EECS 553 Lecture 12
Machine Learning (ECE)
[class version]

Jeff Fessler

University of Michigan
©Laura Balzano and Jeff Fessler, UM, October 8, 2026, 13:16

F26

Logistics

© 2026 Fessler

- PA2: Fri. Oct. 9: reproducibility paper selected
- See L11 for Exam1 info

- Recap
  - KKT sufficient conditions
  - Optimal soft-margin hyperplane
- Today

  - Kernel methods
    - Kernel method for OSM, *i.e.*, SVM
    - Kernel nearest neighbor classification
    - Kernel ridge regression

Kernel methods: Outline

© 2026 Fessler

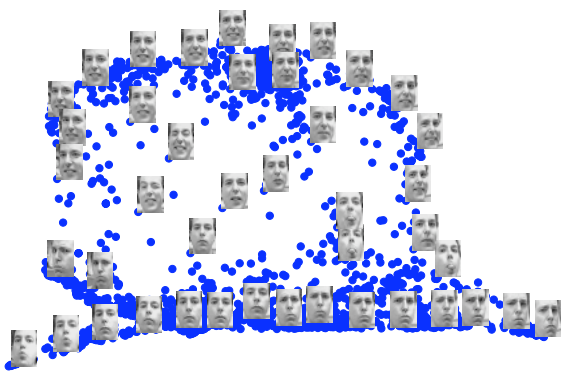
Outline:

- Nonlinear classification or regression via nonlinear feature maps
- Kernels [2]
- Kernel method for OSM classifier,
  i.e., the **support vector machine** (**SVM**)

Kernel methods: Motivation

© 2026 Fessler

- We have explored linear regression and "linear" classification methods.
- They are simple to compute and analyze, but they are often sufficient for complex modern ML data.
- Datasets often have [3, p. 173].



This face image dataset is visualized by taking a nonlinear function of high-dimensional image data, projecting into two dimensions. Even in the two dimensional space, it has nonlinear structure that defines different aspects of the image.

Figure 5: 2-D embedding of Frey face data using SMCE.

Figure courtesy [4].

Kernel methods: Feature map I

- One way to create a nonlinear method for regression or classification is to first transform each feature vector via a **nonlinear feature map**:

When $m > d$, we "lift" the features into a higher-dimensional space.

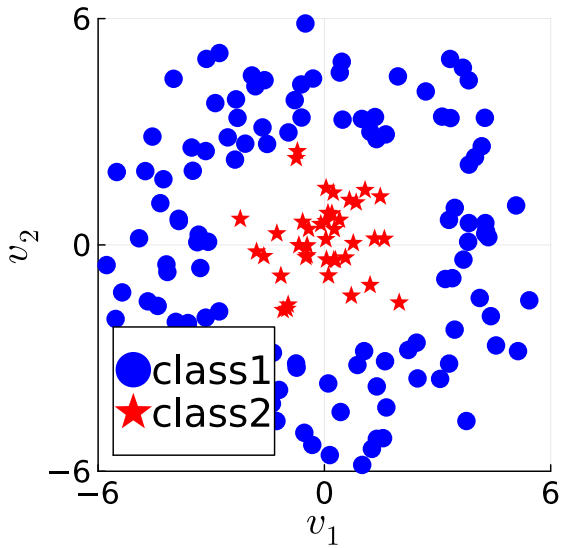
- We call $\mathbb{R}^m$ the "lifted space"
- We apply this transform to each feature vector (both train and test):

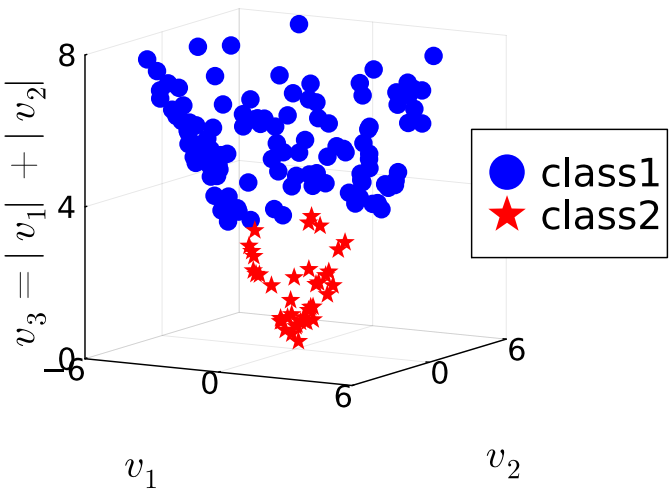
$$\begin{bmatrix} \boldsymbol{x}_1 & \dots & \boldsymbol{x}_n \end{bmatrix} \mapsto$$

- The map Φ might be hand-picked, or might be learned from training data.

Kernel methods: Feature map II

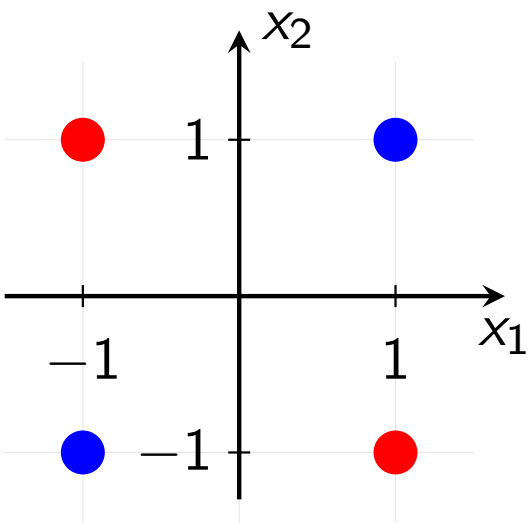
- Then typically we apply a "linear" method to the transformed data.
- Example.





- Resting heart rate in BPM

- The classic "XOR" classification problem [5, p. 264]:



- In regression, the nonlinear **regression** function is:

$$f(\boldsymbol{x}) =$$

  where $\boldsymbol{w} \in \mathbb{R}^m, b \in \mathbb{R}$

- In binary classification, the nonlinear **decision function** is:

$$f(\boldsymbol{x}) =$$

  where $\boldsymbol{w} \in \mathbb{R}^m, b \in \mathbb{R}$.

- The first layer of neural networks typically is a lifting $\boldsymbol{\Phi}(\boldsymbol{x})$
- Later in the semester we will see that kernel methods apply also to unsupervised problems, as in **Kernel PCA** [6].

Kernel example: Cubic polynomial regression

© 2026 Fessler

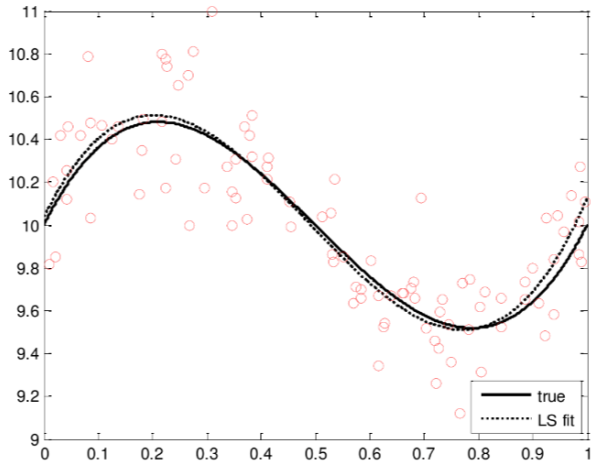


Figure courtesy Clay Scott.

For $x \in \mathbb{R}$, let

$$f(x) =$$

where

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}, \quad \boldsymbol{\Phi}(x) =$$

Note that $\boldsymbol{\Phi}(x)$ is nonlinear.

Example: Cubic polynomial regression

© 2026 Fessler

Because the cost function for fitting the model parameters $(b, \boldsymbol{w})$ is

$$J(\boldsymbol{\theta}) = J(b, \boldsymbol{w}) = \sum_i (f(x_i) - y_i)^2 =$$

the (unregularized) minimizer is

$$\begin{bmatrix} \hat{b} \\ \hat{\boldsymbol{w}} \end{bmatrix} = \begin{bmatrix} \hat{b} \\ \hat{w}_1 \\ \hat{w}_2 \\ \hat{w}_3 \end{bmatrix} =$$

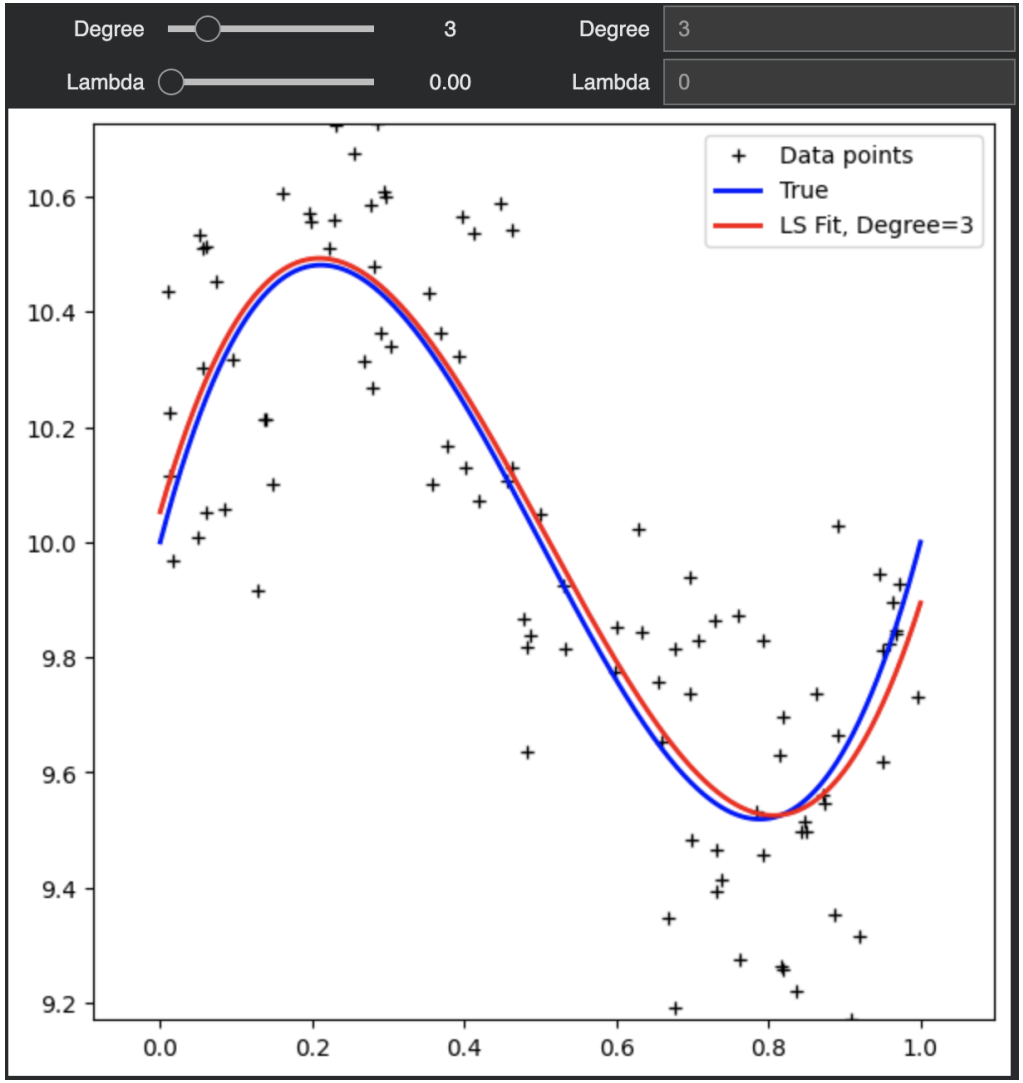
assuming $\bar{\boldsymbol{X}}$ has full column rank, where

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \quad \bar{\boldsymbol{X}} = \begin{bmatrix} 1 & \boldsymbol{\Phi}(x_1)^\top \\ \vdots & \vdots \\ 1 & \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ \vdots & & & \vdots \\ 1 & x_n & x_n^2 & x_n^3 \end{bmatrix}.$$

Polynomial regression demo

© 2026 Fessler

Colab demo



Challenges in using lifting

© 2026 Fessler

- As the polynomial degree p increases, $\bar{\boldsymbol{X}}^\top \bar{\boldsymbol{X}}$ becomes ill-conditioned and hard to invert.
  - This motivates regularization (as seen in the demo and 551 HW).
- More generally, as the dimension d of $\boldsymbol{x} \in \mathbb{R}^d$ grows, the dimension m of $\boldsymbol{\Phi}(\boldsymbol{x}) \in \mathbb{R}^m$ may grow considerably faster.
  - A homogeneous degree-p polynomial needs combinatorially many coefficients:

    (This is called the **number of combinations with repetition**).
  - Some useful functions $\boldsymbol{\Phi}$ even map to infinite dimensional spaces, *e.g.*, the **gaussian kernel** or **exponential kernel**
  - This makes it prohibitive to compute, store, or manipulate $\boldsymbol{\Phi}(\boldsymbol{x})$.
  - This challenge motivates **kernel methods**.

- Fortunately, the following facts allow us to use nonlinear feature maps for large m:
  - Many ML algorithms depend on $\boldsymbol{\Phi}(\boldsymbol{x})$ only via **inner products**:

$$\langle \boldsymbol{\Phi}(\boldsymbol{x}), \boldsymbol{\Phi}(\boldsymbol{x}') \rangle .$$

  - For certain $\boldsymbol{\Phi}$, we can compute efficiently the function

$$k(\boldsymbol{x}, \boldsymbol{x}') =$$

    even if m is huge or possibly infinite!
- The *function* $k : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ is called an **inner product kernel**.
- Let's first look at some examples involving the **dot product**.

Let $\boldsymbol{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \quad \boldsymbol{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \in \mathbb{R}^2.$

- The **inhomogeneous polynomial kernel** of degree 2 is

$$k(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u}^\top \boldsymbol{v} + 1)^2 =$$

- Fact. We (i.e., you!) can determine a feature map $\boldsymbol{\Phi}$ such that

$$k(\boldsymbol{u}, \boldsymbol{v}) = \langle \boldsymbol{\Phi}(\boldsymbol{u}), \boldsymbol{\Phi}(\boldsymbol{v}) \rangle .$$

For $d = 2$, consider the kernel

$$k(\boldsymbol{u}, \boldsymbol{v}) \triangleq (\boldsymbol{u}^\top \boldsymbol{v} + 1)^2 = (u_1 v_1 + u_2 v_2 + 1)^2 = \langle \boldsymbol{\Phi}(\boldsymbol{u}), \boldsymbol{\Phi}(\boldsymbol{v}) \rangle .$$

Q12.1 A mapping $\boldsymbol{\Phi}(\boldsymbol{x})$ that satisfies the above is given by?

A: $\begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2} x_1 x_2 \\ \sqrt{2} x_1 \\ \sqrt{2} x_2 \\ 1 \end{bmatrix}$ B: $\begin{bmatrix} x_1^2 \\ x_2^2 \\ x_1 x_2 \end{bmatrix}$ C: $\begin{bmatrix} x_1 \\ x_2 \\ \sqrt{2} x_1 x_2 \\ 1 \end{bmatrix}$ D: none of these

- Homogeneous **polynomial kernel**:

$$k(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u}^\top \boldsymbol{v})^p, \quad p \in \mathbb{N}$$

- Inhomogeneous polynomial kernel:

$$k(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u}^\top \boldsymbol{v} + c)^p, \quad p \in \mathbb{N}, \ c > 0$$

- **Radius basis function** (**RBF**) kernels: $k(\boldsymbol{u}, \boldsymbol{v}) = \phi(\|\boldsymbol{u} - \boldsymbol{v}\|)$
- **Gaussian kernel** (most important special case of RBF):

$$k(\boldsymbol{u}, \boldsymbol{v}) = \exp\left(-\frac{1}{2\sigma^2} \|\boldsymbol{u} - \boldsymbol{v}\|_2^2 \right), \ \sigma > 0$$

- For the Gaussian kernel, $m = \infty$ and the dot product is replaced by the ℓ_2 inner product: $\langle (a_i), (b_i) \rangle = \sum_{i=1}^{\infty} a_i b_i$.

Kernel methods abound

© 2026 Fessler

Kernels have been developed for many non-Euclidean data types, including

- string kernels
- pyramid match
- Trees (phylogenetic trees, parse trees)
- Graphs
- Probability distributions
- Riemannian manifolds
- ...

A big advantage of kernels is that they can be applied to data types for which there is no obvious or natural representation in Euclidean space.

Kernel methods are also important for understanding popular nonlinear ML methods including

- Convolutional neural networks
- Neural tangent kernel
- Average gradient outer product (AGOP)
- Kernel smoothing and attention mechanism

Let $k : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ be a **kernel**.

- We call k a **symmetric** kernel iff

- We call k **positive definite** iff each (kernel) **Gram matrix**

$$\begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_n, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{bmatrix}$$

  is

  for all $n \in \mathbb{N}$ and $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$.

- If k is both symmetric and positive definite, it is called a **symmetric, positive definite** (**SPD**) kernel.
- Fact. k is an **SPD kernel** iff k is an **inner product kernel**.
- Math foundation: **reproducing kernel Hilbert space** (**RKHS**)

Kernelizability

© 2026 Fessler

A ML method is called **kernelizable** iff it is possible to formulate the algorithm such that all training instances $\{\boldsymbol{x}_i\}$ and any test instance $\boldsymbol{x}$ occur in inner products of the form

Many ML methods have this property.

Kernel trick: Implementation

© 2026 Fessler

- Suppose Φ is a feature map associated with inner product kernel k.
- If we apply a kernelizable method to the **transformed** training data

  then the transformed feature vectors appear in the algorithm only via inner products with other transformed feature vectors.
- We implement efficiently using the key **inner product kernel** equality:

  eliminating the need to ever compute $\Phi(\boldsymbol{x})$ explicitly.
- This is why inner product kernels are so important.

- The problem is convex with linear constraints and strong duality holds.
- Let $\boldsymbol{\alpha}^*$ denote a dual solution, obtained by solving the dual quadratic program (QP).
- Let $(\boldsymbol{w}_*, b_*, \boldsymbol{\xi}^*)$ denote a primal solution.
- By the KKT necessity theorem, $(\boldsymbol{w}_*, b_*, \boldsymbol{\xi}^*, \boldsymbol{\alpha}^*)$ satisfy the KKT conditions.
- We may use these to compute $(\boldsymbol{w}_*, b_*, \boldsymbol{\xi}^*)$ from $\boldsymbol{\alpha}^*$.

- The OSM hyperplane can be solved with the dual:

$$\boldsymbol{\alpha}^* = \arg\max_{\boldsymbol{\alpha}} -\frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + \sum_{i=1}^{n} \alpha_i$$

$$\text{s.t} \sum_i \alpha_i y_i = 0$$

$$0 \le \alpha_i \le \frac{C}{n}, \quad \forall i = 1, \ldots, n.$$

Q12.2 Is this problem **kernelizable**?

A: Yes

B: No

C: Not enough information

(Kernel) Support Vector Machine I

- Let k be a **SPD kernel**.
- The **support vector machine** (**SVM**) is the classifier obtained by first solving

$$\boldsymbol{\alpha}^* = \arg\max_{\boldsymbol{\alpha}} \left(-\frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \quad + \sum_{i=1}^{n} \alpha_i \right)$$

$$\text{s.t.} \ \sum_i \alpha_i y_i = 0, \quad 0 \le \alpha_i \le \frac{C}{n}, \quad \forall i = 1, \ldots, n.$$

- Then find the primal solution via KKT conditions:
  - $\boldsymbol{w}_* = \sum_i \alpha_i^* y_i \boldsymbol{x}_i$
  - For any j with $0 < \alpha_j^* < \frac{C}{n}$:

$$b_* = y_j - \langle \boldsymbol{w}_*, \Phi(\boldsymbol{x}_j) \rangle = y_j - \sum_{i=1}^{n} \alpha_i^* y_i k(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

(Kernel) Support Vector Machine II

- Finally, the SVM classifier is expressed

$$f(\boldsymbol{x}) = \text{sign}(\tilde{f}(\boldsymbol{x})), \;\; \tilde{f}(\boldsymbol{x}) \triangleq$$

$$=$$

- Thankfully, there is no $\boldsymbol{\Phi}$ in the final expression, so this approach looks practical.

Q12.3 If implemented as expressed above, this method would be
A: parameteric B: nonparametric

Support vectors

© 2026 Fessler

- From complementary slackness:

$$\forall i$$

- we call $\boldsymbol{x}_i$ a **support vector** iff it satisfies:

- Thus, if $\boldsymbol{x}_i$ is *not* a support vector, then
- Key conclusion: $\boldsymbol{w}_*$ (and b_*) depend only on the SVs:

$$\boldsymbol{w}_* =$$

- This is very important for practical implementation!

- Previously the **decision boundary** was a **hyperplane** in $\mathbb{R}^d$:

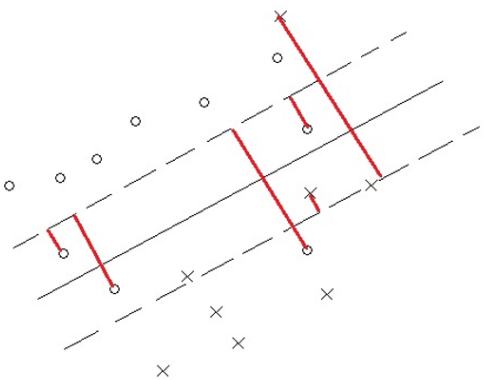
$$H_0 \triangleq \{\boldsymbol{x} \in \mathbb{R}^d : \boldsymbol{w}^\top \boldsymbol{x} + b = 0\} = \{\boldsymbol{x} \in \mathbb{R}^d : \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0\}$$

- After lifting, the **decision boundary** becomes (in general) some curved set:

$$H_0 \triangleq \{\boldsymbol{x} : \langle \boldsymbol{w}_*, \boldsymbol{\Phi}(\boldsymbol{x}) \rangle + b_* = 0\}$$

- The **margin boundaries** are the sets (would be hyperplanes if $\boldsymbol{\Phi}$ were affine) H_+ and H_- defined by

$$H_\pm \triangleq$$



vs

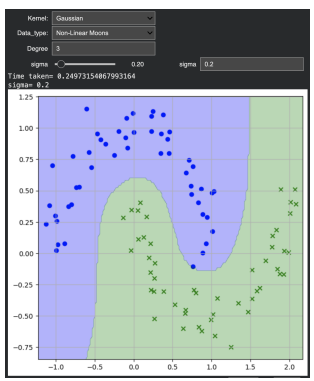
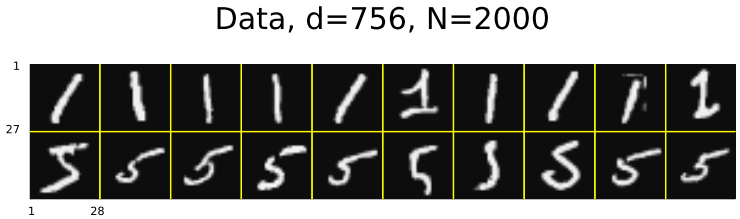
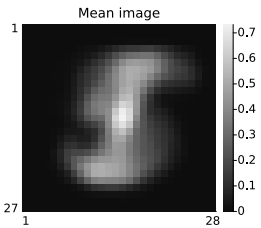
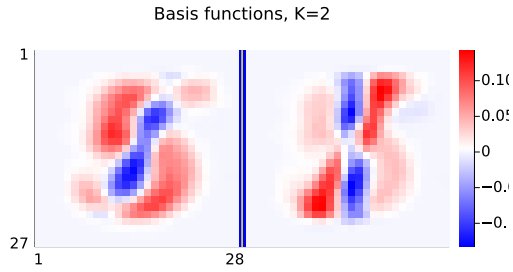


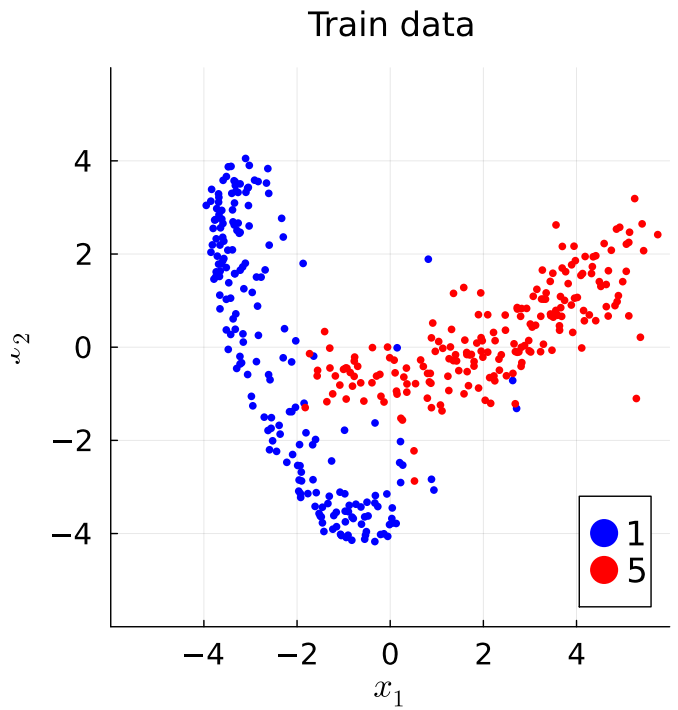
Figure courtesy Clay Scott.

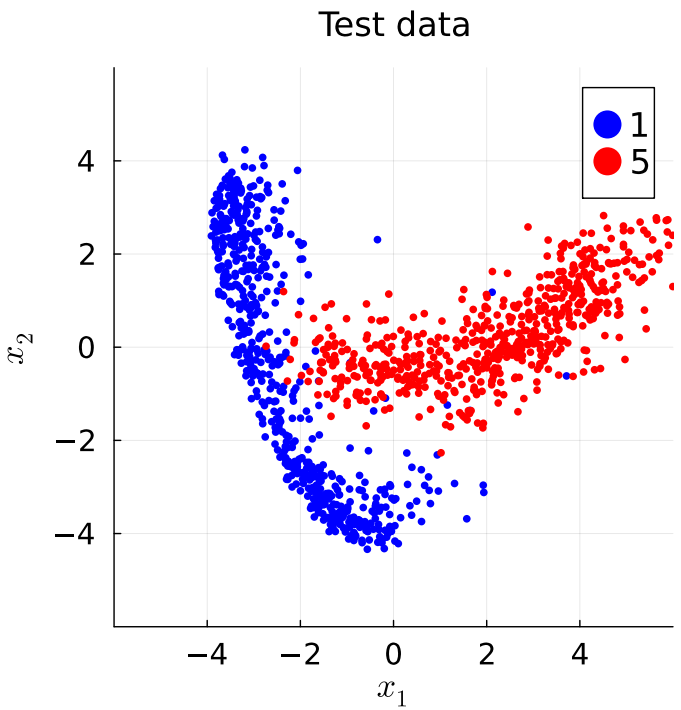
JULIA demo


Data, d=756, N=2000
1
27
1
28


Mean image
1
27
1
28
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0


Basis functions, K=2
1
27
1
28
0.1
0.05
0
−0.
−0.

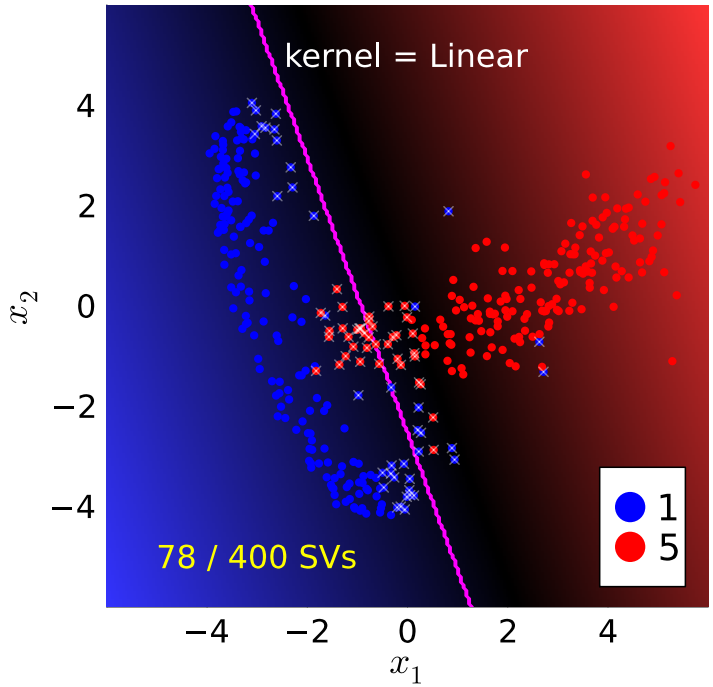

Train data
4
2
0
−2
−4
−4
−2
0
2
4
x_1
x_2
1
5

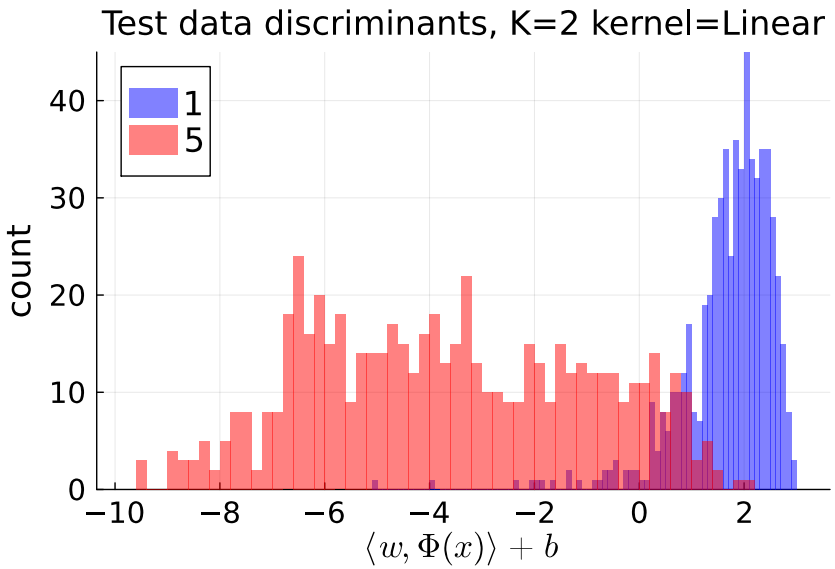

Test data
4
2
0
−2
−4
−4
−2
0
2
4
x_1
x_2
1
5

SVM demo: Linear Kernel, aka OSM

© 2026 Fessler

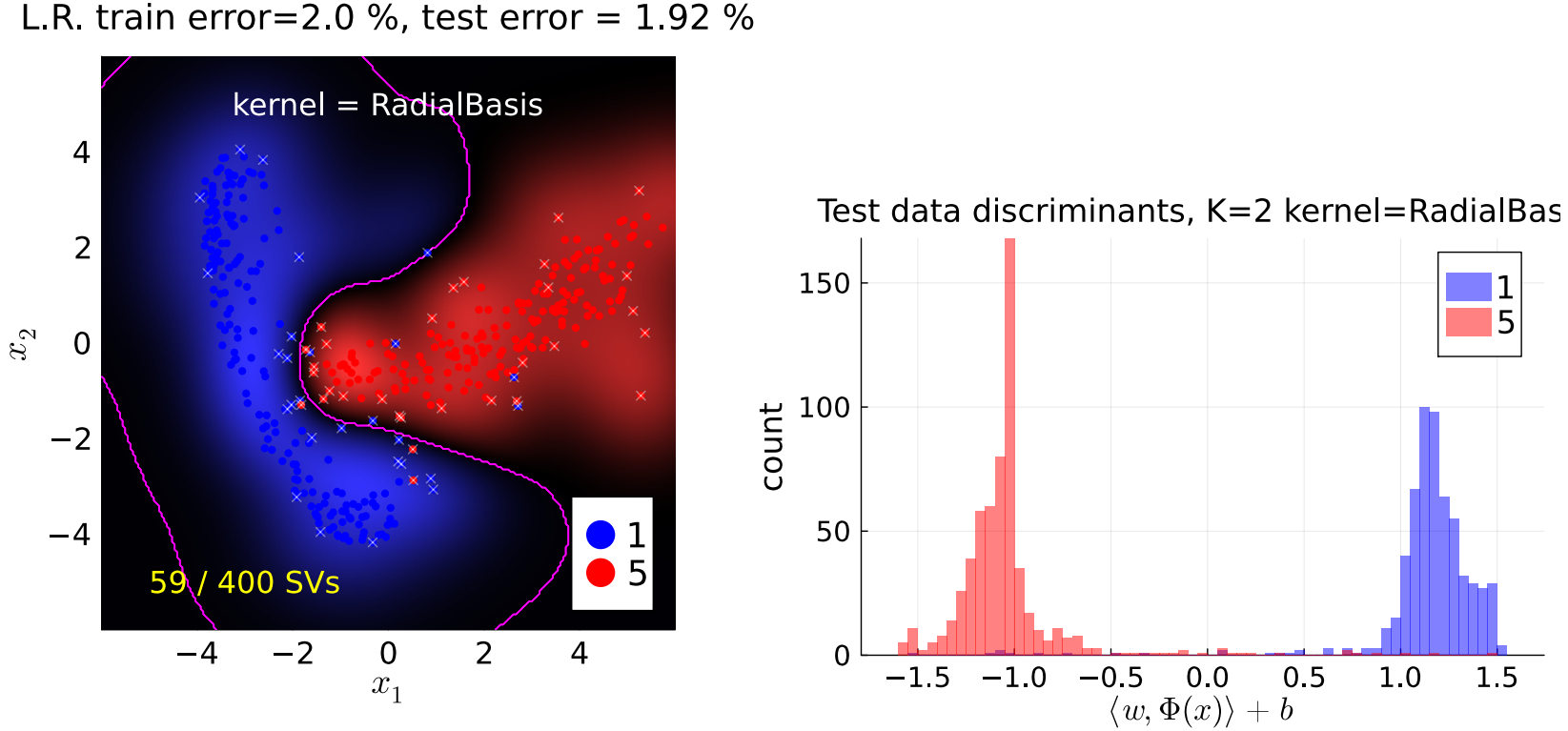
L.R. train error=7.75 %, test error = 7.67 %





SVM demo: Gaussian Kernel

© 2026 Fessler



- Linear kernel leads to hyperplance decision boundary essentially equivalent to OSM classifier $f(\boldsymbol{x}) = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x} + b)$
- Much lower classification error with Gaussian (RBF) kernel.
- Nonlinear decision boundary
- Conceptually: $f(\boldsymbol{x}) = \text{sign}(\langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle + b)$ (with inner products implemented more efficiently using kernel)

Margin errors

© 2026 Fessler

- Optimal slack variables satisfy

$$\xi_i^* = \max(0, 1 - y_i (\langle \boldsymbol{w}_*, \boldsymbol{x} \rangle + b_*)).$$

- There are four cases:
  1. If $y_i (\langle \boldsymbol{w}_*, \boldsymbol{x}_i \rangle + b_*) > 1$, then
  2. If $y_i (\langle \boldsymbol{w}_*, \boldsymbol{x}_i \rangle + b_*) = 1$, then
  3. If $0 \leq y_i (\langle \boldsymbol{w}_*, \boldsymbol{x}_i \rangle + b_*) < 1$, then
  4. If $y_i (\langle \boldsymbol{w}_*, \boldsymbol{x}_i \rangle + b_*) < 0$, then
- Case 2-4 are called **margin errors**.
- So the SVs are on the margin or misclassified.

Margin errors: Illustration

© 2026 Fessler

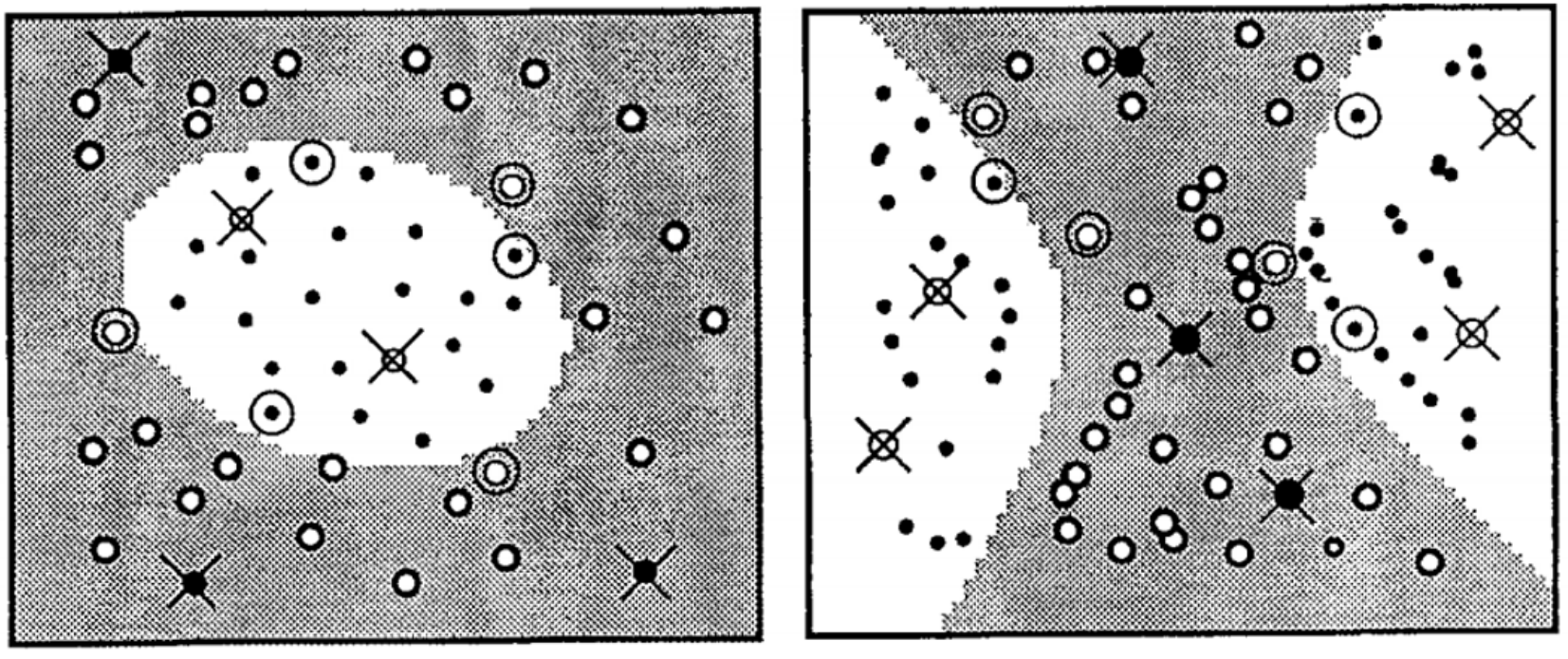


Figure courtesy original SVM paper [7]

Motivation for SVM

© 2026 Fessler

- The support vector machine (SVM) is one of the best "off-the-shelf" classification methods.
- "... looking at Kaggle challenges that are not related to vision or sequential tasks, gradient boosting, random forests, or SVMs are winning most of the competitions" [8]
- "On 75 small datasets with less than 1000 data points, random forests and SVMs outperform [FNNs]. On 46 larger datasets with at least 1000 data points, SNNs show the highest performance followed by SVMs and random forests" [8]
- "We evaluate 179 classifiers arising from 17 families ... We use 121 data sets ... The classifiers most likely to be the best are the random forest (RF) versions ... However, the difference is not statistically significant with the second best, the SVM with Gaussian kernel implemented in C using LibSVM" [9]
- There are multiclass extensions of SVM such as the Weston-Watkins method [10, 11, 12].

(You are not required to get this one right)

Q12.4 As of today, what is the number of citations n to the original 1995 SVM paper [7] on google scholar?

A: $0 < n < 1000$

B: $1000 < n < 10000$

C: $10000 < n < 100000$

D: $100000 < n$

E: none of these

Now we apply kernel trick to other classification and regression methods.

k-nearest neighbors: Recap

© 2026 Fessler

- Given training data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$, the **nearest neighbor classifier** assigns a test point $\boldsymbol{x}$ to the label

- More generally, a k-nearest neighbor (k-NN) classifier takes a "vote" among the k closest training instances (majority vote if binary).

- Suppose Φ is a feature map associated to an inner product/SPD kernel k, *i.e.*,

  for all $\boldsymbol{u}, \boldsymbol{v}$.
- Is the nearest neighbors method **kernelizable**?
- Depends on the norm.
  We focus hereafter on the **Euclidean norm**.

- To compute the nearest neighbors of a test point $\boldsymbol{x}$ we need

$$\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2 =$$

  which depends only on inner products of feature vectors.
- When lifted we have

- So for any $\boldsymbol{x} \in \mathbb{R}^d$, we can find the **nearest neighbor** in the lifted space via:

$$j = \arg\min_i$$

Kernelize the nearest centroid classifier I

- The **nearest centroid classifier** assigns a test point to the class whose **centroid** is closest to that test point:

$$f(\boldsymbol{x}) = \underset{k \in \{1,\ldots,K\}}{\arg\min} \|\boldsymbol{x} - \hat{\boldsymbol{\mu}}_k\|_2^2, \quad \hat{\boldsymbol{\mu}}_k \triangleq \frac{1}{n_k} \sum_{i\,:\,y_i = k} \boldsymbol{x}_i.$$

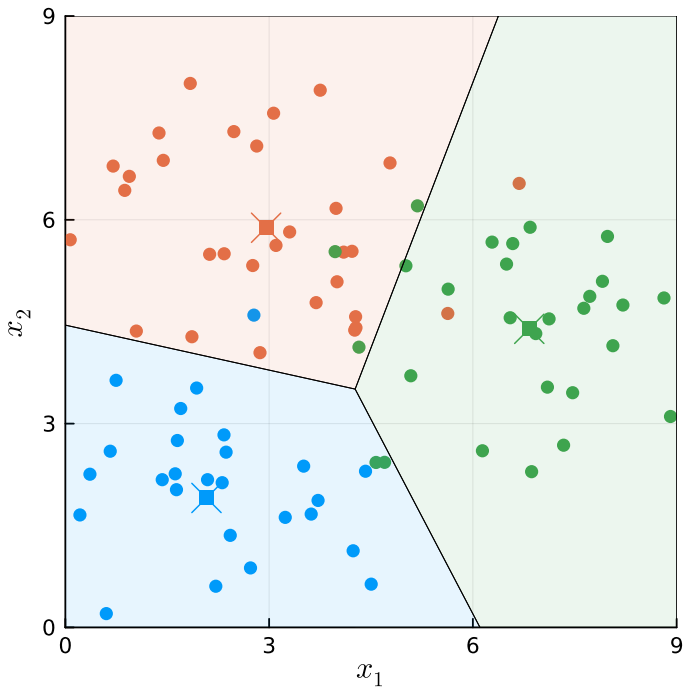
- The decision regions become **Voronoi cells**.

Q12.5 The **nearest centroid classifier** is

A: Parametric

B: Nonparametric

C: neither



Kernelize the nearest centroid classifier II

- The compute here is just $O(dK)$, not the $O(dn)$ needed for 1-NN.
- In the binary case with $\hat{\boldsymbol{\mu}}_{\pm 1} \triangleq \frac{1}{n_\pm} \sum_{i\,:\,y_i=\pm 1} \boldsymbol{x}_i$, the classier is:

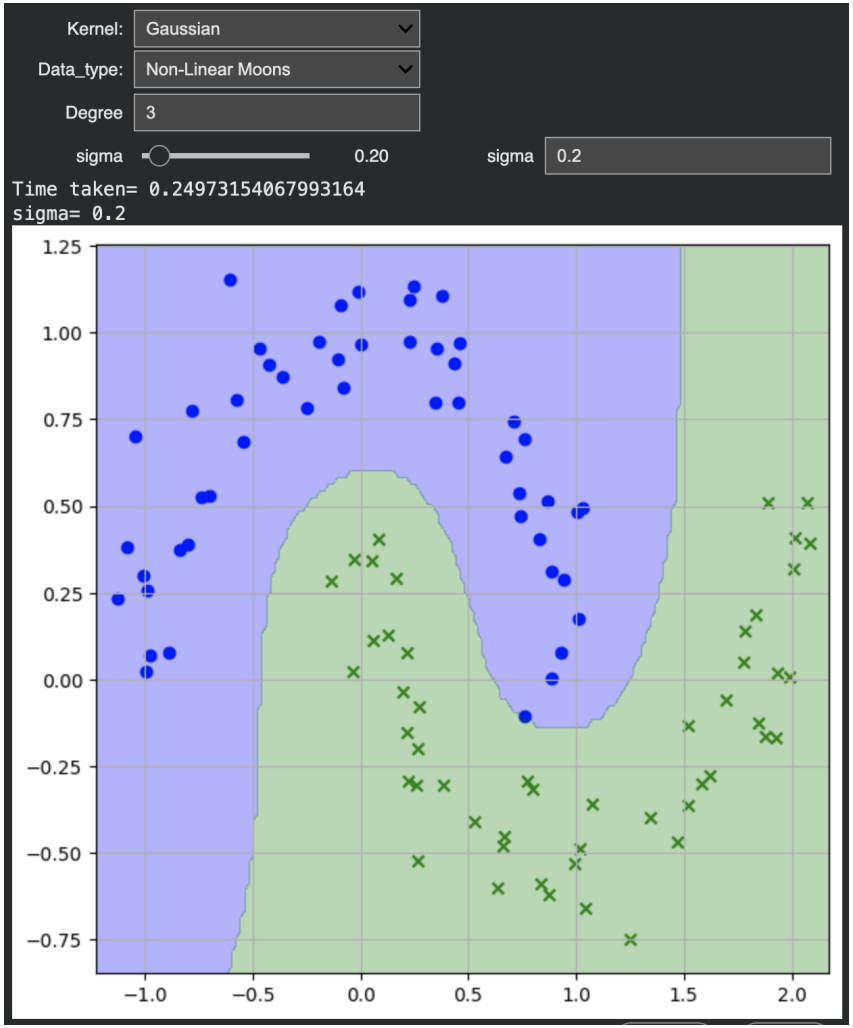
$$f(\boldsymbol{x}) = \text{sign}\Big(\|\boldsymbol{x} - \hat{\boldsymbol{\mu}}_{-1}\|_2^2 - \|\boldsymbol{x} - \hat{\boldsymbol{\mu}}_{+1}\|_2^2 \Big).$$

- By expanding the Euclidean norm squared, this method is readily **kernelizable**.

Demo: Kernel nearest centroid classifier

© 2026 Fessler

colab demo



Kernel ridge regression: Overview

© 2026 Fessler

- We will show that ridge regression is kernelizable, and use the kernel trick to extend it to a nonlinear regression method called **kernel ridge regression** (**KRR**).
- KRR has numerous applications, even quantitative MRI [13].
- To simplify the presentation, we focus on ridge regression *without offset*:

$$\hat{f}(\boldsymbol{x}) = \widehat{\boldsymbol{w}}^\top \boldsymbol{x}, \quad \widehat{\boldsymbol{w}} =$$

- Because some $\boldsymbol{\Phi}$ choices contain a constant term, as with the inhomogeous polynomial kernel, the offset is not always needed.
- KRR with offset is described in Professor Scott's lecture notes.

For $\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$, $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^\top \\ \vdots \\ \boldsymbol{x}_n^\top \end{bmatrix}$, the **ridge regression** (**RR**) training problem is:

$$\widehat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{w}^\top \boldsymbol{x}_i - y_i)^2 + \lambda \|\boldsymbol{w}\|_2^2 .$$

Q12.6 The solution is:

A: $\widehat{\boldsymbol{w}} = (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$

B: $\widehat{\boldsymbol{w}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$

C: $\widehat{\boldsymbol{w}} = (\boldsymbol{X} \boldsymbol{X}^\top + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$

D: $\widehat{\boldsymbol{w}} = \boldsymbol{X}^\top (\boldsymbol{X} \boldsymbol{X}^\top)^{-1} \boldsymbol{y}$

E: none of these

Kernelizing ridge regression I

- Note $\boldsymbol{X}^\top \boldsymbol{X}$ is *not* the **Gram matrix** of the training data, *i.e.*,
- Idea: apply a special case of the **matrix inversion lemma** called the **push-through identity** [14, p. 41]:

$$(a\boldsymbol{I} + \boldsymbol{B}\boldsymbol{D})^{-1}\boldsymbol{B} = \boldsymbol{B}(a\boldsymbol{I} + \boldsymbol{D}\boldsymbol{B})^{-1},$$

  with $\boldsymbol{B} = \boldsymbol{X}^\top$ and $\boldsymbol{D} = \boldsymbol{X}$, to get Gram matrix $\boldsymbol{X}\boldsymbol{X}^\top$.
- After simplification, the RR predictor without offset is

$$\begin{aligned}\hat{f}(\boldsymbol{x}) = \widehat{\boldsymbol{w}}^\top \boldsymbol{x} &= \boldsymbol{y}^\top \boldsymbol{X}(\boldsymbol{X}^\top \boldsymbol{X} + n\lambda \boldsymbol{I})^{-1}\boldsymbol{x} \\ &= \boldsymbol{y}^\top (\boldsymbol{X}\boldsymbol{X}^\top + n\lambda \boldsymbol{I})^{-1}\boldsymbol{X}\boldsymbol{x}.\end{aligned}$$

Kernel ridge regression: Key step

© 2026 Fessler

Rewriting with inner products:

$$\begin{aligned}\hat{f}(\boldsymbol{x}) &= \boldsymbol{y}^\top (\boldsymbol{X}\boldsymbol{X}^\top + n\lambda \boldsymbol{I})^{-1} \boldsymbol{X}\boldsymbol{x} \\ &= \boldsymbol{y}^\top (\boldsymbol{K} + n\lambda \boldsymbol{I})^{-1} \boldsymbol{k}(\boldsymbol{x})\end{aligned}$$

where the **kernel matrix** (Gram matrix) is

$$\boldsymbol{K} \triangleq \begin{bmatrix} \langle \boldsymbol{x}_1, \boldsymbol{x}_1 \rangle & \cdots & \langle \boldsymbol{x}_1, \boldsymbol{x}_n \rangle \\ \vdots & \ddots & \vdots \\ \langle \boldsymbol{x}_n, \boldsymbol{x}_1 \rangle & \cdots & \langle \boldsymbol{x}_n, \boldsymbol{x}_n \rangle \end{bmatrix}, \qquad \boldsymbol{k}(\boldsymbol{x}) = \begin{bmatrix} \langle \boldsymbol{x}_1, \boldsymbol{x} \rangle \\ \vdots \\ \langle \boldsymbol{x}_n, \boldsymbol{x} \rangle \end{bmatrix}.$$

To save test-time compute, precompute and store the vector

CQ: Kernel ridge regression

© 2026 Fessler

To define the KRR method, we simply replace all the inner products above (in both $\boldsymbol{K}$ and $\boldsymbol{k}$) with kernel values $k(\cdot, \cdot)$.

Q12.7 If k is such that $k(\boldsymbol{u}, \boldsymbol{v}) = \langle \boldsymbol{\Phi}(\boldsymbol{u}), \boldsymbol{\Phi}(\boldsymbol{v}) \rangle$, then the above KRR method is equivalent to

A: Apply ridge regression and then transform the solution into a nonlinear space with $\boldsymbol{\Phi}$.

B: Apply a nonlinear $\boldsymbol{\Phi}$ to the ridge regularization term.

C: Apply the feature map $\boldsymbol{\Phi}$ to all feature vectors, and then apply ridge regression in the new feature space.

D: none of these

Theorem 1 (Nonparametric Representer Theorem). *Suppose we are given a nonempty set $\mathcal{X}$, a positive definite real-valued kernel k on $\mathcal{X} \times \mathcal{X}$, a training sample $(x_1, y_1), \ldots, (x_m, y_m) \in \mathcal{X} \times \mathbf{R}$, a strictly monotonically increasing real-valued function g on $[0, \infty[$, an arbitrary cost function $c : (\mathcal{X} \times \mathbf{R}^2)^m \to \mathbf{R} \cup \{\infty\}$, and a class of functions*

$$\mathcal{F} = \left\{ f \in \mathbf{R}^{\mathcal{X}} \,\middle|\, f(\cdot) = \sum_{i=1}^{\infty} \beta_i k(\cdot, z_i), \beta_i \in \mathbf{R}, z_i \in \mathcal{X}, \|f\| < \infty \right\}. \tag{13}$$

Here, $\|\cdot\|$ is the norm in the RKHS H_k associated with k, i.e. for any $z_i \in \mathcal{X}, \beta_i \in \mathbf{R}$ ($i \in \mathbf{N}$),

$$\left\| \sum_{i=1}^{\infty} \beta_i k(\cdot, z_i) \right\|^2 = \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \beta_i \beta_j k(z_i, z_j). \tag{14}$$

Then any $f \in \mathcal{F}$ minimizing the regularized risk functional

$$c\left((x_1, y_1, f(x_1)), \ldots, (x_m, y_m, f(x_m))\right) + g\left(\|f\|\right) \tag{15}$$

admits a representation of the form

$$f(\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, x_i). \tag{16}$$

[15]

References I

[1] K. P. Murphy.
Probabilistic machine learning: An introduction.
MIT Press, 2022.

[2] T. Hofmann, B. Scholkopf, and A. J. Smola.
Kernel methods in machine learning.
Ann. Stat., 36(3):1171–220, June 2008.

[3] C. Bishop.
Pattern recognition and machine learning.
Springer, 2006.

[4] Ehsan Elhamifar and René Vidal.
Sparse manifold clustering and embedding.
Advances in neural information processing systems, 24, 2011.

[5] R. O. Duda, P. E. Hart, and D. G. Stork.
Pattern classification.
Wiley, New York, 2001.

[6] Sebastian Mika, Bernhard Schölkopf, Alex Smola, Klaus-Robert Müller, Matthias Scholz, and Gunnar Rätsch.
Kernel pca and de-noising in feature spaces.
Advances in neural information processing systems, 11, 1998.

[7] Corinna Cortes and Vladimir Vapnik.
Support-vector networks.
Machine learning, 20(3):273–297, 1995.

[8] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter.
Self-normalizing neural networks.
Advances in neural information processing systems, 30, 2017.

References II

[9] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim.
Do we need hundreds of classifiers to solve real world classification problems?
The journal of machine learning research, 15(1):3133–3181, 2014.

[10] J. Weston and C. Watkins.
Support vector machines for multi-class pattern recognition.
In *Proc. 7th European Symp. on Artif. Neural Net.*, 1999.

[11] C-W. Hsu and C-J. Lin.
A comparison of methods for multiclass support vector machines.
IEEE Trans. Neural Net., 13(2):415–425, 2002.

[12] Y. Wang and C. Scott.
An exact solver for the Weston-Watkins SVM subproblem.
In *Proc. Intl. Conf. Mach. Learn*, volume 139, pages 10894–904, 2021.

[13] G. Nataraj, J-F. Nielsen, C. D. Scott, and J. A. Fessler.
Dictionary-free MRI PERK: Parameter estimation via regression with kernels.
IEEE Trans. Med. Imag., 37(9):2103–14, September 2018.

[14] J. A. Fessler and R. R. Nadakuditi.
Linear algebra for data science, machine learning, and signal processing.
Cambridge, 2024.

[15] Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola.
A generalized representer theorem.
In *International conference on computational learning theory*, pages 416–426. Springer, 2001.