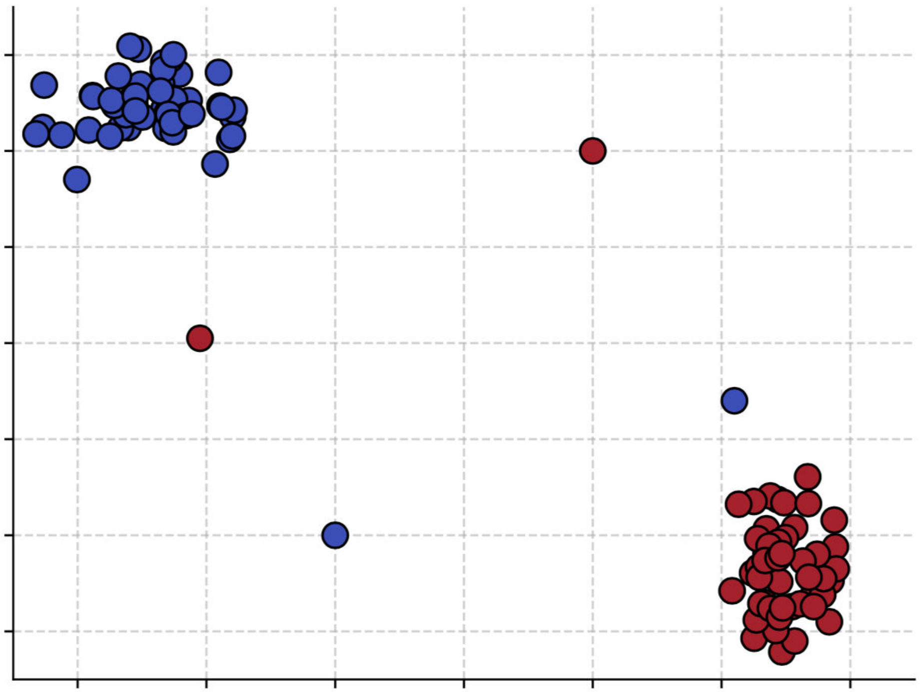
- What happens when we train a hard-margin SVM on this data set?
- It fails, because the samples are not linearly separable



SVM (Soft Margin)

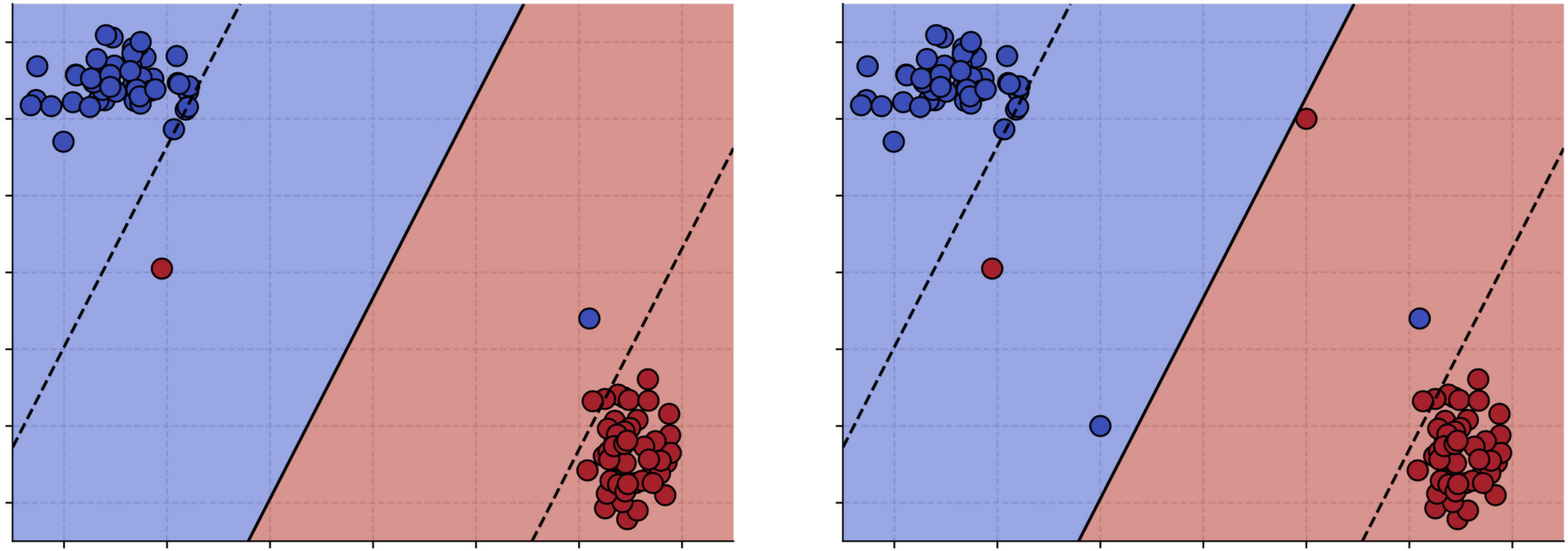
- What would the more “reasonable” solutions be?





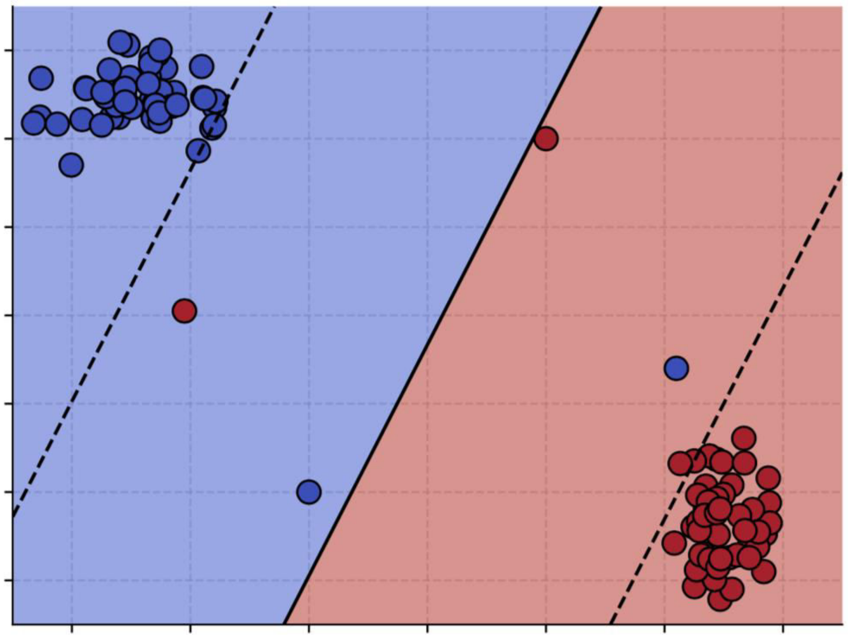
SVM (Soft Margin)

- What would the more “reasonable” solutions be?



SVM (Soft Margin)

- Allow some training samples to be misclassified, at a cost.

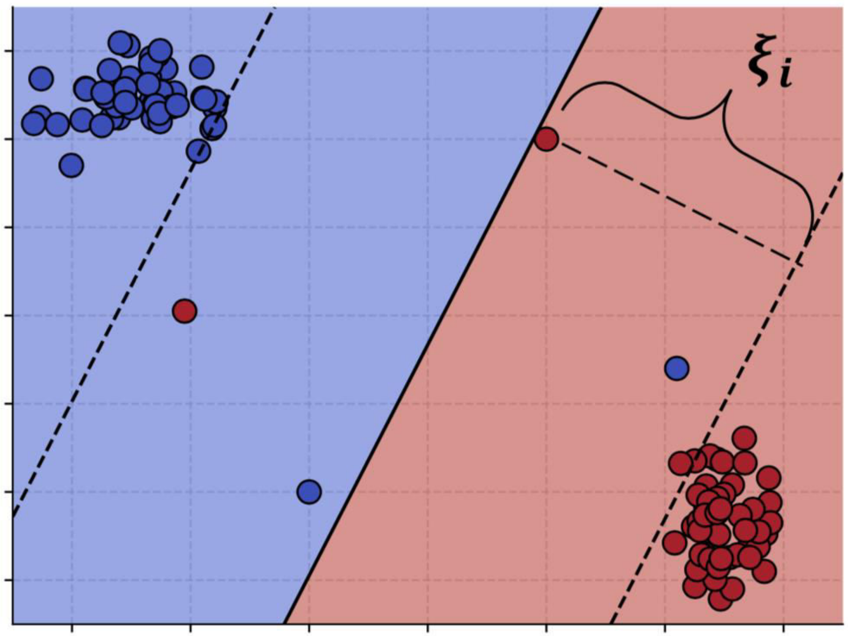


SVM (Soft Margin)

- Allow some training samples to be misclassified, at a cost.
- We introduce slack variables $\xi_i \geq 0$ (how much example x_i is allowed to cross the border):

$$y_i(\langle w, x_i \rangle + b) - 1 \geq 0 \text{ becomes}$$

$$y_i(\langle w, x_i \rangle + b) - 1 \geq -\xi_i$$



- The optimization problem becomes:

$$min\|w\|^2 + C \sum_i \xi_i$$

$$subject\ to\ y_i(\langle w, x_i \rangle + b) - 1 \geq -\xi_i$$

- Trade-off between making the margin wide and allowing training mistakes
- C controls the weight of the mistakes

SVM (Soft Margin)

- When the examples are non-linearly separable:

$$\min_{\mathbf{w},b,\gamma,\xi} -\gamma + C\sum_{i=1}^{\ell}\xi_i$$

subject to

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq \gamma - \xi_i$$

$$\xi_i \geq 0 \quad i = 1, \ldots, \ell$$

$$\|\mathbf{w}\|^2 = 1$$

$$\Longleftrightarrow$$

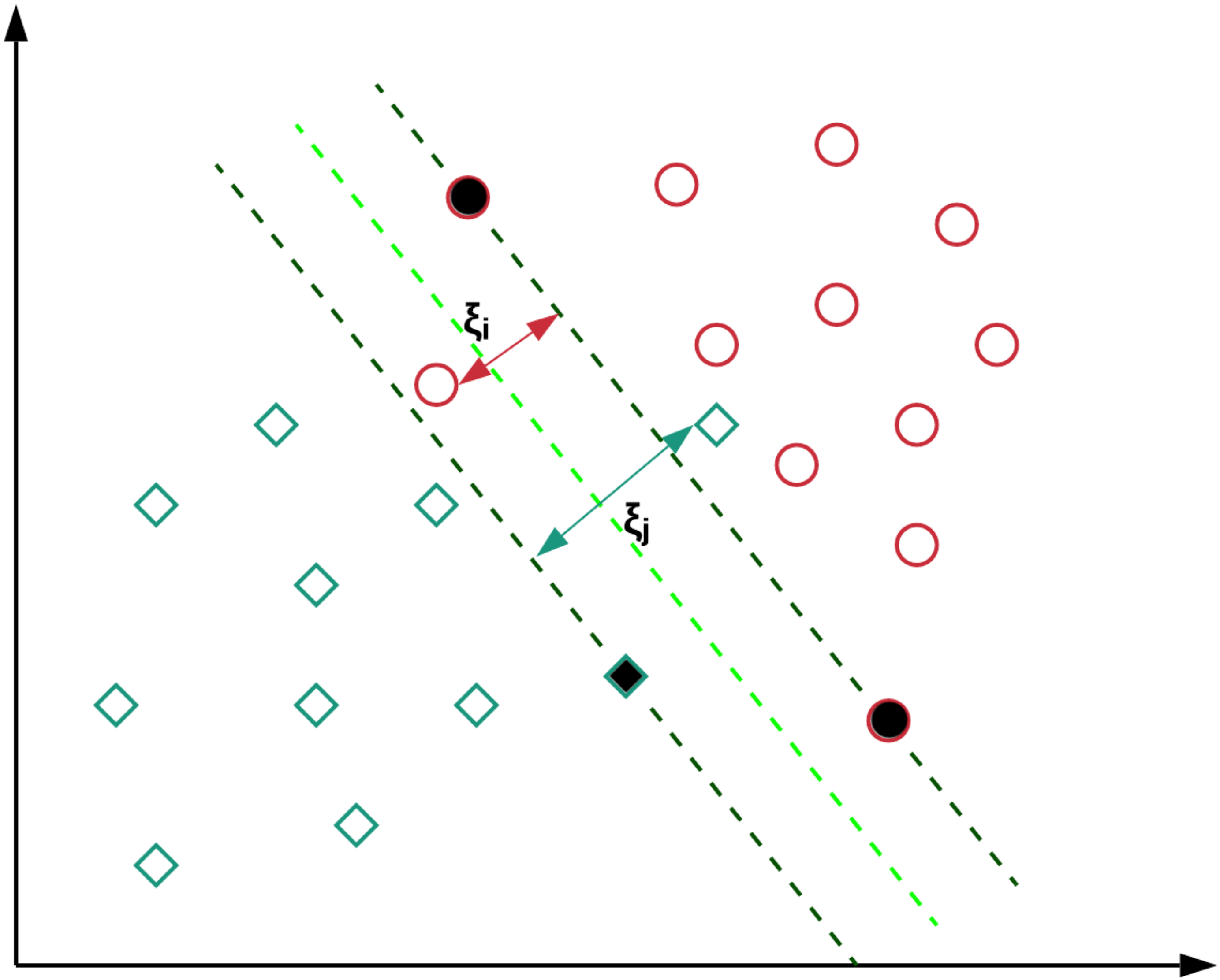
$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{\ell}\xi_i$$

subject to

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0 \quad i = 1, \ldots, \ell$$

SVM (Soft Margin)



Hinge Loss

$$\left.\begin{aligned} & min\|w\|^2 + C\sum_i \xi_i \\ & subject\ to\ y_i(\langle w, x_i\rangle + b) - 1 \geq -\xi_i \\ & \qquad\qquad \xi_i \geq 0 \end{aligned}\right\} \xi_i = \max\big(0, 1 - y_i(\langle w, x_i\rangle + b)\big)$$

- By replacing ξ_i in the minimization problem, we get:

$$min\|w\|^2 + C\sum_i \max\big(0, 1 - y_i(\langle w, x_i\rangle + b)\big)$$

Hinge Loss

$$min\|w\|^2 + C\sum_i \max\big(0, 1 - y_i(\langle w, x_i\rangle + b)\big)$$

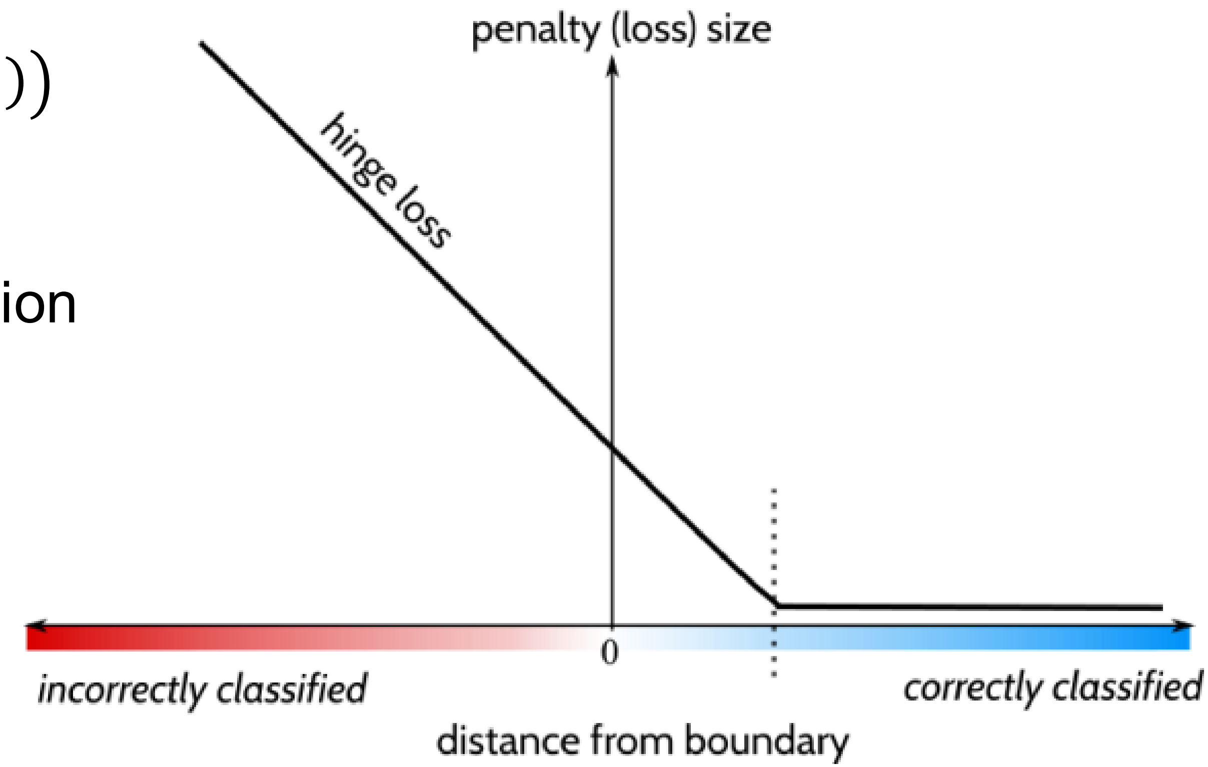
- The hinge loss is defined as follows:

$$\mathrm{L}(x_i) = \max\big(0, 1 - y_i(\langle w, x_i\rangle + b)\big)$$

- We can rewrite the minimization

problem as follows:

$$min\left(C\sum_i \mathrm{L}(x_i) + \|w\|^2\right)$$



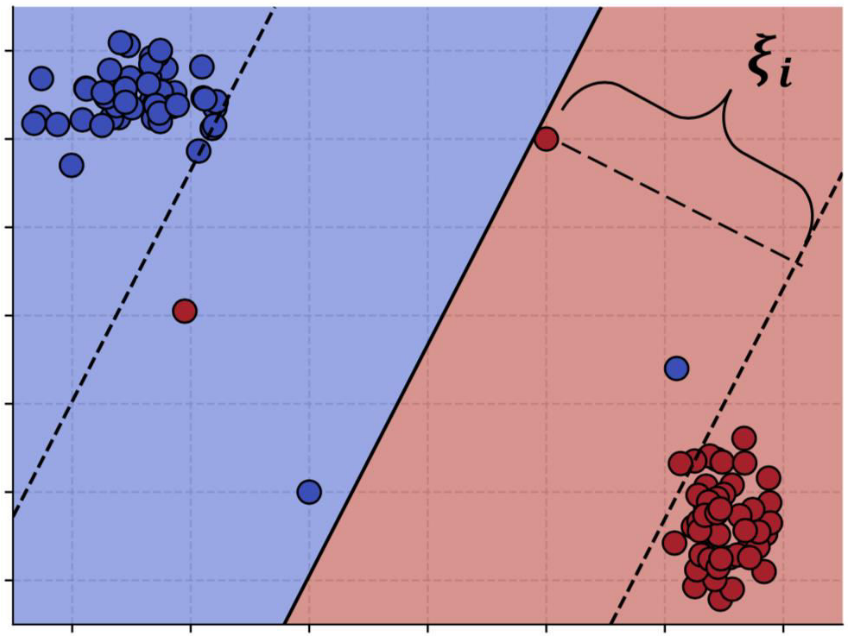
- Note: The soft-margin SVM is actually minimizing the hinge loss with regularization

SVM (Soft Margin)

- Allow some training samples to be misclassified, at a cost.
- We introduce slack variables $\xi_i \geq 0$ (how much example x_i is allowed to cross the border):

$$y_i(\langle w, x_i \rangle + b) - 1 \geq 0 \text{ becomes}$$

$$y_i(\langle w, x_i \rangle + b) - 1 \geq -\xi_i$$



- The optimization problem becomes:

$$min\|w\|^2 + C\sum_i \xi_i$$

$$subject\ to\ y_i(\langle w, x_i \rangle + b) - 1 \geq -\xi_i$$

- Trade-off between making the margin wide and allowing training mistakes
- C controls the weight of the mistakes

SVM (Soft Margin)

- The decision rule (in primal form) is:

$$x \text{ is positive if } \langle w, x \rangle + b \geq 0$$

- In order to obtain w and b, we have to:

$$min\|w\|^2 + C\sum_i \xi_i$$

$$subject\ to\ y_i(\langle w, x_i \rangle + b) - 1 \geq -\xi_i$$

Or

$$min\|w\|^2 + C\sum_i \max\big(0, 1 - y_i(\langle w, x_i \rangle + b)\big)$$

SVM Dual (Soft Margin)

- The decision rule (in dual form) is:

  x is positive if $\sum_i \alpha_i y_i \langle x_i, x \rangle + b \geq 0$

- In order to obtain α_i and b, we have to:

$$min \sum_i \alpha_i - \sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

$$subject\ to\ 0 \leq \alpha_i \leq C, \forall i$$

- Note: we add an upper bound on the Lagrange multipliers

SVM (Python)

- Scikit-learn:

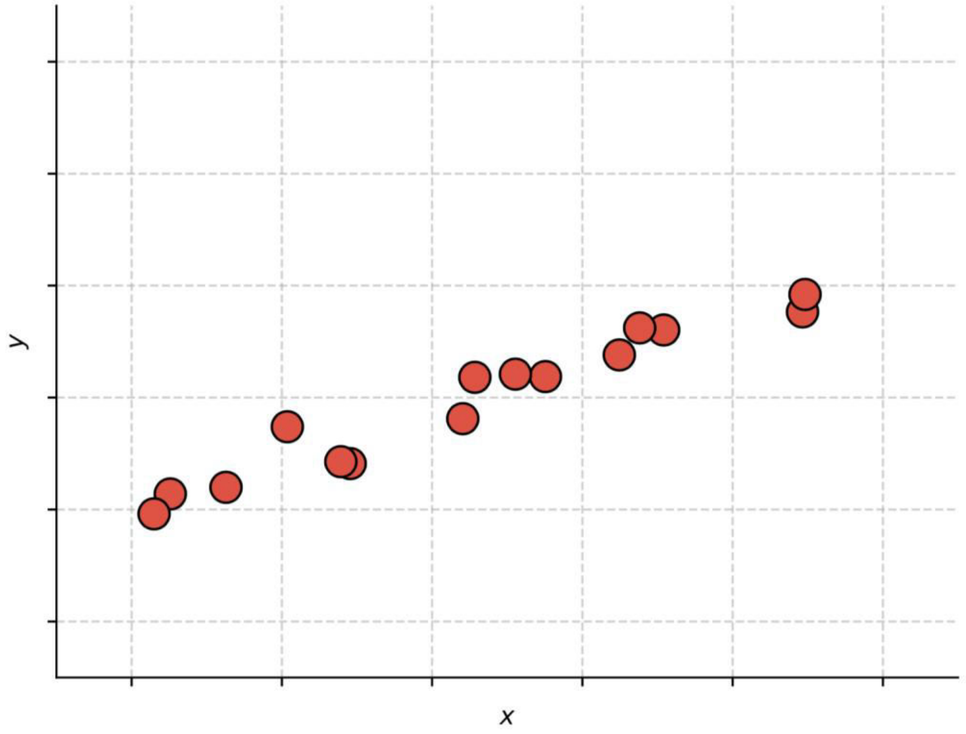
https://scikit-learn.org/stable/modules/svm.html#svm-classification

```
from sklearn import svm

clf = svm.SVC(C = 1.0)

clf.fit(X_train, T_train)

Y_test = clf.predict(X_test)
```

- Plus many other classifiers (based on same use pattern)

Support Vector Regression

- Training: find the flattest function that can fit the training data inside a tube of radius ϵ
- Inference: prediction is based on: $\hat{y} = \langle w, x \rangle + b$



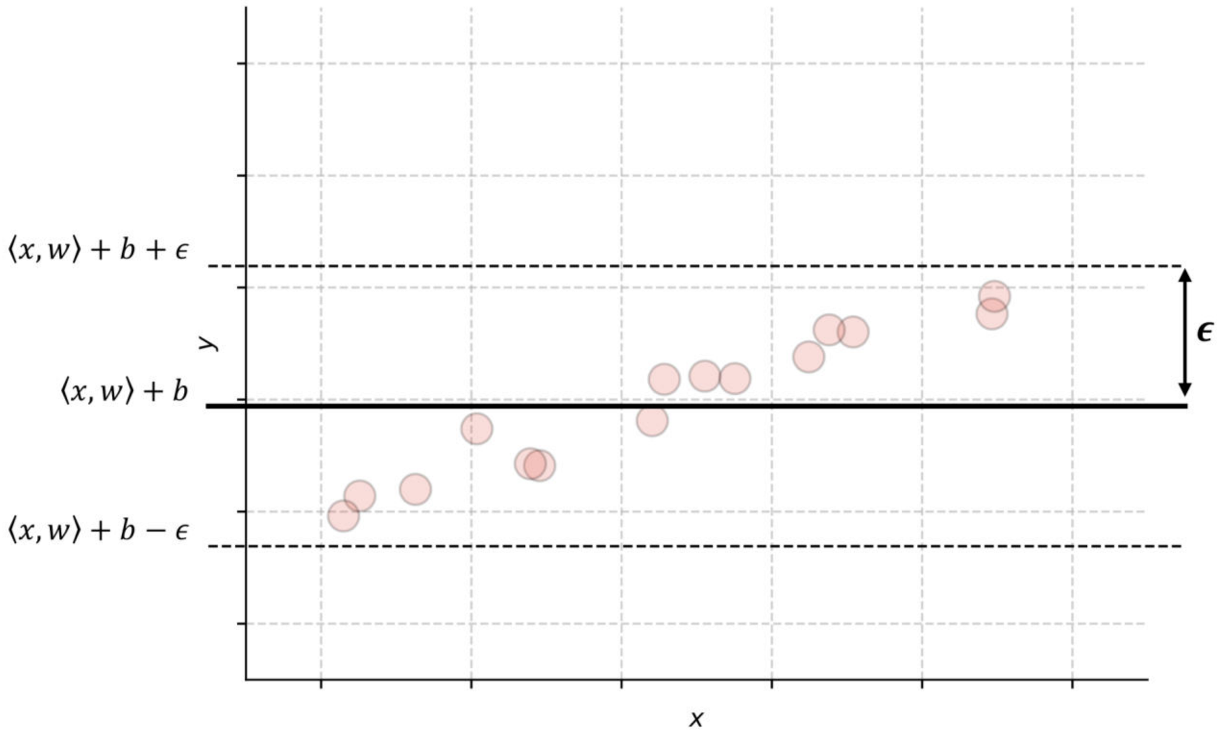
Support Vector Regression

- Training: find the flattest function that can fit the training data inside a tube of radius ϵ
- Inference: prediction is based on: $\hat{y} = \langle w, x \rangle + b$

- Flat function = does not change much with parameters, i.e. small $\|w\|$
- We need to find the smallest $\|w\|$ for which the prediction error is at most ϵ:

$$min\|w\|^2$$

$$subject\ to\ |y_i - (\langle w, x_i \rangle + b)| \leq \epsilon$$



Extreme case: we can fit all data points with a function that does not change at all, i.e. $\|w\| = 0$.

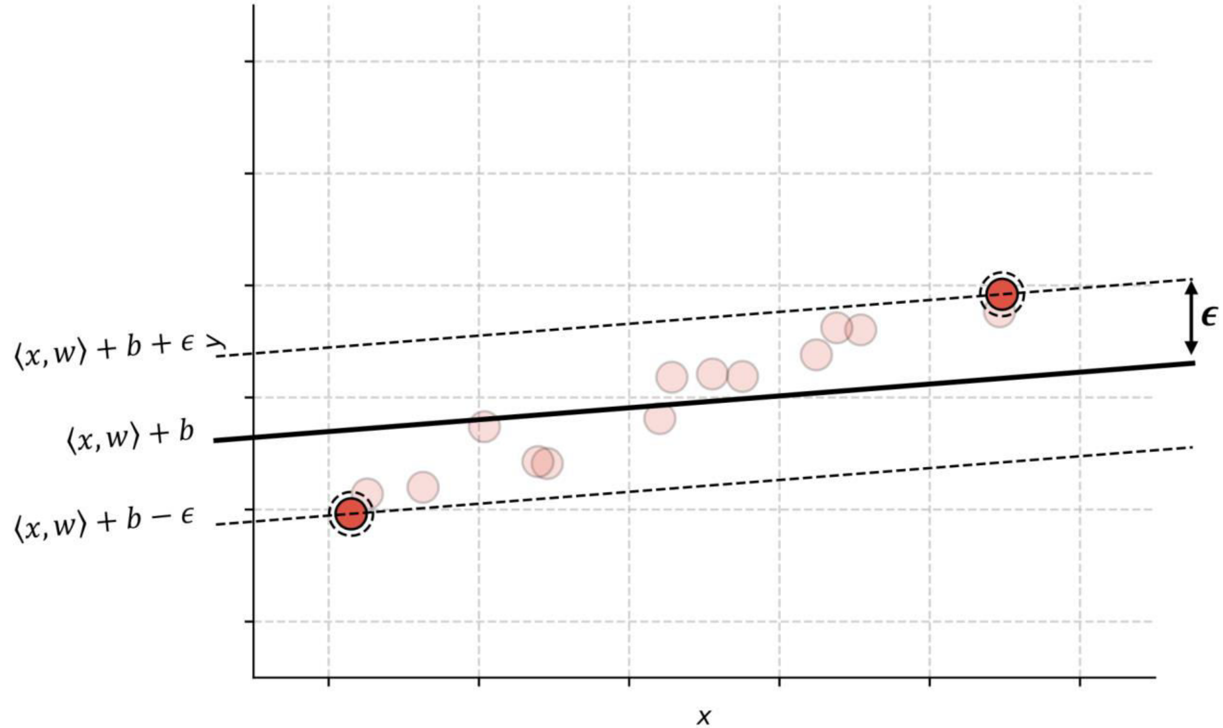
Support Vector Regression

- Training: find the flattest function that can fit the training data inside a tube of radius ϵ
- Inference: prediction is based on: $\hat{y} = \langle w, x \rangle + b$

- Flat function = does not change much with parameters, i.e. small $\|w\|$
- We need to find the smallest $\|w\|$ for which the prediction error is at most ϵ:

$$min\|w\|^2$$

$$subject\ to\ |y_i - (\langle w, x_i \rangle + b)| \leq \epsilon$$



When ϵ is small, w and b need to be adjusted in order to fit all data points.

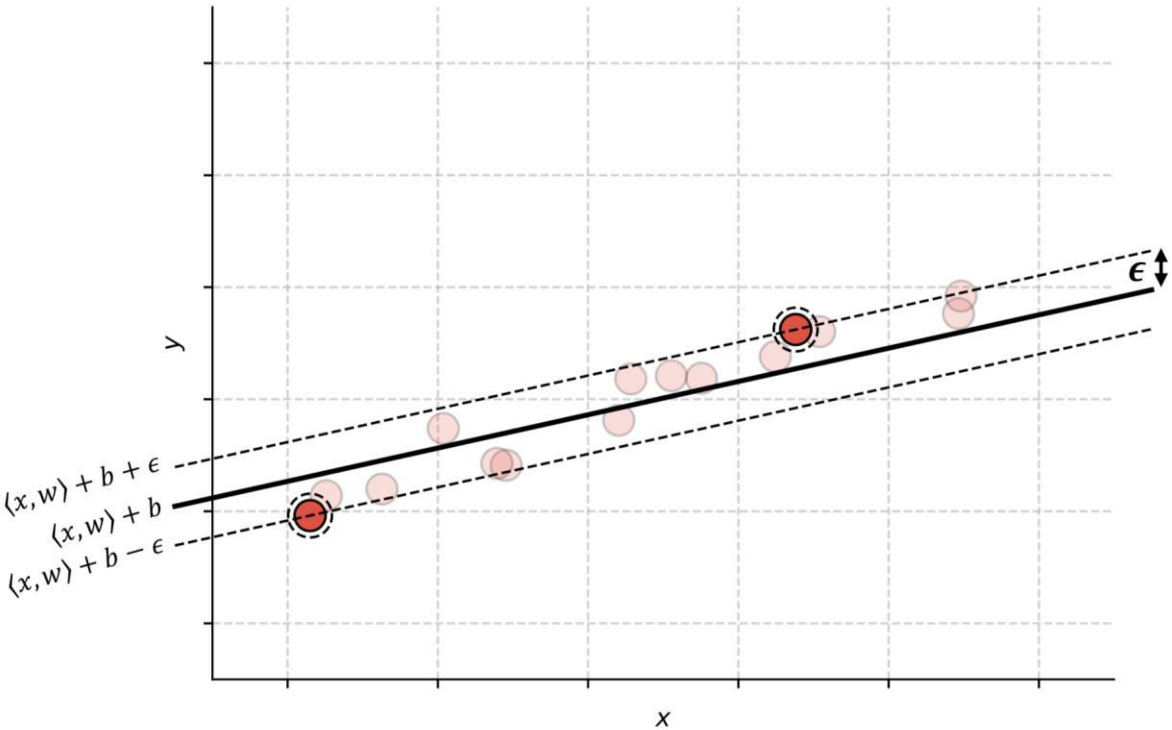
Support Vector Regression

- Training: find the flattest function that can fit the training data inside a tube of radius ϵ
- Inference: prediction is based on: $\hat{y} = \langle w, x \rangle + b$

- Flat function = does not change much with parameters, i.e. small $\|w\|$
- We need to find the smallest $\|w\|$ for which the prediction error is at most ϵ:

$$min\|w\|^2$$

$$subject\ to\ |y_i - (\langle w, x_i \rangle + b)| \leq \epsilon$$



When ϵ is small, w and b need to be adjusted in order to fit all data points.

Support Vector Regression

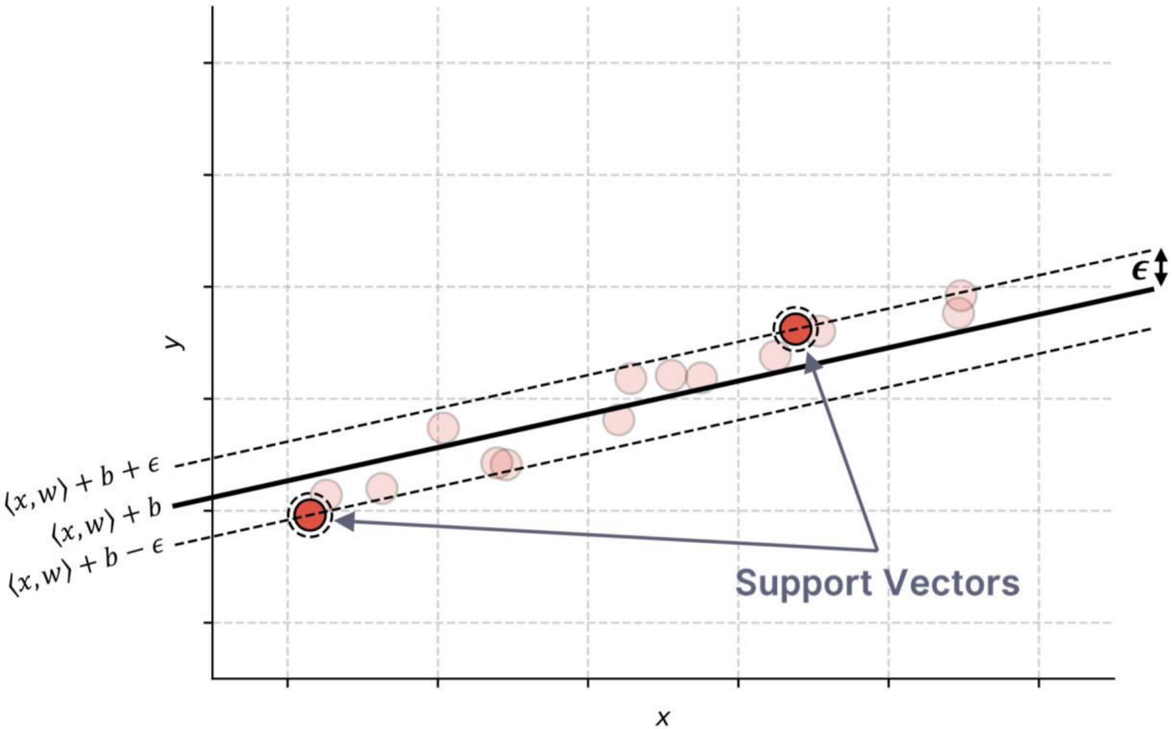
- Training: find the flattest function that can fit the training data inside a tube of radius ϵ
- Inference: prediction is based on: $\hat{y} = \langle w, x \rangle + b$

- Flat function = does not change much with parameters, i.e. small $||w||$
- We need to find the smallest $||w||$ for which the prediction error is at most ϵ:

$$min||w||^2$$

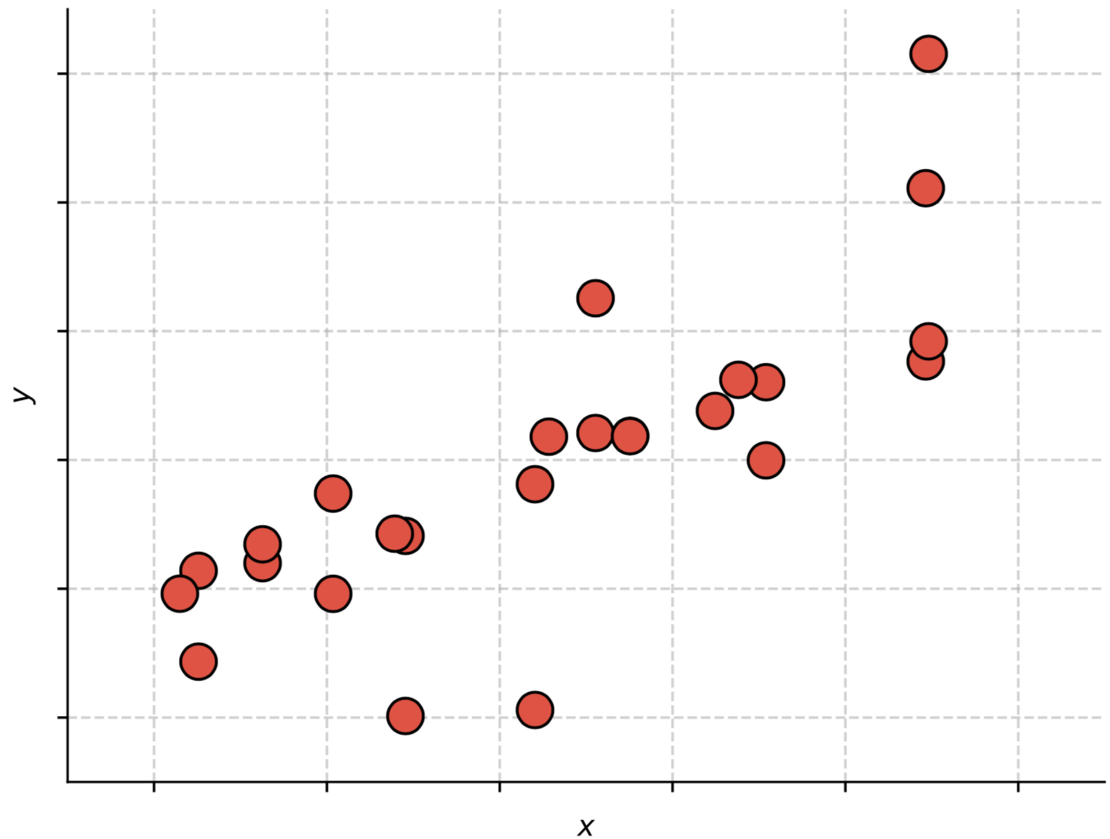
$$subject\ to\ |y_i - (\langle w, x_i \rangle + b)| \leq \epsilon$$

- The model can be expressed only in terms of the scalar product of some training samples (support vectors), i.e. we can apply kernel functions



SVR (Soft Margin)

- Training: find the flattest function that can fit the training data inside a tube of radius ϵ
- Inference: prediction is based on: $\hat{y} = \langle w, x \rangle + b$
- Sometimes it is not reasonable to find a function that fits all training samples inside a tube of radius ϵ



SVR (Soft Margin)

- Training: find the flattest function that can fit the training data inside a tube of radius ϵ
- Inference: prediction is based on: $\hat{y} = \langle w, x \rangle + b$

- Sometimes it is not reasonable to find a function that fits all training samples inside a tube of radius ϵ
- We introduce slack variables:

$$min\|w\|^2 + C\sum_i \xi_i$$

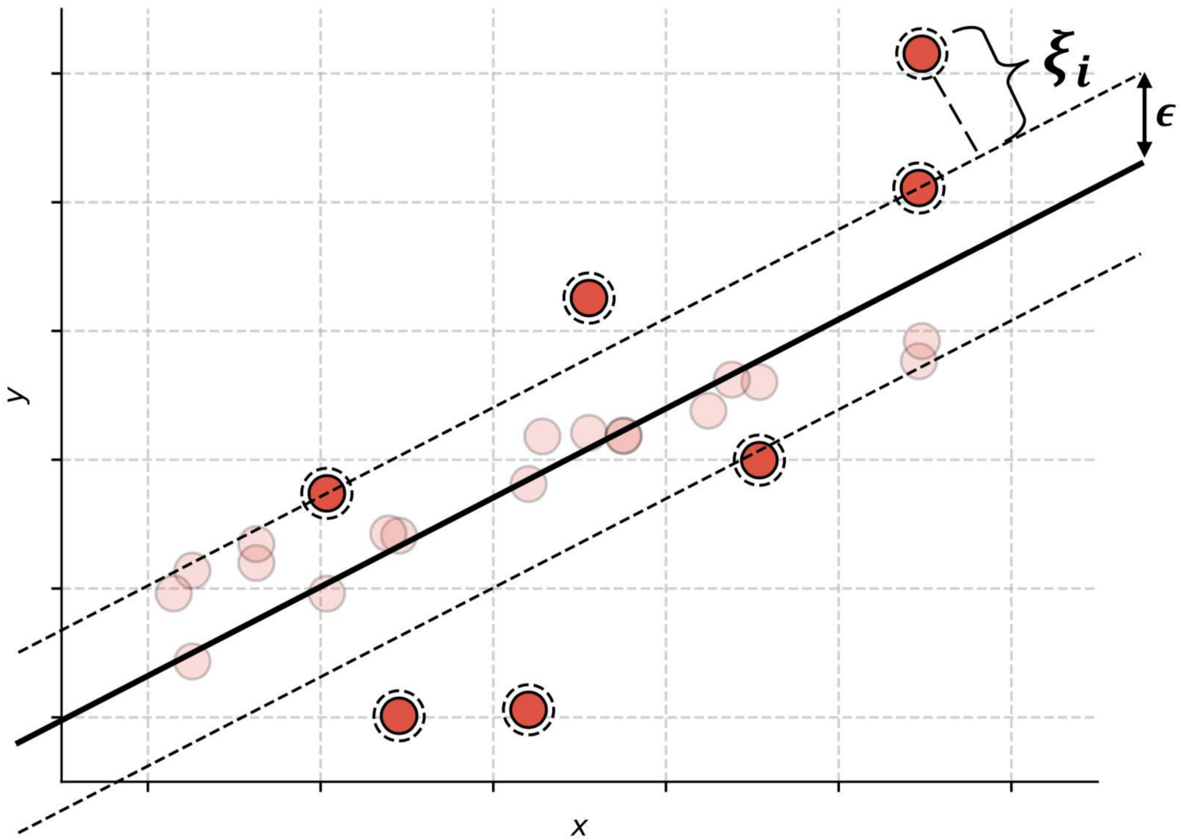
$$subject\ to\ |y_i - (\langle w, x_i \rangle + b)| \leq \epsilon + \xi_i$$

- We define the ϵ-insensitive loss:

$$\mathrm{L}(x_i) = \max(0, |y_i - (\langle w, x_i \rangle + b)| - \epsilon)$$

- And the optimization becomes:

$$min\left(C\sum_i \mathrm{L}(x_i) + \|w\|^2\right)$$

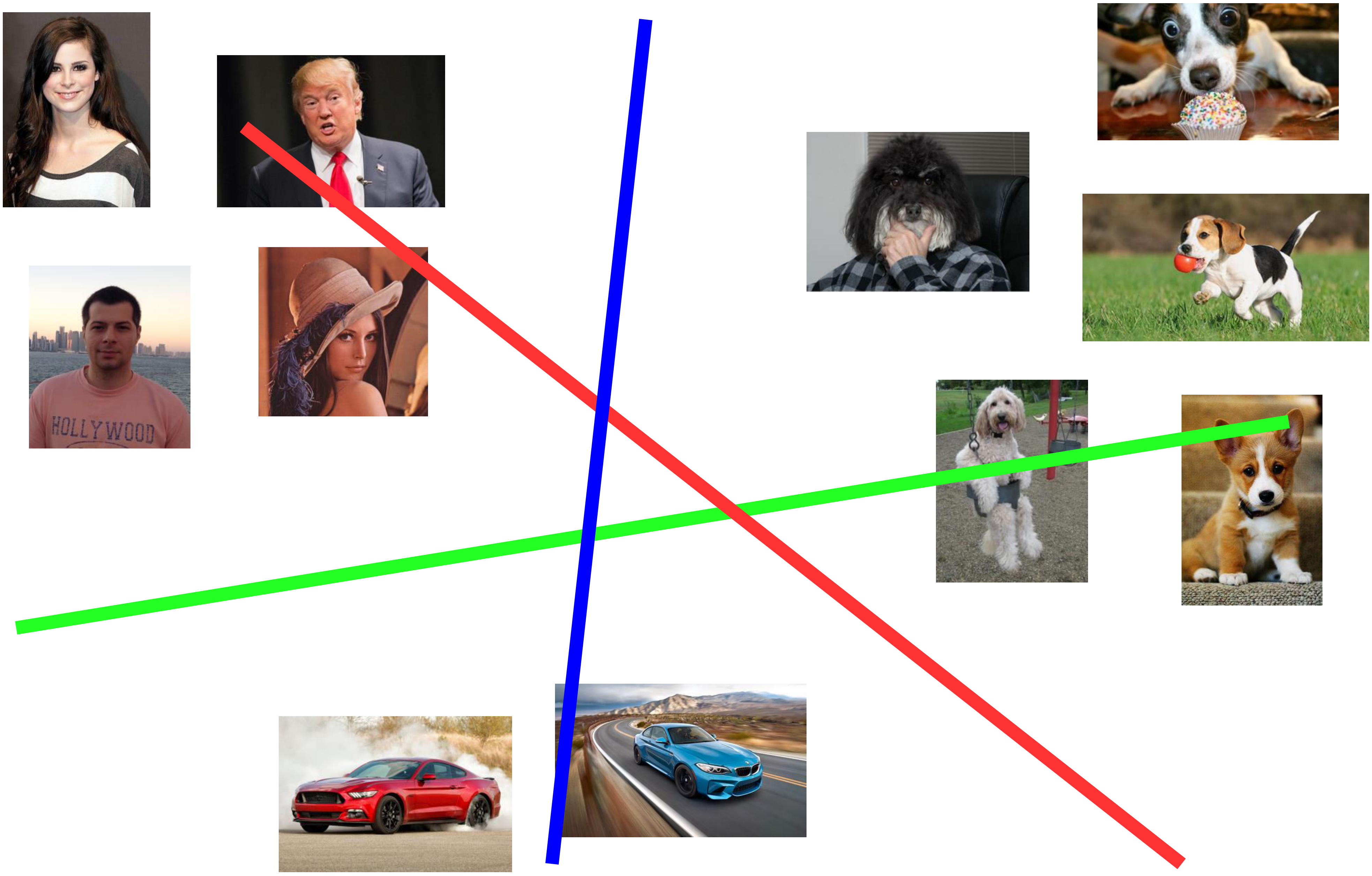


How do we solve many class problems?

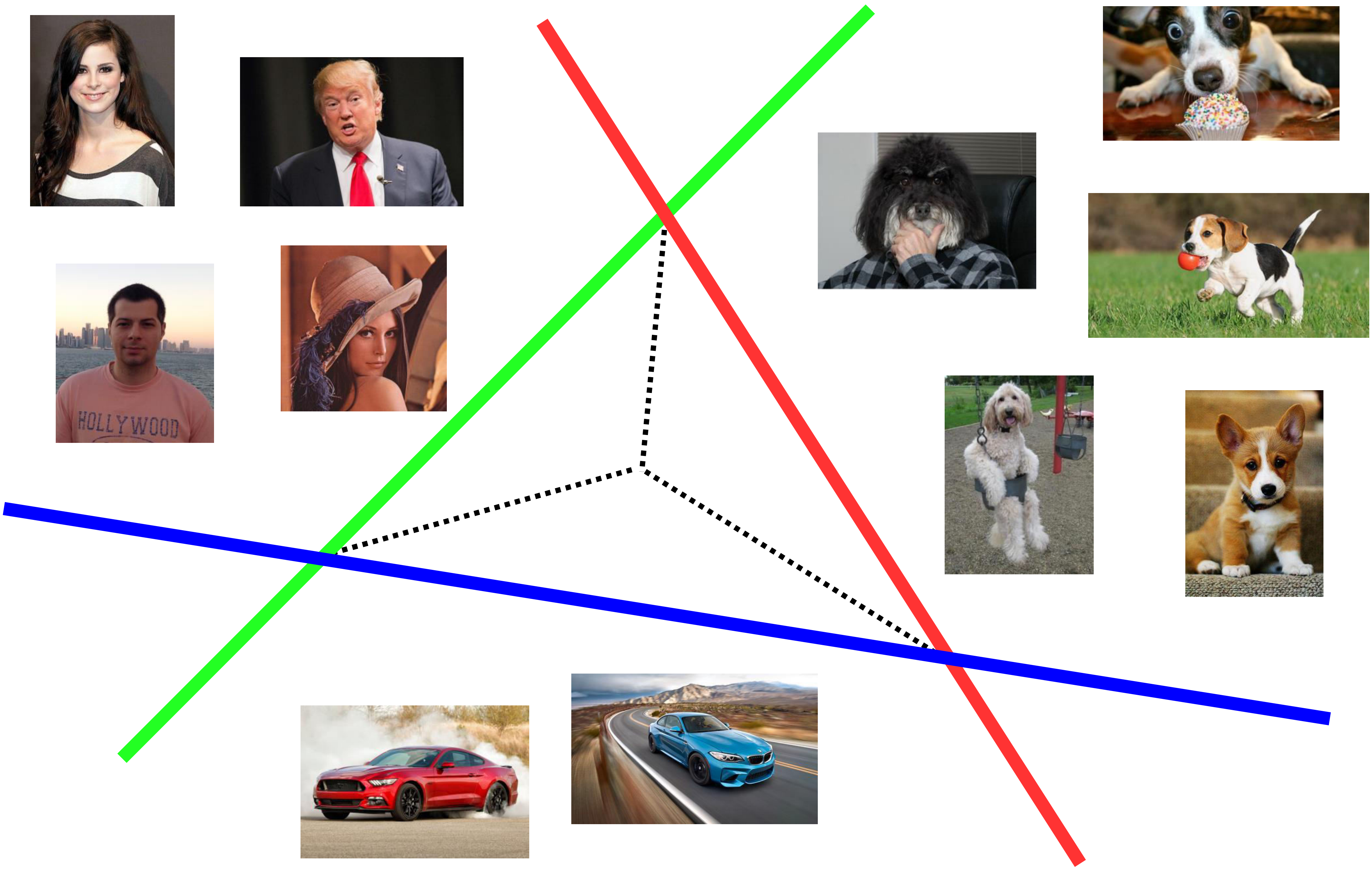
- Schemes for combining multiple binary classifiers:

1) One-versus-one
2) One-versus-all

One-versus-one



One-versus-all



How do we solve many class problems?

- Using classification methods able to handle the multi-class problem directly:

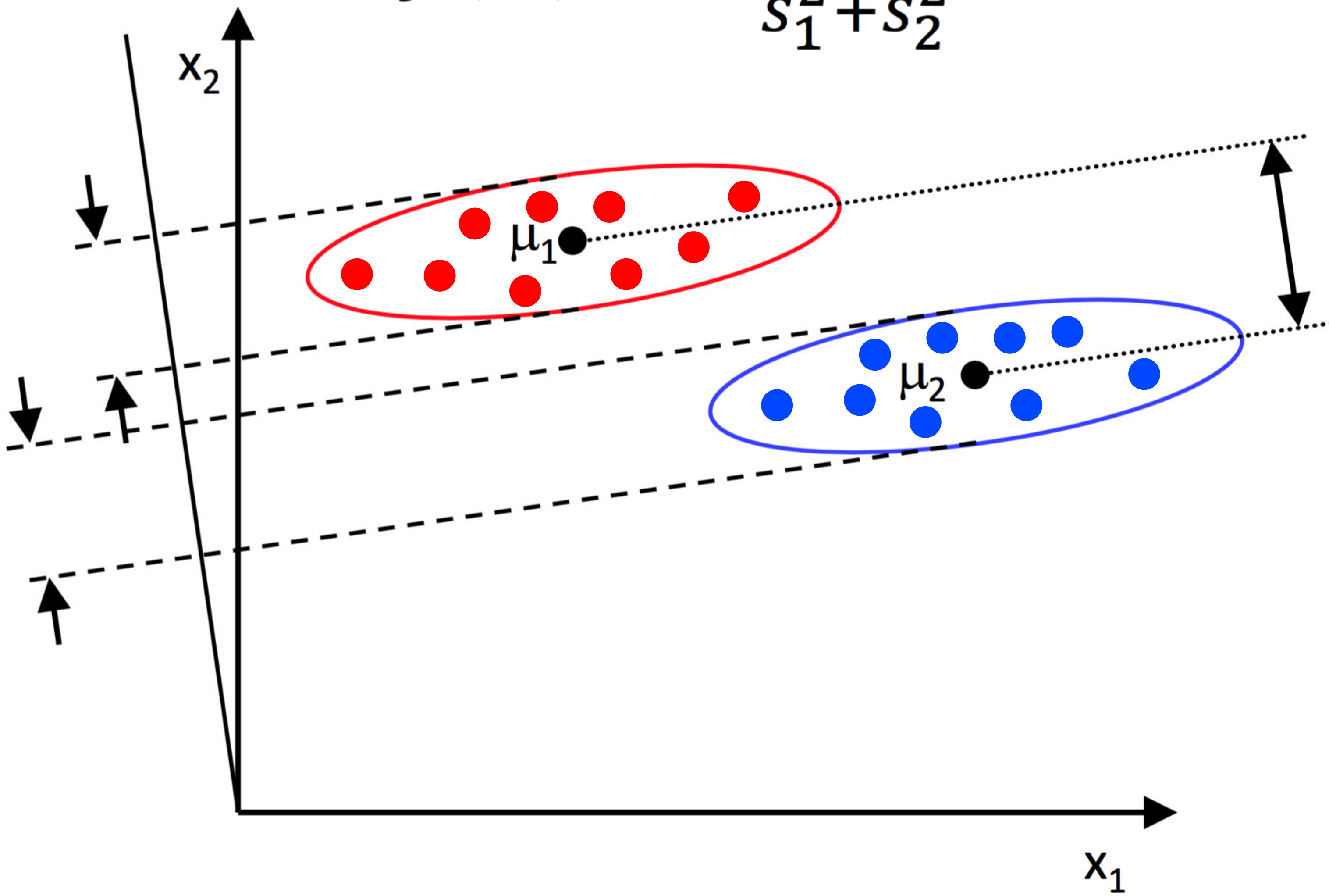
1) Linear (Fisher) Discriminant Analysis
2) Neural Networks (Lectures 7, 8, 9)
3) Random forests (Lecture 5)

Linear Discriminant Analysis

- Each class is approximated with a multinomial Gaussian distribution
- The optimization algoritm is based on finding a hyperplane to project the data points such that:
  - The distance among class means is maximized
  - The variance of each class is minimized

Linear Discriminant Analysis

$$J(w) = \frac{|\mu_1 - \mu_2|^2}{s_1^2 + s_2^2}$$



Linear Discriminant Analysis (Python)

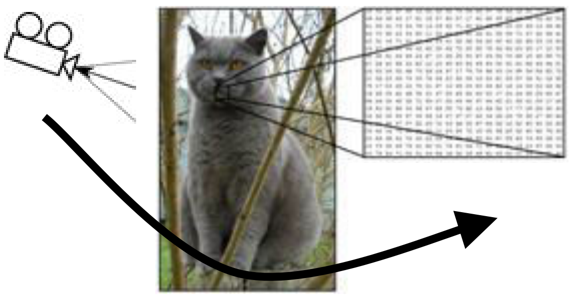
- Scikit-learn:

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

```
from sklearn.discriminant_analysis
     import LinearDiscriminantAnalysis
clf = LinearDiscriminantAnalysis()
clf.fit(X_train, T_train)
Y_test = clf.predict(X_test)
```

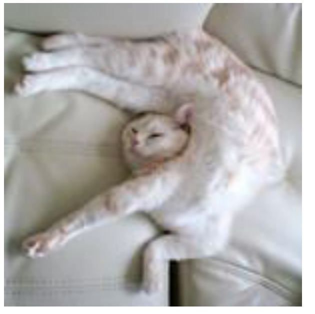
Intra-class variation

Camera angle

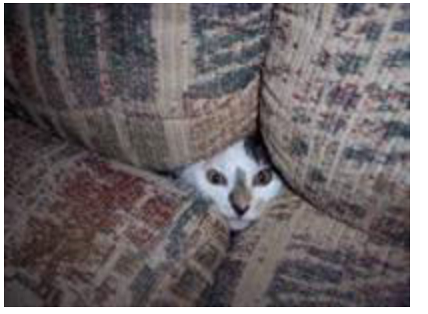


Illumination

Deformation



Occlusion



Confusing background

Intra-class variation

Inter-class similarity

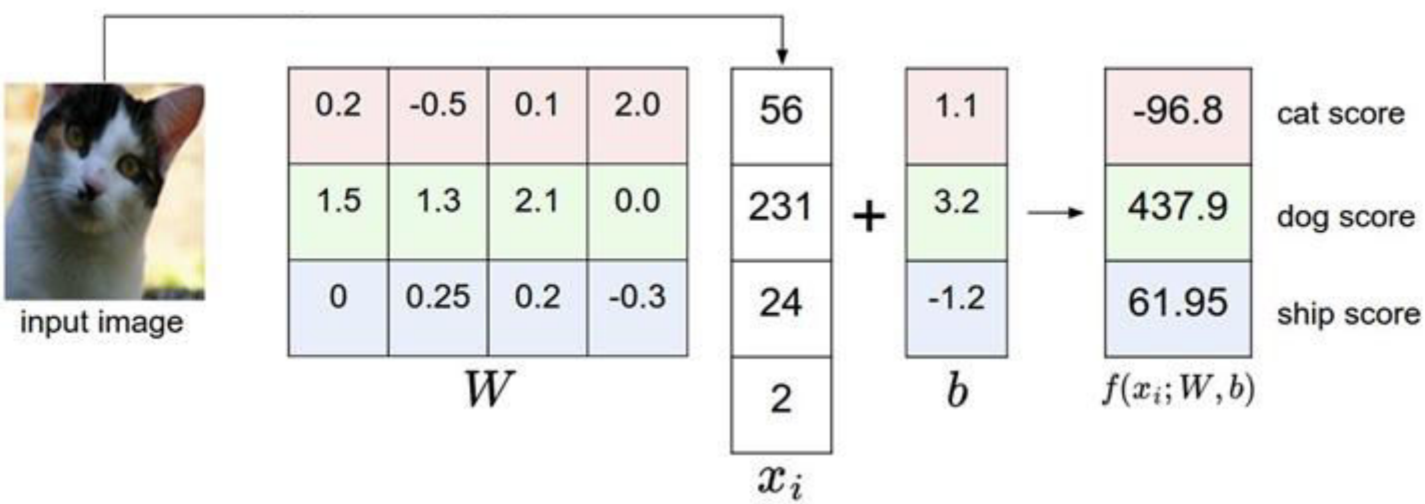
Linear classifier for the multi-class problem

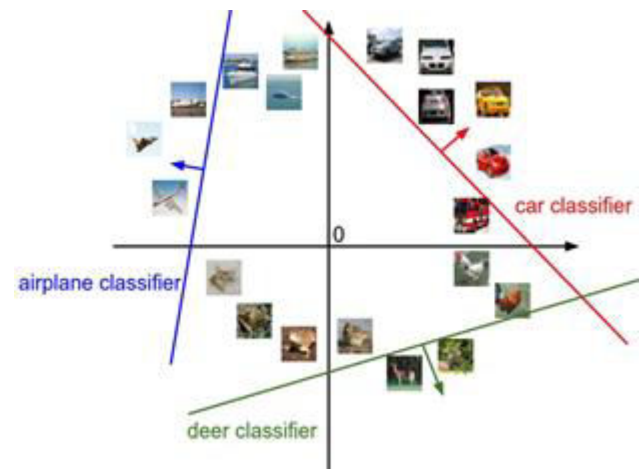
features parameters

$$f(\mathbf{x}, \mathbf{W})$$

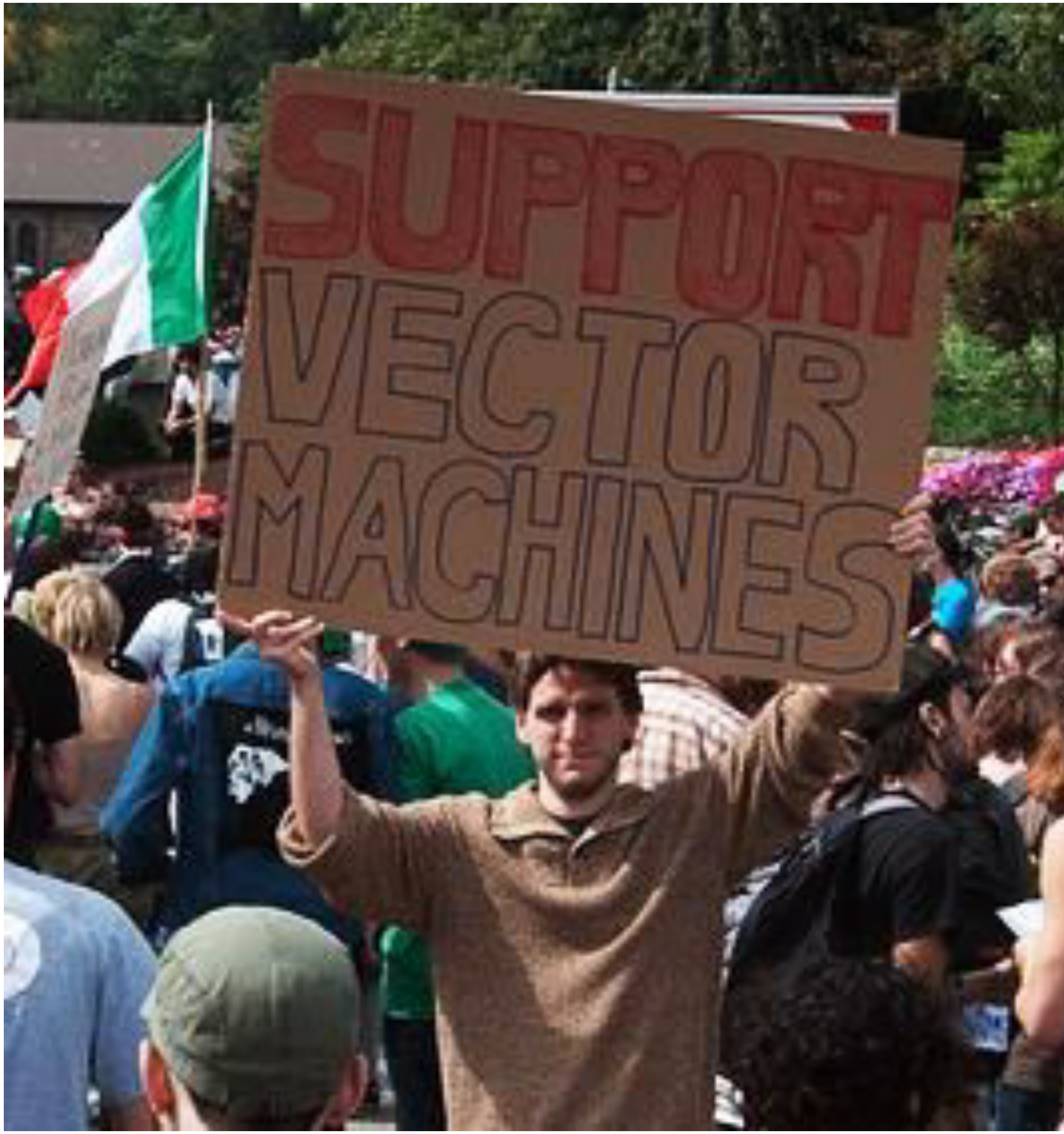
N values that indicate the scores for each class

Feature vector





Linear classifier for the multi-class problem



Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

Multiclass SVM loss:

Given an example (x_i, y_i), where x_i is the image and where y_i is the (integer) label,

and using the shorthand for the scores vector:

$$s = f(x_i, W)$$

the SVM loss has the form:

$$\boxed{L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)}$$

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9		

Multiclass SVM loss:

Given an example (x_i, y_i), where x_i is the image and where y_i is the (integer) label,

and using the shorthand for the scores vector:

$$s = f(x_i, W)$$

the SVM loss has the form:

$$\boxed{L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)}$$

$$\begin{aligned} &= \max(0, 5.1 - 3.2 + 1) \\ &\quad + \max(0, -1.7 - 3.2 + 1) \\ &= \max(0, 2.9) + \max(0, -3.9) \\ &= 2.9 + 0 \\ &= 2.9 \end{aligned}$$

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	

Multiclass SVM loss:

Given an example (x_i, y_i), where x_i is the image and where y_i is the (integer) label,

and using the shorthand for the scores vector:

$$s = f(x_i, W)$$

the SVM loss has the form:

$$\boxed{L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)}$$

$= \max(0, 1.3 - 4.9 + 1)$
$+\max(0, 2.0 - 4.9 + 1)$
$= \max(0, -2.6) + \max(0, -1.9)$
$= 0 + 0$
$= 0$

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	10.9

Multiclass SVM loss:

Given an example (x_i, y_i), where x_i is the image and where y_i is the (integer) label,

and using the shorthand for the scores vector:

$$s = f(x_i, W)$$

the SVM loss has the form:

$$\boxed{L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)}$$

= max(0, 2.2 - (-3.1) + 1)
+max(0, 2.5 - (-3.1) + 1)
= max(0, 5.3) + max(0, 5.6)
= 5.3 + 5.6
= 10.9

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	10.9

Multiclass SVM loss:

Given an example (x_i, y_i), where x_i is the image and where y_i is the (integer) label,

and using the shorthand for the scores vector:

$$s = f(x_i, W)$$

the SVM loss has the form:

$$\boxed{L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)}$$

L = (2.9 + 0 + 10.9)/3
= **4.6**

Softmax (Multinomial Logistic Regression)

scores = unnormalized log probabilities of the classes

$$P(Y=k|X=x_i) = \boxed{\frac{e^{s_k}}{\sum_j e^{s_j}}} \quad \text{where} \quad s = f(x_i; W)$$

Softmax function

cat	**3.2**
car	5.1
frog	-1.7

Want to maximize the log likelihood, or (for a loss function) to minimize the negative log likelihood of the correct class:

$$L_i = -\log P(Y = y_i | X = x_i)$$

In summary: $L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$

Softmax (Multinomial Logistic Regression)

$$L_i = -\log(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}})$$

Q: What is the min/max possible loss L_i?

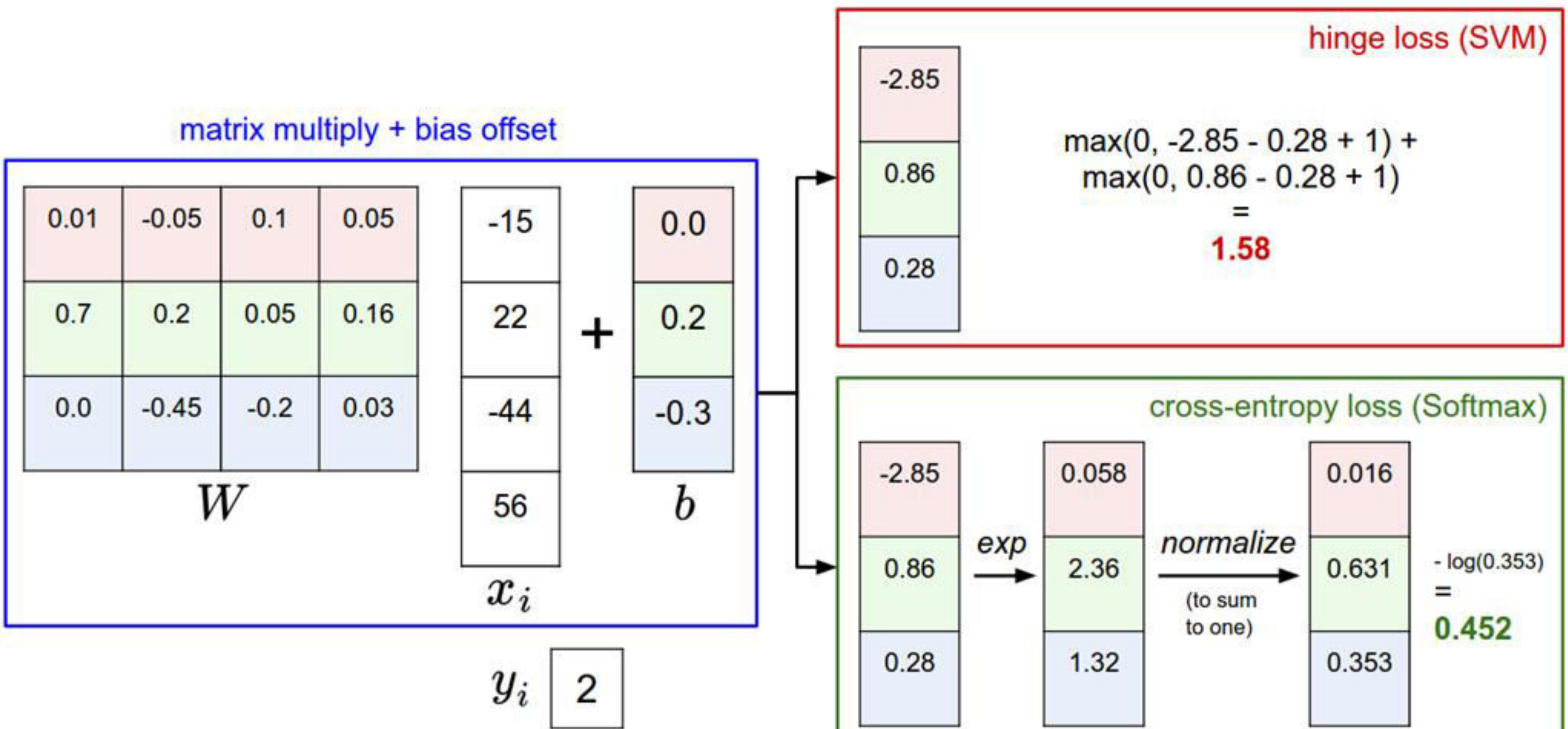
unnormalized probabilities

	unnormalized log probabilities	exp →	unnormalized probabilities	normalize →	probabilities
cat	**3.2**		**24.5**		**0.13**
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

→ $L_i = -\log(0.13)$ = **0.89**

unnormalized log probabilities

probabilities



matrix multiply + bias offset
0.01
-0.05
0.1
0.05
0.7
0.2
0.05
0.16
0.0
-0.45
-0.2
0.03
W
-15
22
-44
56
x_i
+
0.0
0.2
-0.3
b
y_i
2
hinge loss (SVM)
-2.85
0.86
0.28
max(0, -2.85 - 0.28 + 1) +
max(0, 0.86 - 0.28 + 1)
=
1.58
cross-entropy loss (Softmax)
-2.85
0.86
0.28
exp
0.058
2.36
1.32
normalize
(to sum
to one)
0.016
0.631
0.353
- log(0.353)
=
0.452

Softmax vs. SVM

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Assume scores:
[10, -2, 3]
[10, 9, 9]
[10, -100, -100]
and $\boxed{y_i = 0}$

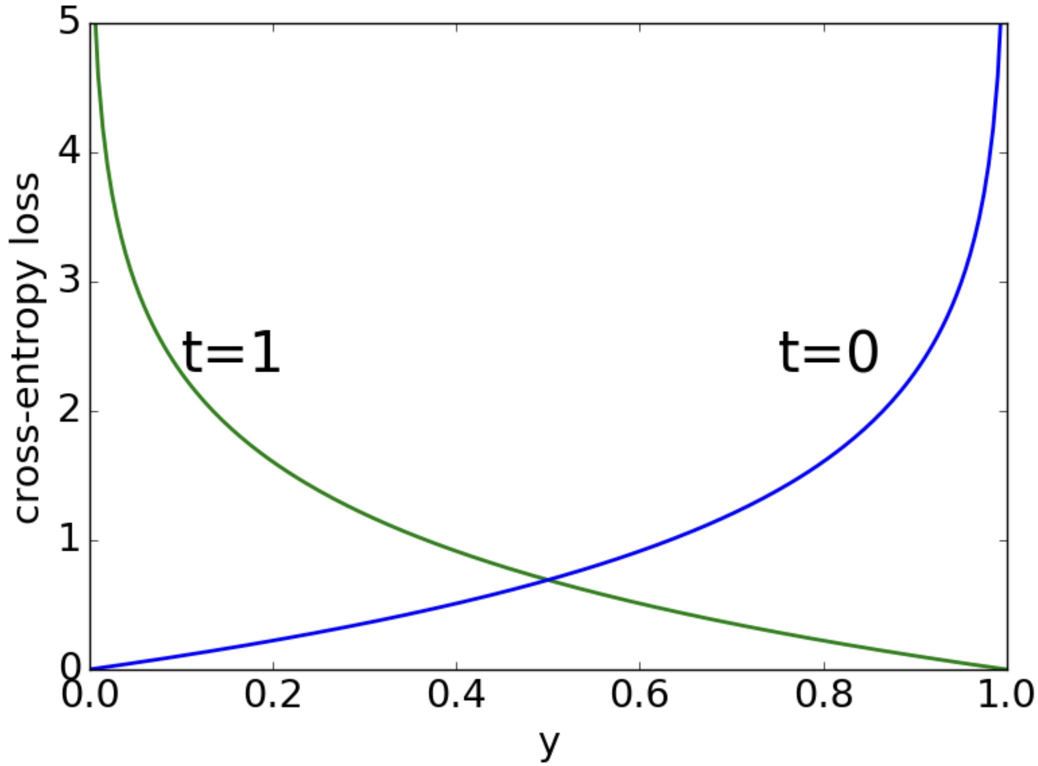
Q: Suppose we take a datapoint and we add some small perturbations (changing its score slightly). What happens to the loss in both cases?

Binary cross-entropy loss

- Binary cross-entropy (logistic) loss:

$$L_i = -(t_i \cdot log(y_i) + (1 - t_i) \cdot log(1 - y_i))$$

where t_i is the ground-truth binary label (0 or 1) of sample x_i, and y_i is the prediction for the same sample



What is the best classification method?

- **“No free lunch” theorem:**

Any two algorithms are equivalent when their (average) performance is measured on all possible tasks

- It follows that there is no shortcut in choosing the right algorithm for you task
- Usually, you have to try several and see which one works best
- Intuition and experience will be useful

Bibliography

Advances in Computer Vision and Pattern Recognition

Radu Tudor Ionescu
Marius Popescu

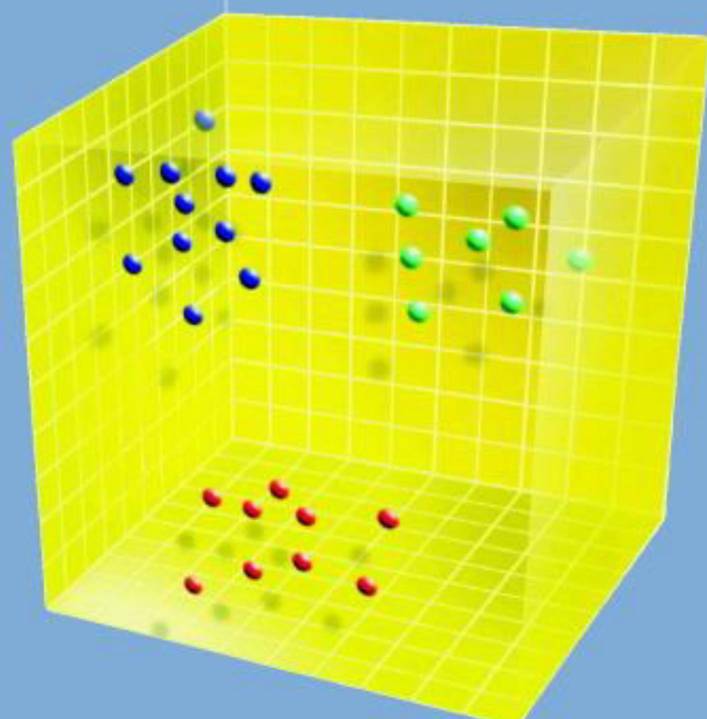
Knowledge Transfer between Computer Vision and Text Mining

Similarity-based Learning Approaches

Springer

John Shawe-Taylor
and Nello Cristianini

Kernel Methods for Pattern Analysis



CAMBRIDGE