Kernel Methods.
Ridge Regression.

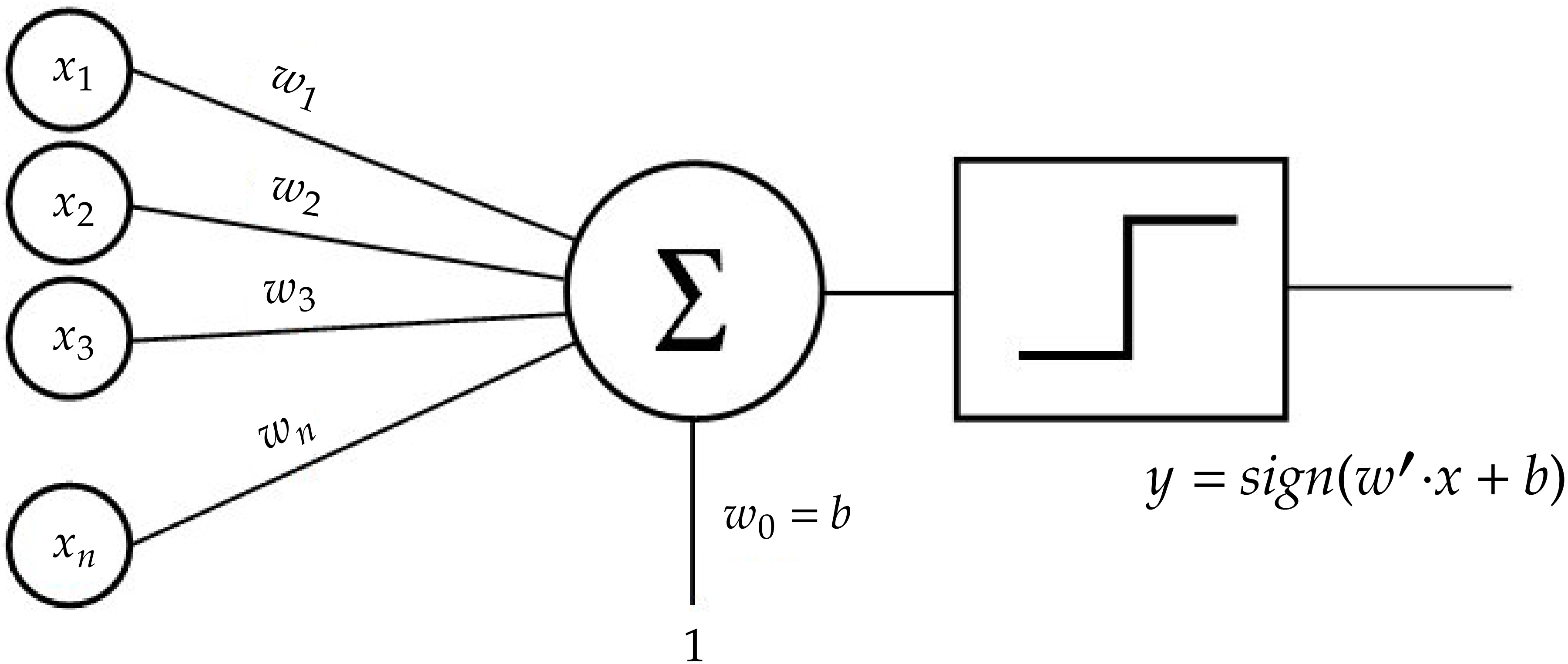
Radu Ionescu, Prof. PhD.
raducu.ionescu@gmail.com

Faculty of Mathematics and Computer Science
University of Bucharest

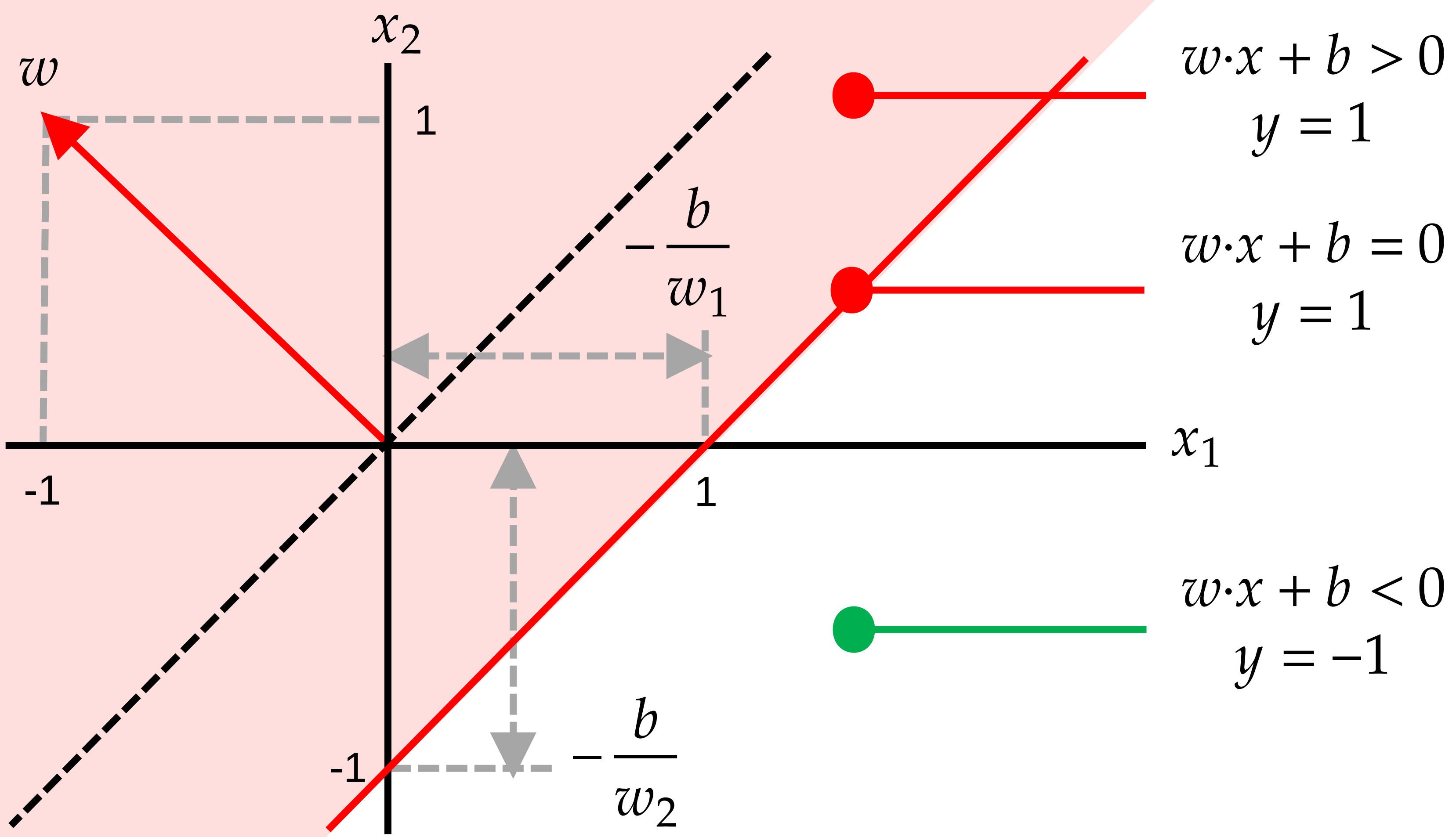
The evolution of learning methods

- 1950s: the introduction of the perceptron (Rosenblatt, 1957)
- 1980s: the back-propagation algorithm for training multi-layer perceptron becomes widely popular (Hinton, 1986)
- 1990s: the introduction of kernel methods (Cortes, 1995)

The Perceptron



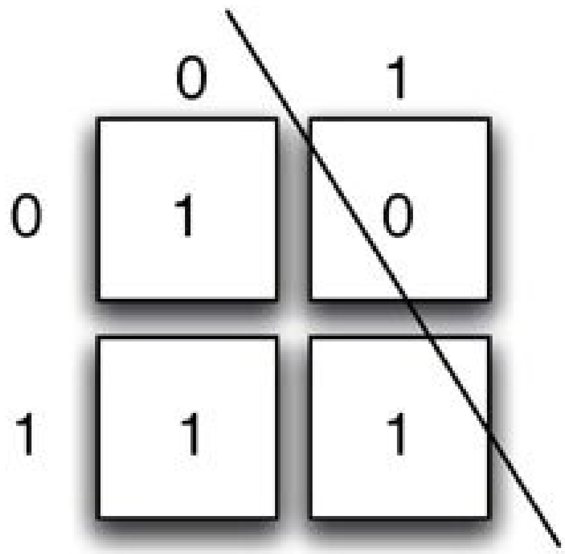
Linear separating hyperplane

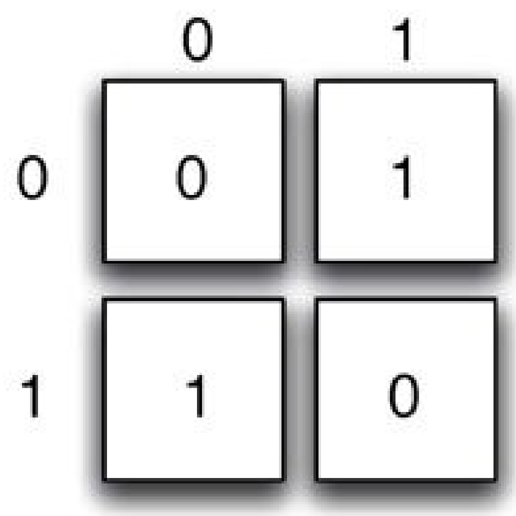


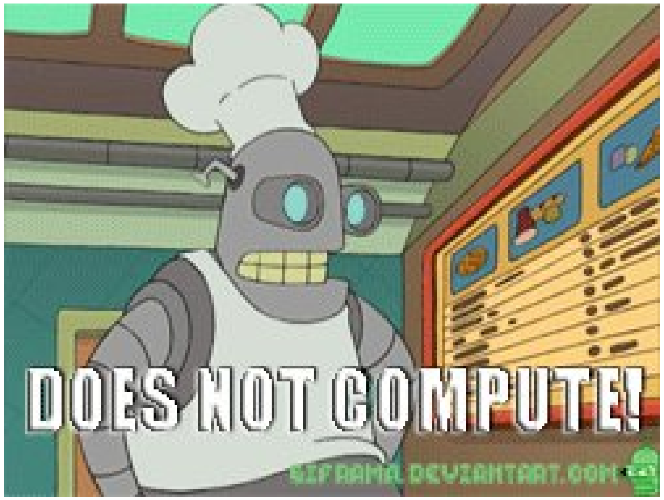
Where $w_1 = -1,\ \ w_2 = 1, b = 1$

XOR (Minsky & Papert, 1969)

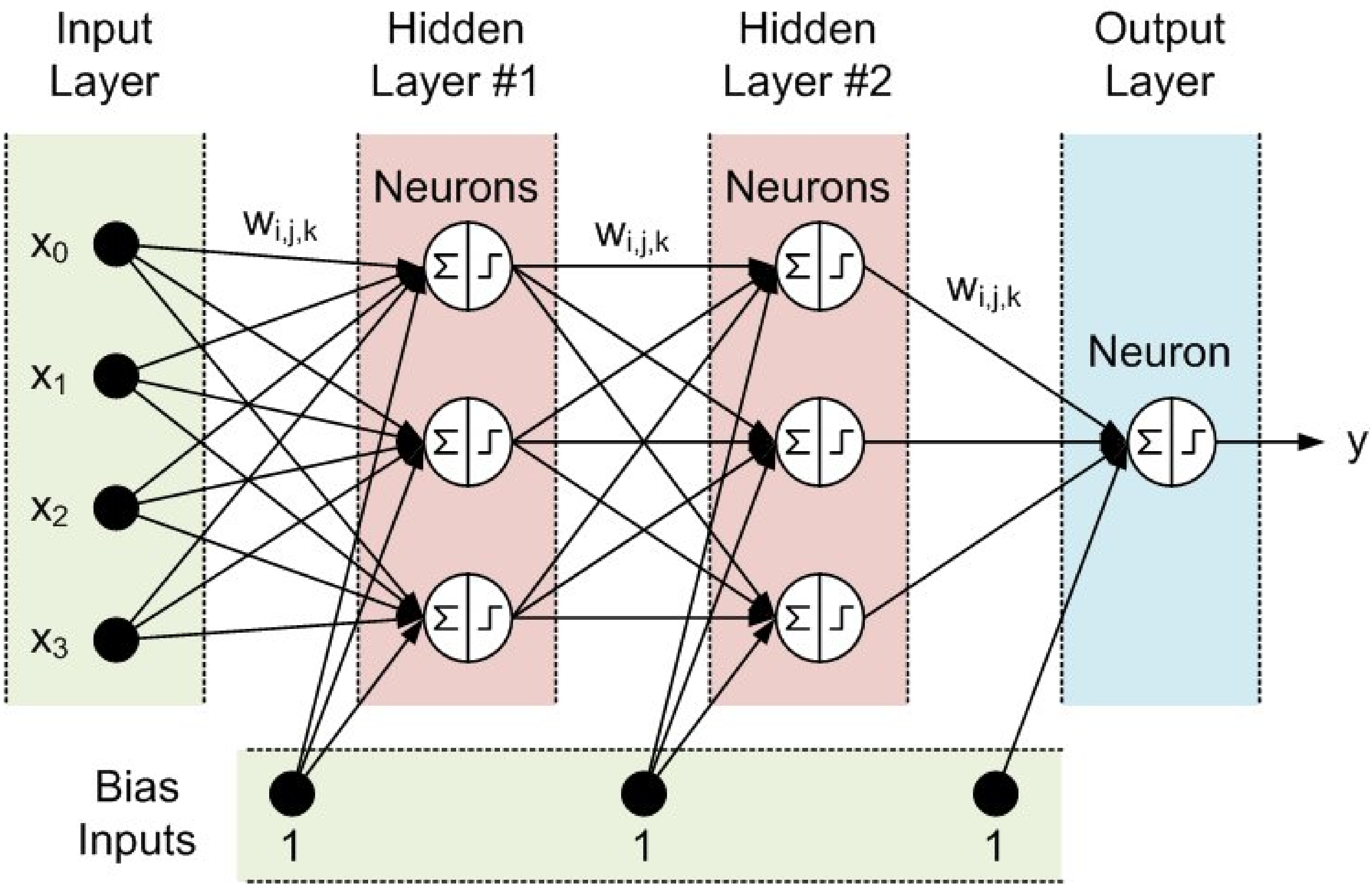
- A linear classification method cannot solve the XOR problem





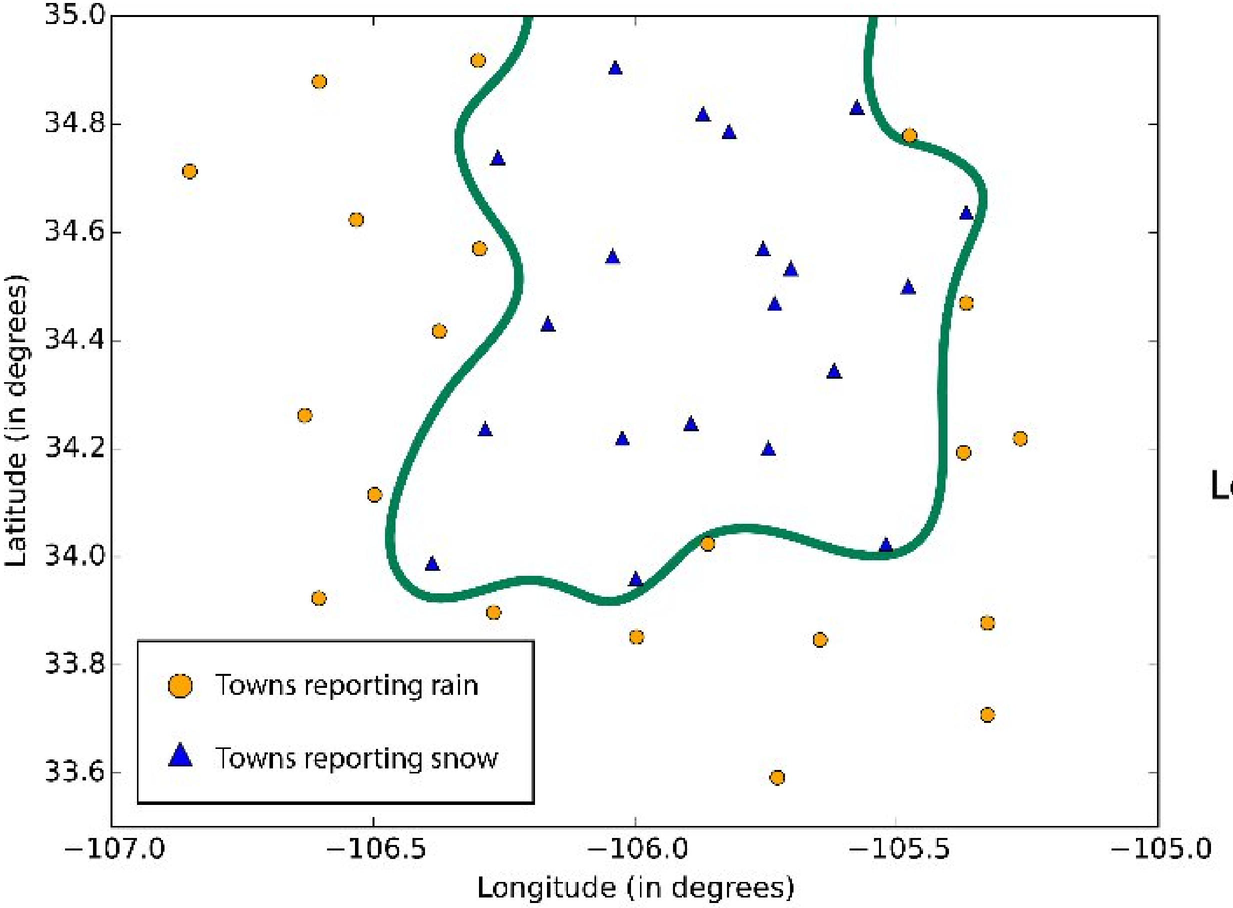


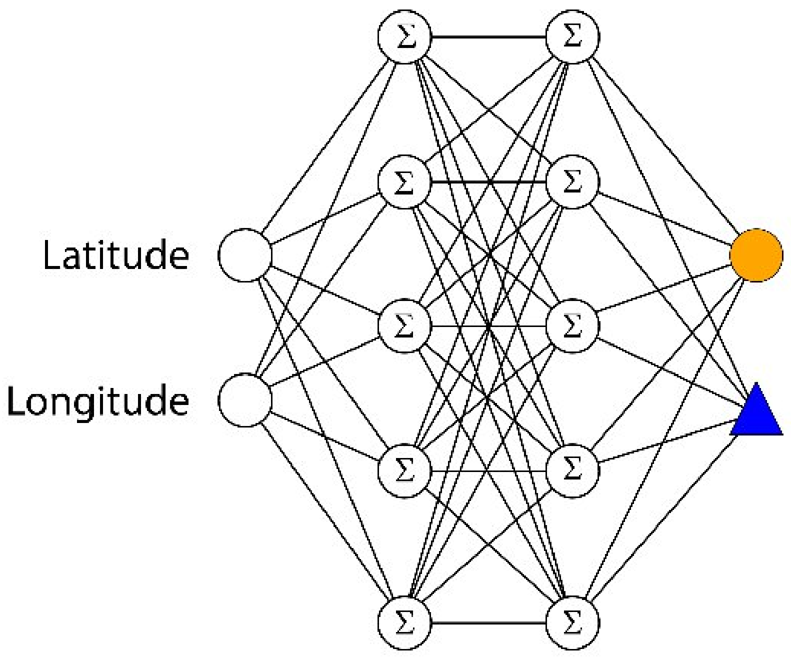
Solution 1: Neural Networks



Solution 1: Neural Networks

- The decision border is non-linear



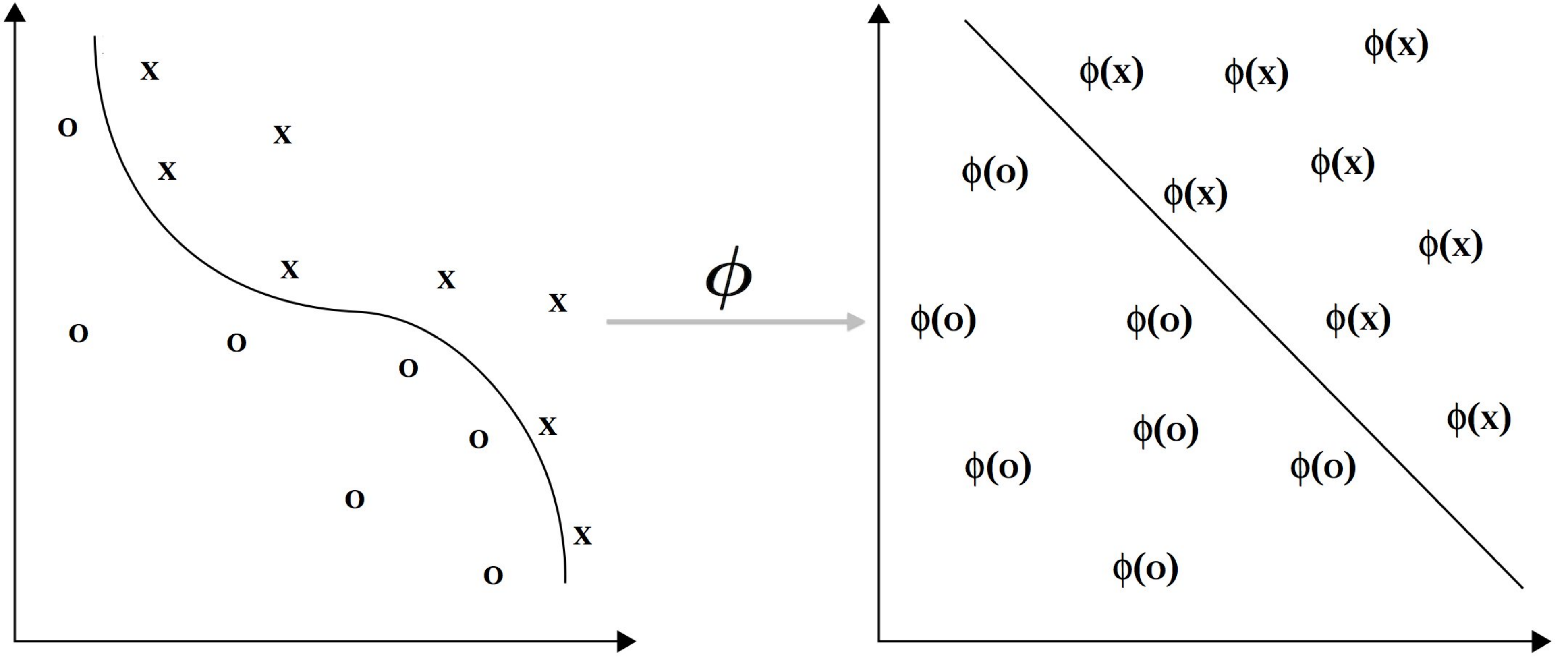


Solution 2: Kernel Methods

- 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!

Embedding the data with an embedding map

- Non-linear relations from the original space become linear in the embedding space



Kernel methods

- The learning algorithms are usually implemented such that the coordinates of the embedded points are not needed
- Specifying the scalar product between pairs of points is enough!
- “Kernel trick”: The scalar product is replaced with any pairwise similarity function, also termed kernel function

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

	x_1	x_2	x_3	x_4
x_1	55	31	16	11
x_2	31	30	14	7
x_3	16	14	35	17
x_4	11	7	17	16

$= X * X' = K_X$

l_1	1
l_2	1
l_3	-1
l_4	-1

$= L$

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

Test samples: y_1, y_2, y_3

	x_1	x_2	x_3	x_4
y_1	36	26	14	9
y_2	16	13	15	12
y_3	25	18	18	9

$= Y * X' = K_Y$

p_1	?
p_2	?
p_3	?

$= P$

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

Linear regression

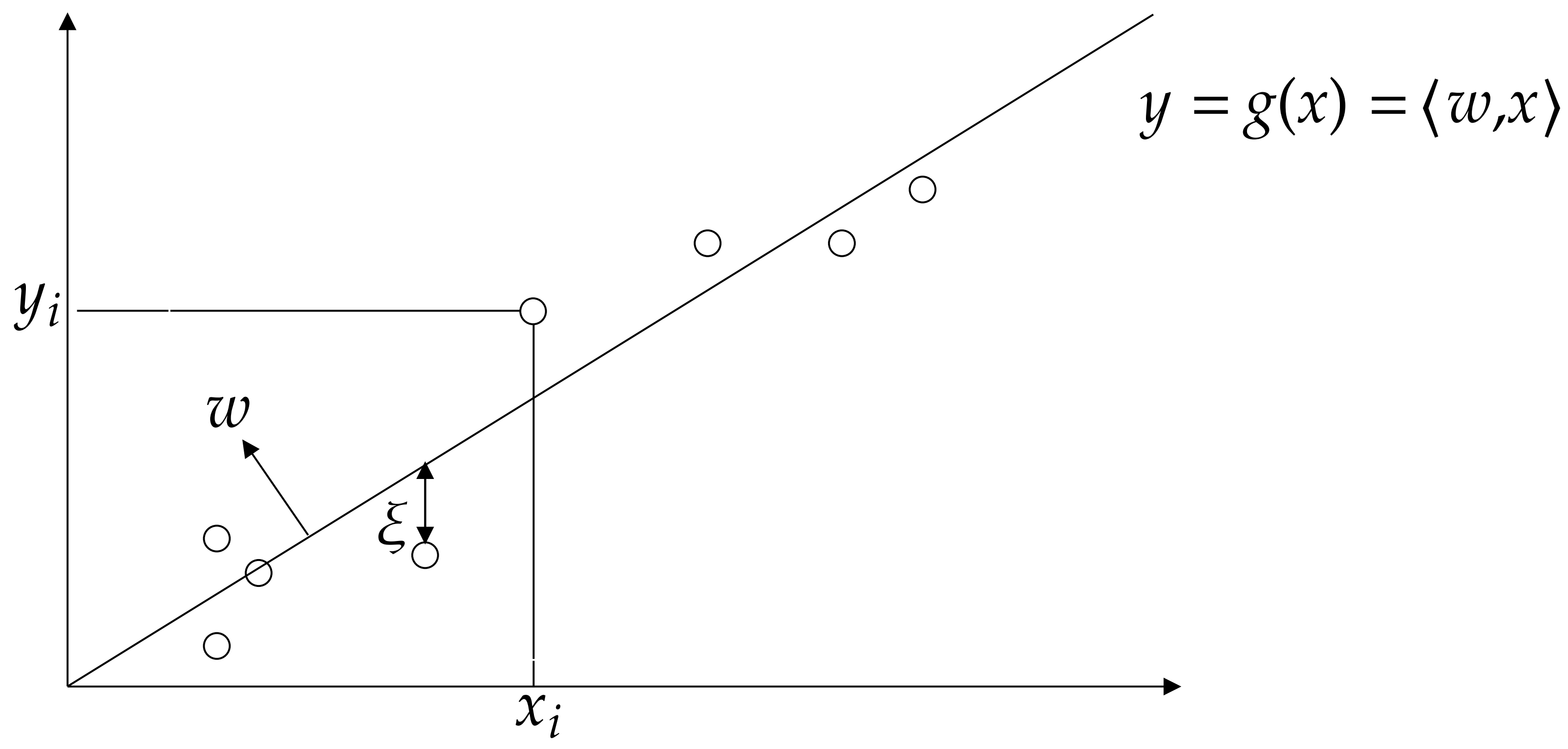
- The problem of finding *g* of the form:

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \mathbf{w}'\mathbf{x} = \sum_{i=1}^{n} w_i x_i$$

- That best interpolates a training set:

$$S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_\ell, y_\ell)\}$$

Linear regression



Linear regression

- The error of the linear function on a particular training sample is:

$$\xi = (y - g(\mathbf{x}))$$

- The loss on all training data points is:

$$\mathcal{L}(g,S) = \mathcal{L}(\mathbf{w},S) = \sum_{i=1}^{\ell} (y_i - g(\mathbf{x_i}))^2 =$$

$$= \sum_{i=1}^{\ell} \xi^2 = \sum_{i=1}^{\ell} \mathcal{L}(g,(\mathbf{x_i}, y_i))$$

Linear regression

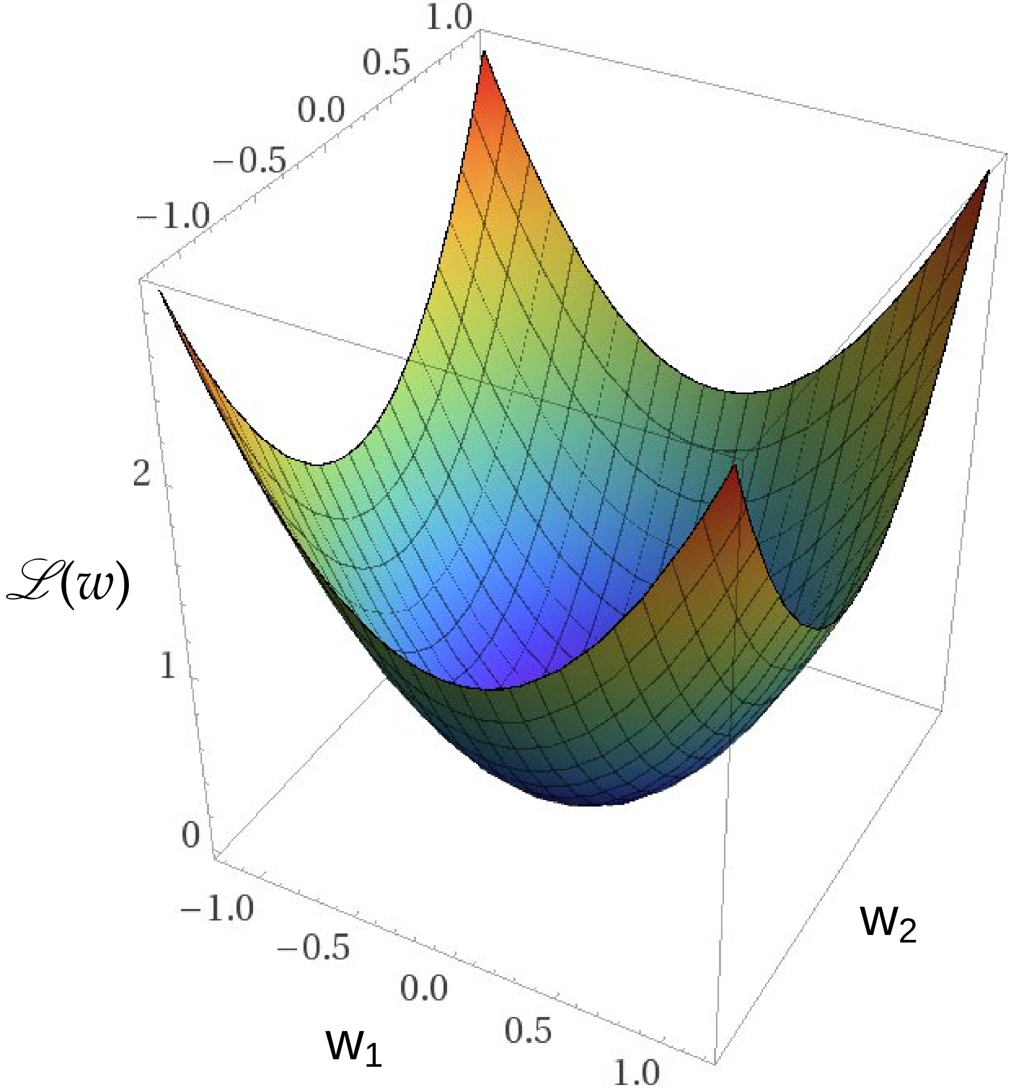
- Loss written vectorially:

$$\boldsymbol{\xi} = \mathbf{y} - \mathbf{X}\mathbf{w}$$

$$\mathcal{L}(\mathbf{w}, S) = \|\boldsymbol{\xi}\|_2^2 = (\mathbf{y} - \mathbf{X}\mathbf{w})'(\mathbf{y} - \mathbf{X}\mathbf{w})$$

- What is the optimal value for **w**?

Loss is convex



Linear regression

- The optimal **w**:

$$\frac{\partial \mathcal{L}(\mathbf{w}, S)}{\partial \mathbf{w}} = -2\mathbf{X}'\mathbf{y} + 2\mathbf{X}'\mathbf{X}\mathbf{w} = \mathbf{0}$$

- We get the normal equation:

$$\mathbf{X}'\mathbf{X}\mathbf{w} = \mathbf{X}'\mathbf{y}$$

- We can compute **w**, if the inverse exists:

$$\mathbf{w} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

Ridge Regression

- If the inverse does not exist, the problem is “ill-conditioned” and it needs regularization
- The optimization criterion becomes:

$$\min_{\mathbf{w}} \mathcal{L}_\lambda(\mathbf{w}, S) = \min_{\mathbf{w}} (\lambda \|\mathbf{w}\|^2 + \sum_{i=1}^{\ell} (y_i - g(\mathbf{x_i}))^2)$$

- And the optimal solution will be given by:

$$\frac{\partial \mathcal{L}_\lambda(\mathbf{w}, S)}{\partial \mathbf{w}} = \frac{\partial (\lambda \|\mathbf{w}\|^2 + \sum_{i=1}^{\ell} (y_i - g(\mathbf{x_i}))^2)}{\partial \mathbf{w}} = \mathbf{0}$$

Ridge Regression

- The optimal solution:

$$\frac{\partial \mathcal{L}_\lambda(\mathbf{w}, S)}{\partial \mathbf{w}} = \frac{\partial(\lambda\|\mathbf{w}\|^2 + (\mathbf{y} - \mathbf{X}\mathbf{w})'(\mathbf{y} - \mathbf{X}\mathbf{w}))}{\partial \mathbf{w}} =$$

$$= 2\lambda\mathbf{w} - 2\mathbf{X}'\mathbf{y} + 2\mathbf{X}'\mathbf{X}\mathbf{w} = \mathbf{0}$$

$$\mathbf{X}'\mathbf{X}\mathbf{w} + \lambda\mathbf{w} = \mathbf{X}'\mathbf{y}$$

$$(\mathbf{X}'\mathbf{X} + \lambda\mathbf{I}_n)\mathbf{w} = \mathbf{X}'\mathbf{y}$$

$$\mathbf{w} = (\mathbf{X}'\mathbf{X} + \lambda\mathbf{I}_n)^{-1}\mathbf{X}'\mathbf{y}$$

Dual Ridge Regression

$$\mathbf{X}'\mathbf{X}\mathbf{w} + \lambda\mathbf{w} = \mathbf{X}'\mathbf{y}$$

$$\mathbf{w} = \lambda^{-1}(\mathbf{X}'\mathbf{y} - \mathbf{X}'\mathbf{X}\mathbf{w}) = \lambda^{-1}\mathbf{X}'(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{X}'\boldsymbol{\alpha}$$

$$\lambda^{-1}\mathbf{X}'(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{X}'\boldsymbol{\alpha}$$

$$\boldsymbol{\alpha} = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{w})$$

- But:

[redacted]

- So:

$$\boldsymbol{\alpha} = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{X}'\boldsymbol{\alpha})$$

Dual Ridge Regression

$$\boldsymbol{\alpha} = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{X}'\boldsymbol{\alpha})$$

$$\lambda\boldsymbol{\alpha} = (\mathbf{y} - \mathbf{X}\mathbf{X}'\boldsymbol{\alpha})$$

$$\mathbf{X}\mathbf{X}'\boldsymbol{\alpha} + \lambda\boldsymbol{\alpha} = \mathbf{y}$$

$$(\mathbf{X}\mathbf{X}' + \lambda\mathbf{I}_\ell)\boldsymbol{\alpha} = \mathbf{y}$$

$$\boldsymbol{\alpha} = (\mathbf{G} + \lambda\mathbf{I}_\ell)^{-1}\mathbf{y}$$

- Where:

[redacted]

- is called the Gram matrix:

$$\mathbf{G}_{ij} = \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle$$

Dual Ridge Regression

- In the dual form, the information from the training samples is given only by the inner product between pairs of training points:

$$\boldsymbol{\alpha} = (\boxed{\mathbf{G}} + \lambda \mathbf{I}_{\ell})^{-1} \mathbf{y}$$

- The predictive function is given by:

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \left\langle \sum_{\mathbf{i}=\mathbf{1}}^{\ell} \alpha_{\mathbf{i}} \mathbf{x}_i, \mathbf{x} \right\rangle = \sum_{i=1}^{\ell} \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle$$

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

	x_1	x_2	x_3	x_4
x_1	55	31	16	11
x_2	31	30	14	7
x_3	16	14	35	17
x_4	11	7	17	16

$= X * X' = K_X$

l_1	1
l_2	1
l_3	-1
l_4	-1

$= L$

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

	x_1	x_2	x_3	x_4
y_1	36	26	14	9
y_2	16	13	15	12
y_3	25	18	18	9

$= Y * X' = K_Y$

p_1	?
p_2	?
p_3	?

$= P$

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

Kernel Ridge Regression

- We can now apply the "kernel trick", replacing the scalar product with another kernel function *k*:

$$\langle \ \rangle \mapsto k$$

$$\mathbf{G} = \begin{pmatrix} \langle \mathbf{x}_1, \mathbf{x}_1 \rangle & \langle \mathbf{x}_1, \mathbf{x}_2 \rangle & \cdots & \langle \mathbf{x}_1, \mathbf{x}_n \rangle \\ \langle \mathbf{x}_2, \mathbf{x}_1 \rangle & \langle \mathbf{x}_2, \mathbf{x}_2 \rangle & \cdots & \langle \mathbf{x}_2, \mathbf{x}_n \rangle \\ \vdots & \vdots & \vdots & \vdots \\ \langle \mathbf{x}_n, \mathbf{x}_1 \rangle & \langle \mathbf{x}_n, \mathbf{x}_2 \rangle & \cdots & \langle \mathbf{x}_n, \mathbf{x}_n \rangle \end{pmatrix} \mapsto \mathbf{K} = \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \cdots & k(\mathbf{x}_1, \mathbf{x}_n) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \cdots & k(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \vdots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & k(\mathbf{x}_n, \mathbf{x}_2) & \cdots & k(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix}$$

Kernel Ridge Regression

- The dual weights are computed as follows:

$$\boldsymbol{\alpha} = (\mathbf{G} + \lambda \mathbf{I}_\ell)^{-1}\mathbf{y} \;\rightarrow\; \boldsymbol{\alpha} = (\mathbf{K} + \lambda \mathbf{I}_\ell)^{-1}\mathbf{y}$$

- The predictive function becomes:

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \left\langle \sum_{\mathbf{i=1}}^{\ell} \alpha_{\mathbf{i}} \mathbf{x}_i, \mathbf{x} \right\rangle = \sum_{i=1}^{\ell} \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle$$

$$\downarrow$$

$$g(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i k(\mathbf{x}_i, \mathbf{x})$$

Kernel Ridge Regression (Python)

```
# Regularization parameter lambda:
lmb = 10 ** -6

# X_train – training data (one sample per line)
# T_train – training labels

n = X_train.shape[0]
K = np.matmul(X_train, X_train.T)

# Training the method:
alpha = np.matmul(np.linalg.inv(K + lmb * np.eye(n)),
        T_train)

# Predicting the training labels:
Y_train = np.matmul(K, alpha)
Y_train = np.sign(Y_train)

acc_train = (T_train == Y_train).mean())
print('Train accuracy: %.4f' % acc_train)
```

Kernel Ridge Regression (Python)

```
# X_test – test data (one sample per line)
# T_test – test labels

K_test = np.matmul(X_test, X_train.T)

# Predicting the test labels:
Y_test = np.matmul(K_test, alpha)
Y_test = np.sign(Y_test)

acc_test = (T_test == Y_test).mean()
print('Test accuracy: %.4f' % acc_test)
```

The Kernel Function

- **Definition:** A kernel is a function

$$k : X \times X \longmapsto \mathbb{R}$$

for which there is a mapping from X to a Hilbert space F

$$\phi : x \in \mathbb{R}^m \longmapsto \phi(x) \in F$$

such that for any $x, z \in X$ we have:

$$k(x, z) = \langle \phi(x), \phi(z) \rangle$$

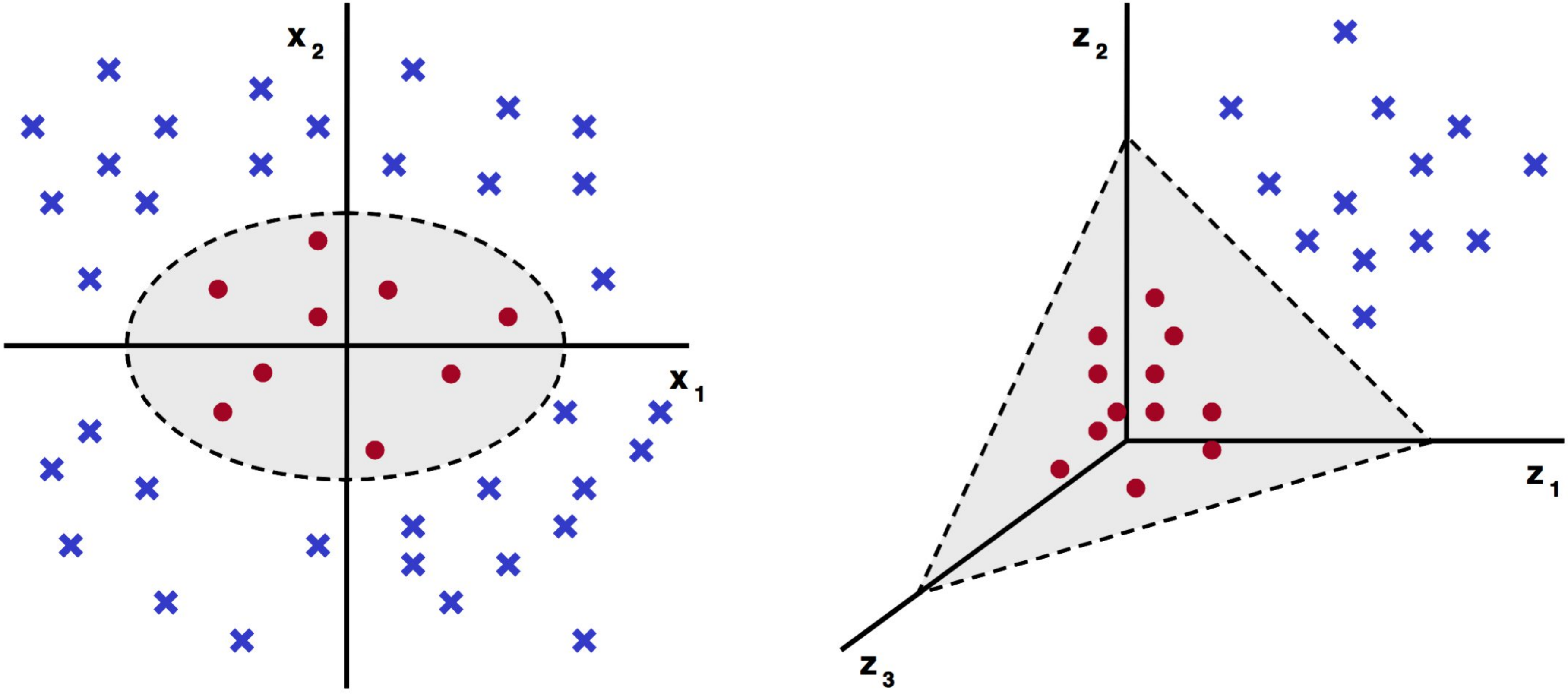
- **Theorem:** The function k is a kernel if and only if k is symmetric and finitely positive semi-definite

Examples of kernels

- By explicitly providing the embedding map:

$$\phi : \mathbb{R}^2 \to \mathbb{R}^3$$

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) = \left(x_1^2, \sqrt{2}x_1 x_2, x_2^2\right)$$



Examples of kernels

- The kernel from the previous example:

$$\left\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \right\rangle_F = \left\langle (x_1^2, x_2^2, \sqrt{2}x_1 x_2), (z_1^2, z_2^2, \sqrt{2}z_1 z_2) \right\rangle$$

$$\left\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \right\rangle_F = x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1 x_2 z_1 z_2$$

$$\left\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \right\rangle_F = (x_1 z_1 + x_2 z_2)^2$$

$$\left\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \right\rangle_F = \left\langle \mathbf{x}, \mathbf{z} \right\rangle^2$$

$$k(\mathbf{x}, \mathbf{z}) = \left\langle \mathbf{x}, \mathbf{z} \right\rangle^2$$

- The same kernel also corresponds to this map:

$$\phi : \mathbf{x} = (x_1, x_2) \mapsto \phi(\mathbf{x}) = (x_1^2, x_2^2, x_1 x_2, x_2 x_1)$$

The Polynomial Kernel

- For a real positive constant *c* and a natural number *d*:

$$k(\mathbf{x},\mathbf{z}) = (\langle \mathbf{x},\mathbf{z} \rangle + c)^d$$

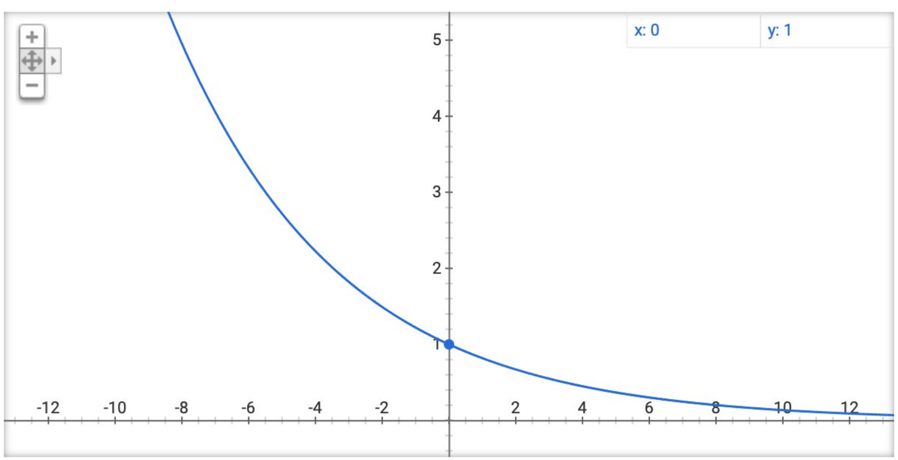
- The constant c controls the amount of influence of polynomials of various degrees

The Gaussian (RBF) Kernel

- For $x = (1, 2, 4, 1)$ and $z = (5, 1, 2, 3)$ from $\mathbb{R}^4$:

$$
\begin{aligned}
k(x, z) &= \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right) \\
&= \exp\left(-\frac{\sqrt{(1-5)^2 + (2-1)^2 + (4-2)^2 + (1-3)^2}}{2 \cdot 1^2}\right) \\
&= \exp\left(-\frac{\sqrt{16 + 1 + 4 + 4}}{2}\right) \\
&= \exp\left(-\frac{5}{2}\right) \\
&\approx 0.0821.
\end{aligned}
$$

Graph for e^((-x)/5)



The Intersection Kernel

- For $x = (1, 2, 4, 1)$ and $z = (5, 1, 2, 3)$ from $\mathbb{R}^4$:

$$
\begin{aligned}
k(x, z) &= \sum_i \min\{x_i, z_i\} \\
&= \min\{1, 5\} + \min\{2, 1\} + \min\{4, 2\} + \min\{1, 3\} \\
&= 1 + 1 + 2 + 1 \\
&= 5.
\end{aligned}
$$

Other kernel functions

- The Hellinger kernel:

$$k(x, z) = \sum_i \sqrt{x_i \cdot z_i}$$

- The PQ kernel [Ionescu & Popescu, PRL15]:

$$k_{PQ}(X, Y) = 2(P - Q)$$

$$P = |\{(i, j) : 1 \leq i < j \leq n, (x_i - x_j)(y_i - y_j) > 0\}|$$

$$Q = |\{(i, j) : 1 \leq i < j \leq n, (x_i - x_j)(y_i - y_j) < 0\}|$$

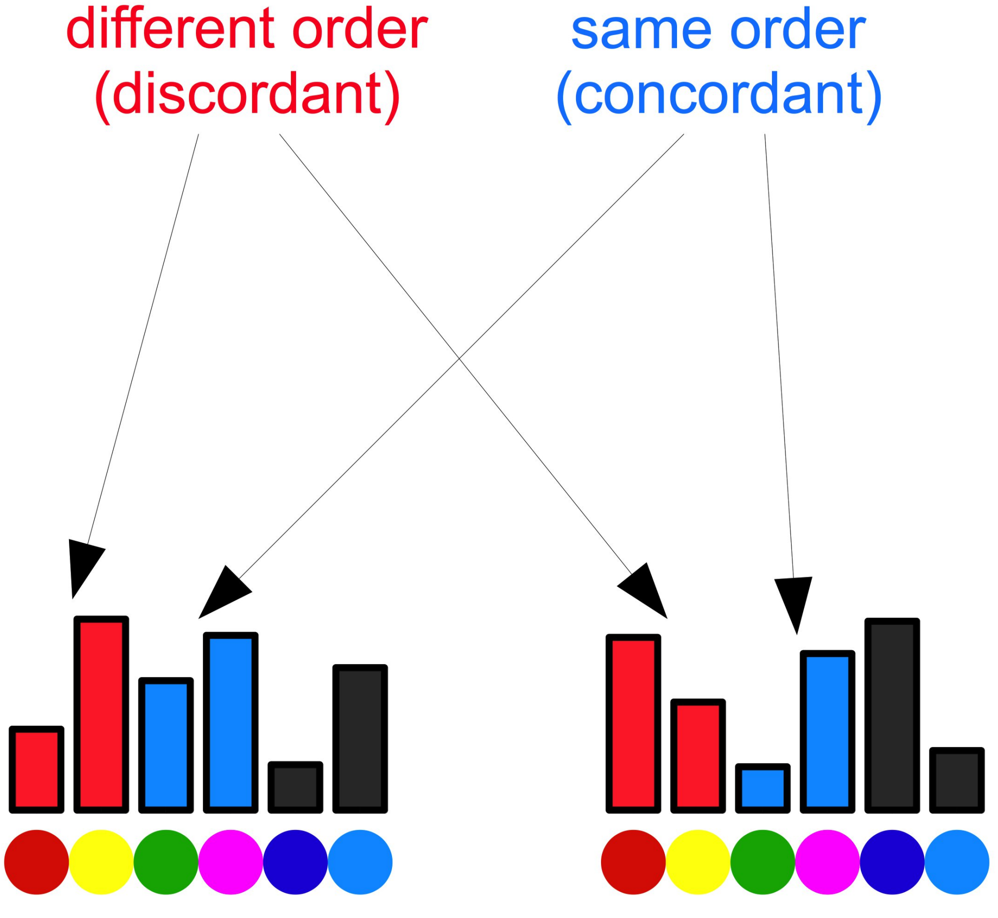
Other kernel functions

- The Hellinger kernel:

$$k(x,z) = \sum_i \sqrt{x_i \cdot z_i}$$

- The PQ kernel:

$$k_{PQ}(X,Y) = 2(P-Q)$$



String kernels

- String kernels measure the similarity of strings, by counting the number of contiguous subsequences (n-grams) of characters that two strings have in common
- Text documents can be interpreted as strings
- Advantages:

➢ We do not have to tokenize the text

➢ Language independence (can be applied to any language, only re-training is necessary)

String kernels

- Example:

Given s = “pineapple pi” and t = “apple pie” over an alphabet Σ, and the n-gram length p = 2,

the hash maps S and T contain <key>:<value> pairs of the type <2-gram>:<number of occurrences> in s and t:

S = {pi:2, in:1, ne:1, ea:1, ap:1, pp:1, pl:1, le:1, e_:1, _p:1},

T = {ap:1, pp:1, pl:1, le:1, e_:1, _p:1, pi:1, ie:1}

Presence bits string kernel

- The presence bits string kernel is defined as follows:

$$k_2^{0/1}(s,t) = \sum_{v \in \Sigma^p} S^{0/1}(v) \cdot T^{0/1}(v)$$

- Example (continued):

S = {pi:2, in:1, ne:1, ea:1, ap:1, pp:1, pl:1, le:1, e_:1, _p:1},

T = {ap:1, pp:1, pl:1, le:1, e_:1, _p:1, pi:1, ie:1}

$$\begin{aligned} k_2^{0/1}(s,t) &= 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 \\ &= 1 + 1 + 1 + 1 + 1 + 1 + 1 \\ &= 7 \end{aligned}$$

Why kernel methods?

- They obtain state-of-the-art results in various NLP tasks:
  - ➢ Native Language Identification [Ionescu & Popescu, BEA17]
  - ➢ Arabic Dialect Identification [Butnaru & Ionescu, VarDial18]
  - ➢ Romanian Dialect Identification [Butnaru & Ionescu, ACL19]
- Useful for building a compact representation when:

number of samples << number of features

- E.g., in TOEFL11 dataset, the number of unique n-grams (n={5,6,7,8,9}) is:

4,662,520

- … versus the number of training samples:

11,000

Why kernel methods?

- They generalize better than words
- Examples of native language transfer patterns on TOEFL11

German		French		Arabic		Hindi		Spanish		Chinese	
1	, that	1	indeed	1	alot	2	as compa	1	, is	2	t most
6	german	19	onnal	9	any	9	hence	2	difer	4	chin
11	. but	21	is to	13	them	16	then	13	, but	7	just
13	often	26	franc	16	thier	17	indi	15	, etc	8	still
207	special	28	to concl	19	his	21	towards	17	cesar	14	. take

Italian		Japanese		Korean		Telugu		Turkish			
1	ital	1	japan	1	korea	1	i concl	1	i agree.		
3	o beca	15	. if	24	e that	6	days	11	turk		
4	fact	19	i disa	27	. as	7	.the	21	. becau		
9	, for	27	. the	30	soci	11	where as	32	s about		
24	the life	38	. it	36	. also	13	e above	37	being		

French→English transfer patterns

- {onnal}

"…many academics subjects. Additionnally, people always have a subject…"
"I would not be in control of my personnal schedule during the trip."

- {evelopp}

"…and who will have the curiosity to developp research on the disease."
"…be able to do so. Underdevelopped countries are a case in point."

- {n France}

"…studied law in both England and in France, I have had the chance…"
"Numbers have actually shown that in France the number of new cars…"

- {to conc}

"…without a tour guide. To conclude, there are several advantages…"
"…job they will enjoy. To conclude, I think that the best solution is…"

- {exemple}

"…after using them. Onother exemple is my underwear that I bougth…"
"Science is a great exemple of how successful people want to improve…"

New kernels based on combinations

- Given two kernels k_1 and k_2, a real positive constant a, a function f with real values and a symmetric and positive semi-definite matrix B, the following functions are also kernels:

$$(i)\ k(x, z) = k_1(x, z) + k_2(x, z);$$

$$(ii)\ k(x, z) = ak_1(x, z);$$

$$(iii)\ k(x, z) = k_1(x, y) \cdot k_2(x, z);$$

$$(iv)\ k(x, z) = f(x) \cdot f(z);$$

$$(v)\ k(x, z) = x'Bz.$$

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

	x_1	x_2	x_3	x_4
x_1	55	31	16	11
x_2	31	30	14	7
x_3	16	14	35	17
x_4	11	7	17	16

= X * X' = K_X

l_1	1
l_2	1
l_3	-1
l_4	-1

= L

Linear classifier: C = (α_1, α_2, α_3, α_4, b) such that sign(K_X * α' + b) = L

Test samples:
y_1, y_2, y_3

	x_1	x_2	x_3	x_4
y_1	36	26	14	9
y_2	16	13	15	12
y_3	25	18	18	9

= Y * X' = K_Y

p_1	?
p_2	?
p_3	?

= P

Apply C to obtain predictions: P = sign(K_Y * α' + 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)
```

Bibliography

Advances in Computer Vision and Pattern Recognition

ACVPR

Radu Tudor Ionescu
Marius Popescu

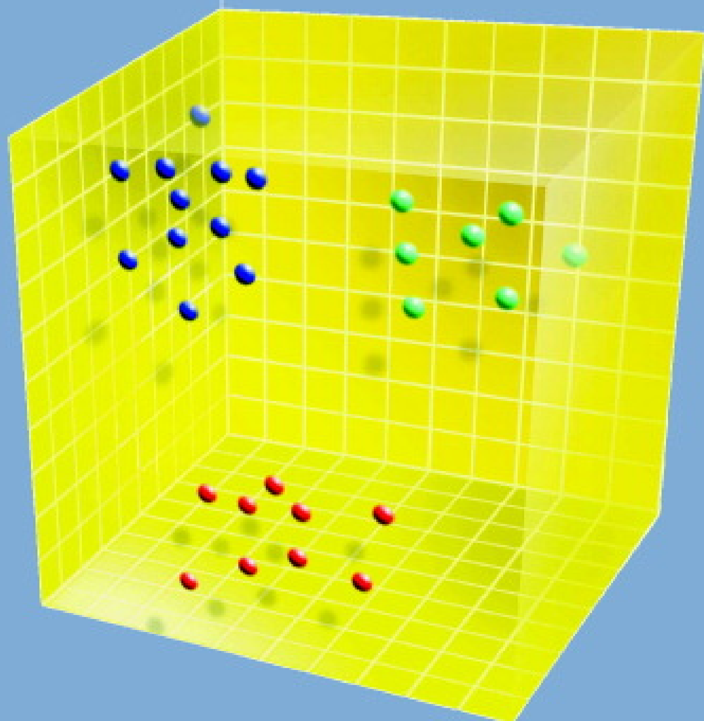
Knowledge Transfer between Computer Vision and Text Mining

Similarity-based Learning Approaches

Springer

John Shawe-Taylor
and Nello Cristianini

Kernel Methods for Pattern Analysis



CAMBRIDGE