Introduction to Machine Learning 6

Classification ii

Tong Zhang UCAS 2025 fall

Recap

- Evaluating your trained model
- Cross-validation (k-fold)
- Set regularization for your models

Pipeline of Machine Learning

01

Data Preparation

(x_1, y_1)

(x_2, y_2)

(x_n, y_n)

- Cleaned dataset
- Feature matrix
- Training and testing datasets

02

Machine Learning Algorithm

Ml method, based on the data, you should efficiently pick the model

- Selected algorithm
- Algorithm configuration
- Baseline model
- Put your regularization loss

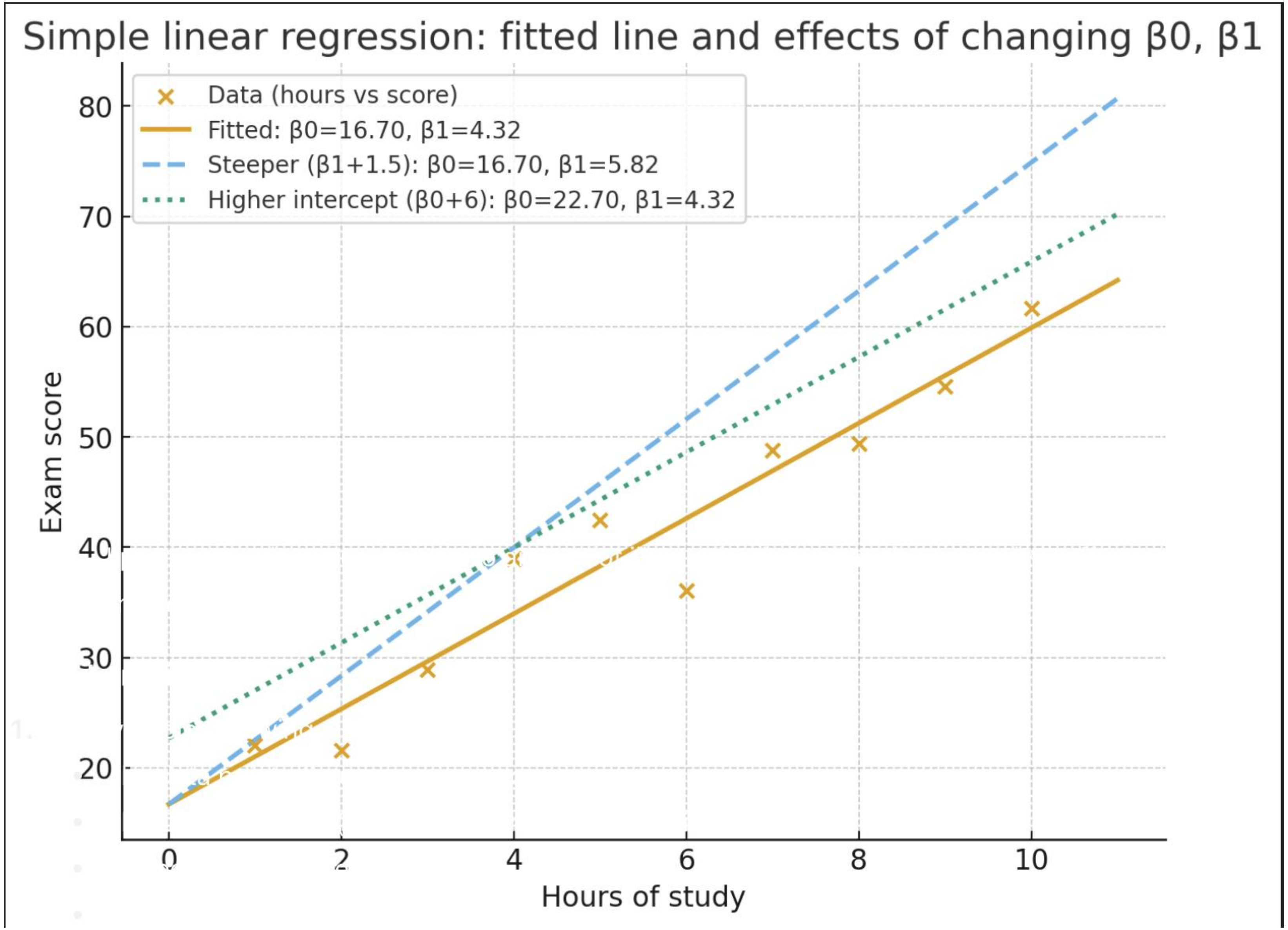
03

Training Recipes

Start to learn the rules in-explicitly:
How to optimize the model?
When do we need to stop?

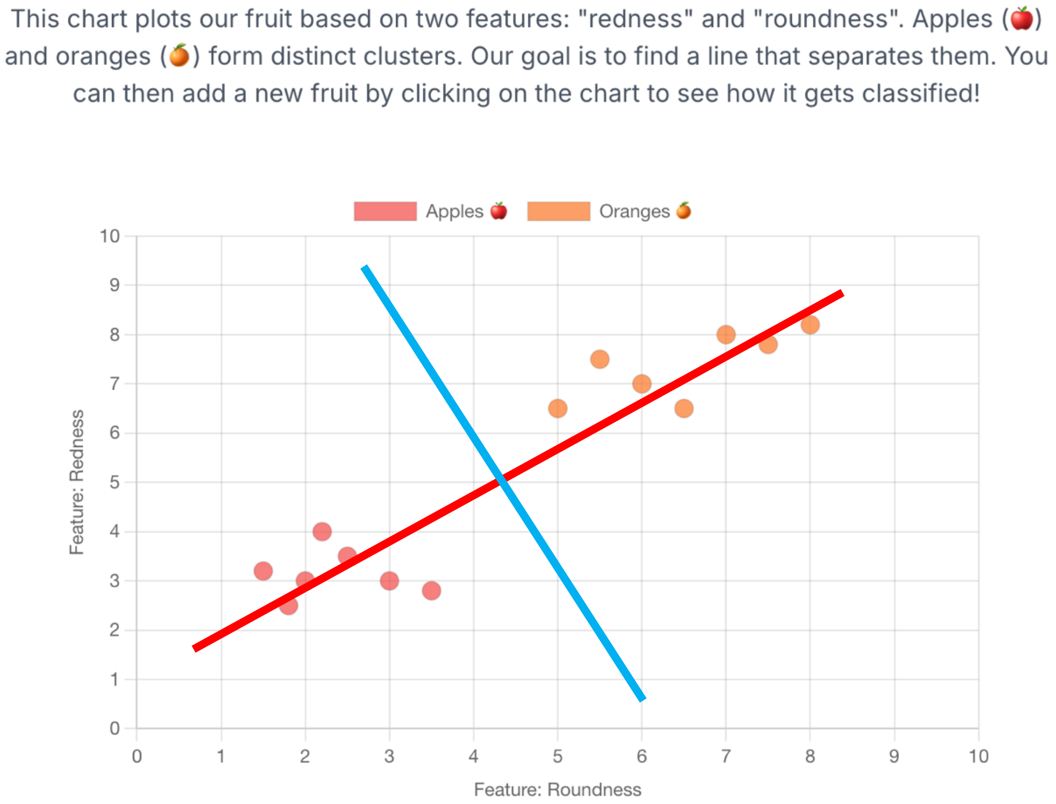
- Trained model: closed form and gradient descent (ADMM)
- Validation results: cross-validation
- Optimized hyperparameters

Regression


Simple linear regression: fitted line and effects of changing β0, β1
Data (hours vs score)
Fitted: β0=16.70, β1=4.32
Steeper (β1+1.5): β0=16.70, β1=5.82
Higher intercept (β0+6): β0=22.70, β1=4.32
Exam score
80
70
60
50
40
30
20
0
2
4
6
8
10
Hours of study

Classification v.s. Regression

In machine learning, the type of outcome we want to predict determines whether we are solving a regression or classification problem. This section provides a direct comparison to clarify the key differences between these two fundamental tasks.

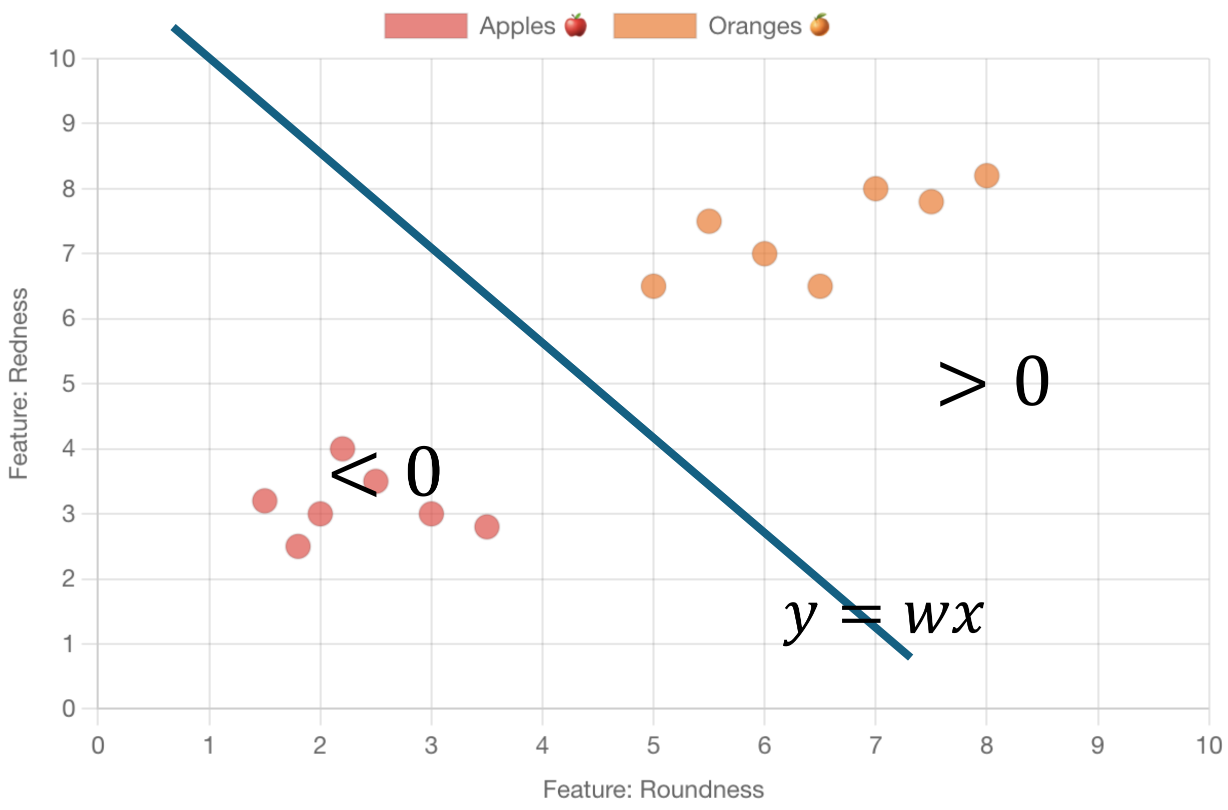


This chart plots our fruit based on two features: "redness" and "roundness". Apples (🍎) and oranges (🍊) form distinct clusters. Our goal is to find a line that separates them. You can then add a new fruit by clicking on the chart to see how it gets classified!

Naïve Bayes filter: Spam classification

Binary Classification

This chart plots our fruit based on two features: "redness" and "roundness". Apples (🍎) and oranges (🍊) form distinct clusters. Our goal is to find a line that separates them. You can then add a new fruit by clicking on the chart to see how it gets classified!



> 0

< 0

$y = wx$

Loss function

$$m = y \cdot (w^T z)$$

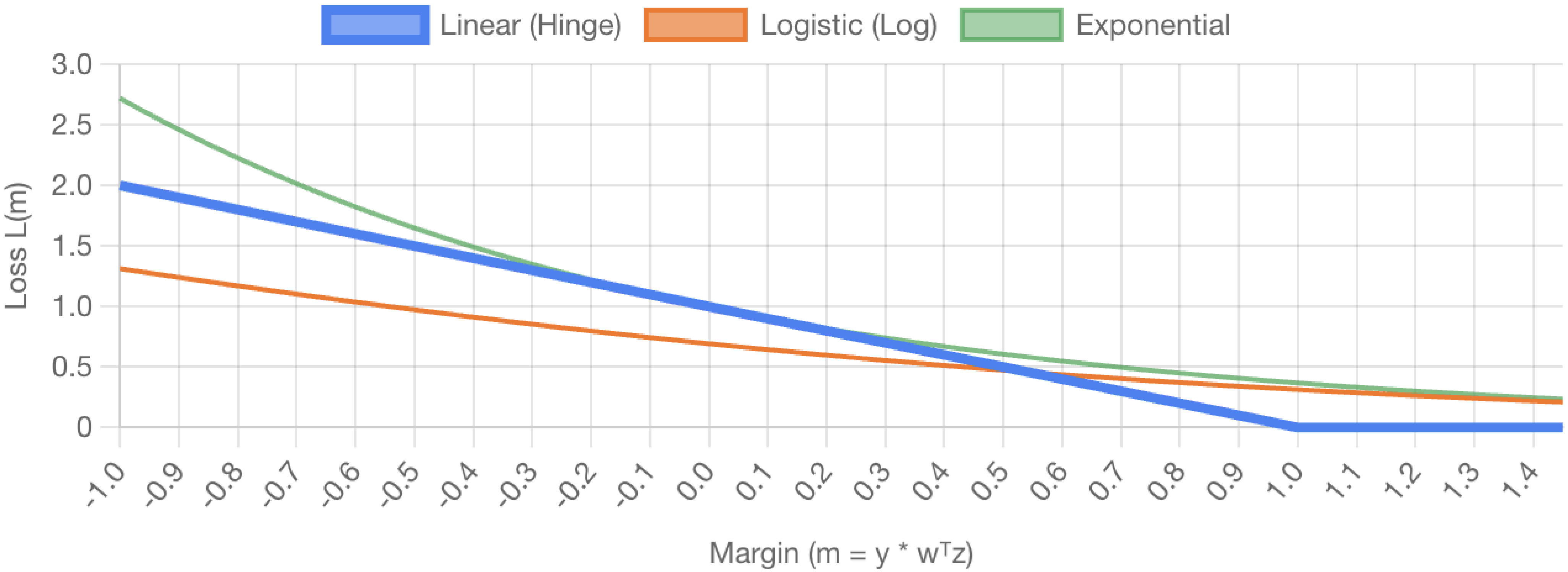
Hinge Loss	Logistic Loss	Exponential Loss
$L(m) = \max(0, 1 - m)$	$L(m) = \log(1 + e^{-m})$	$L(m) = e^{-m}$

we need to penalize when m is very negative or −m is big

If **m is positive**, it means the model's prediction had the same sign as the true label. **The classification was correct.** A larger positive m means it was classified correctly and is far from the decision boundary (a confident, correct prediction).

If **m is negative**, it means the model's prediction had the opposite sign of the true label. **The classification was incorrect.** A more negative m means it was a very confident, but wrong, prediction.

How to get such classifier?



$z = f_{\theta}(x), x$ is input data, y is its label 1 or -1

The behavior

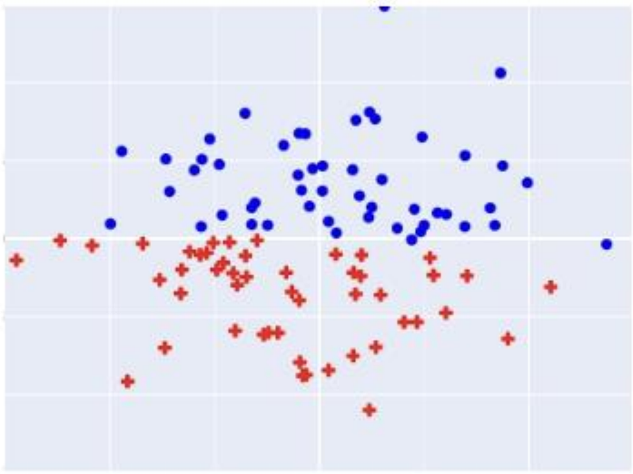
Hinge Loss

The penalty for a wrong prediction grows proportionally to how wrong it is. This approach is robust and less sensitive to outliers.

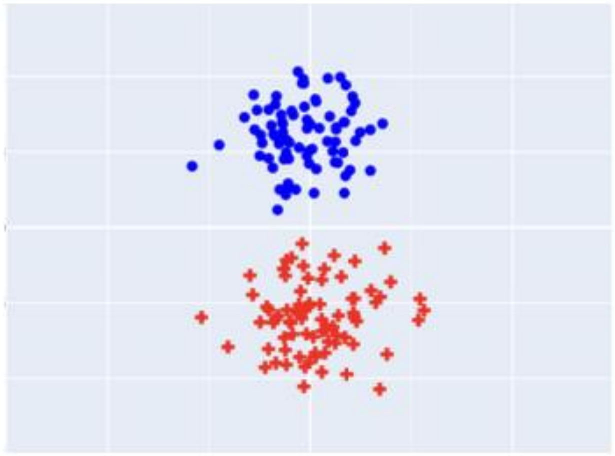
Logistic Loss

The penalty for a wrong prediction grows exponentially. This makes the model extremely sensitive to errors, especially confident mistakes

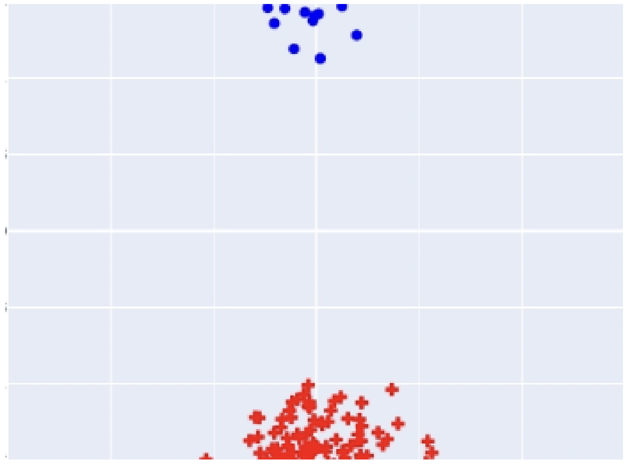
Examples



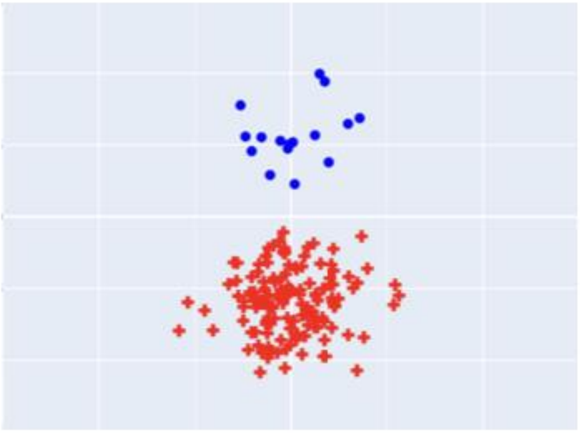
(a)



(b)



(c)



(d)

Loss function

$$L(w) = \sum \max(0, 1 - y_i \cdot w^T x_i) \qquad L(w) = \sum \log(1 + e^{-y_i \cdot w^T x_i})$$

What are we missing here? $\lambda||w||_2$

There will be infinite solutions with out such constraint

Exponential family

Logistic Loss

$$L(w) = \log(1 + e^{-y\cdot(w^T z)})$$

$$\nabla L(w) = \frac{-y}{1 + e^{y(w^T z)}} w$$

[1/2, 1] for all misclassified points

Less prone to the noise

Exponential Loss

$$L(w) = e^{-y\cdot(w^T z)}$$

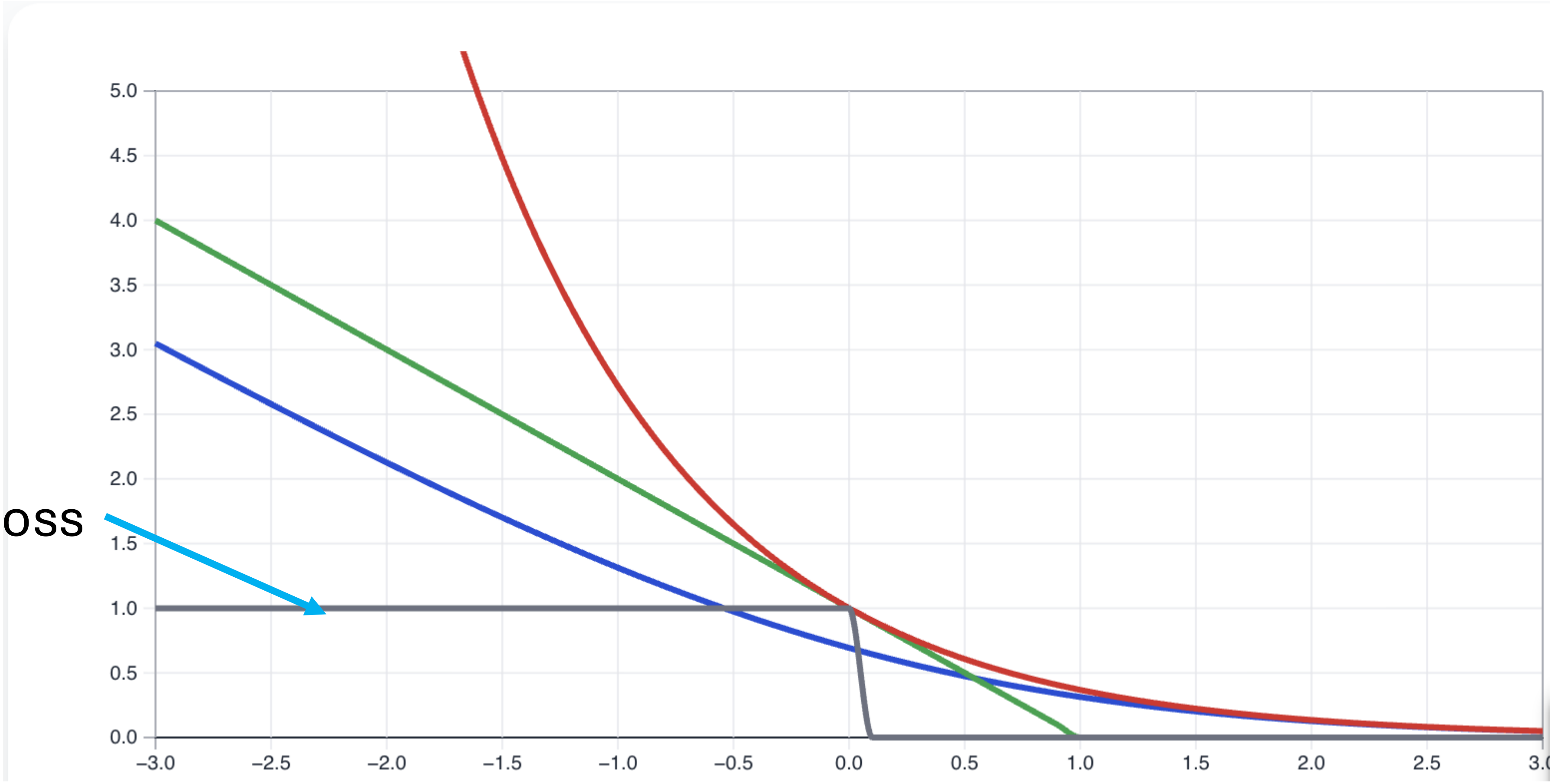
$$\nabla L(w) = -ye^{-y(w^T z)} w$$

Exploding gradients for outliers, very unstable

Very sensitive

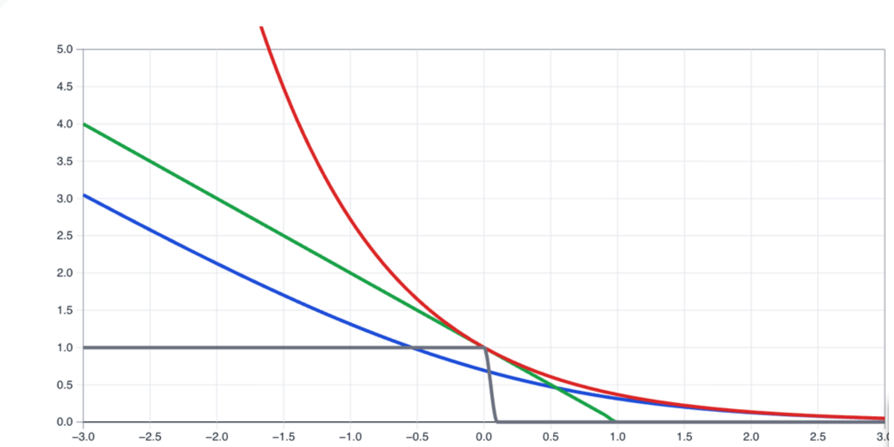
Is the logistic loss convex and does it have a unique solution that gradient descent converges to?

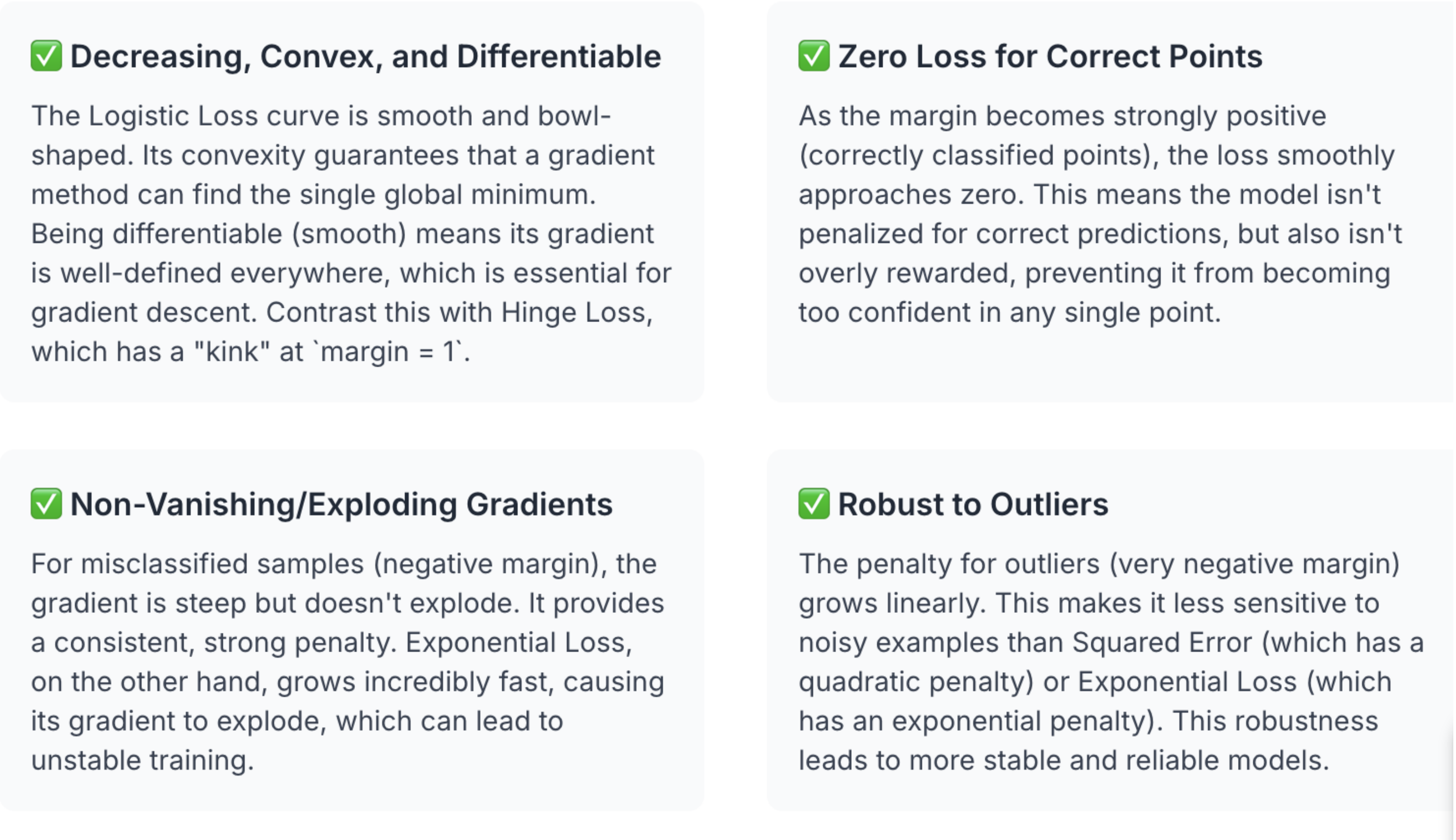
Summarize



Ideal Loss

Summarize





✅ Decreasing, Convex, and Differentiable

The Logistic Loss curve is smooth and bowl-shaped. Its convexity guarantees that a gradient method can find the single global minimum. Being differentiable (smooth) means its gradient is well-defined everywhere, which is essential for gradient descent. Contrast this with Hinge Loss, which has a "kink" at `margin = 1`.

✅ Zero Loss for Correct Points

As the margin becomes strongly positive (correctly classified points), the loss smoothly approaches zero. This means the model isn't penalized for correct predictions, but also isn't overly rewarded, preventing it from becoming too confident in any single point.

✅ Non-Vanishing/Exploding Gradients

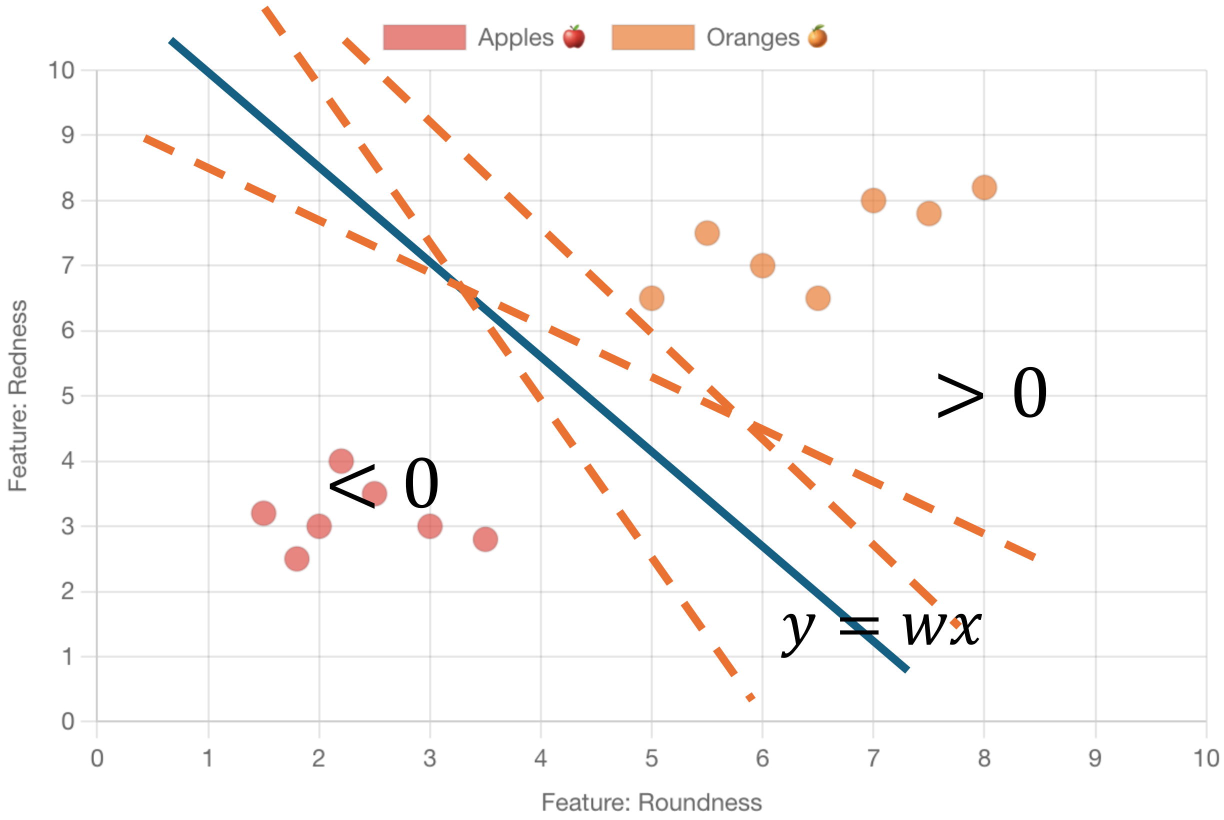
For misclassified samples (negative margin), the gradient is steep but doesn't explode. It provides a consistent, strong penalty. Exponential Loss, on the other hand, grows incredibly fast, causing its gradient to explode, which can lead to unstable training.

✅ Robust to Outliers

The penalty for outliers (very negative margin) grows linearly. This makes it less sensitive to noisy examples than Squared Error (which has a quadratic penalty) or Exponential Loss (which has an exponential penalty). This robustness leads to more stable and reliable models.

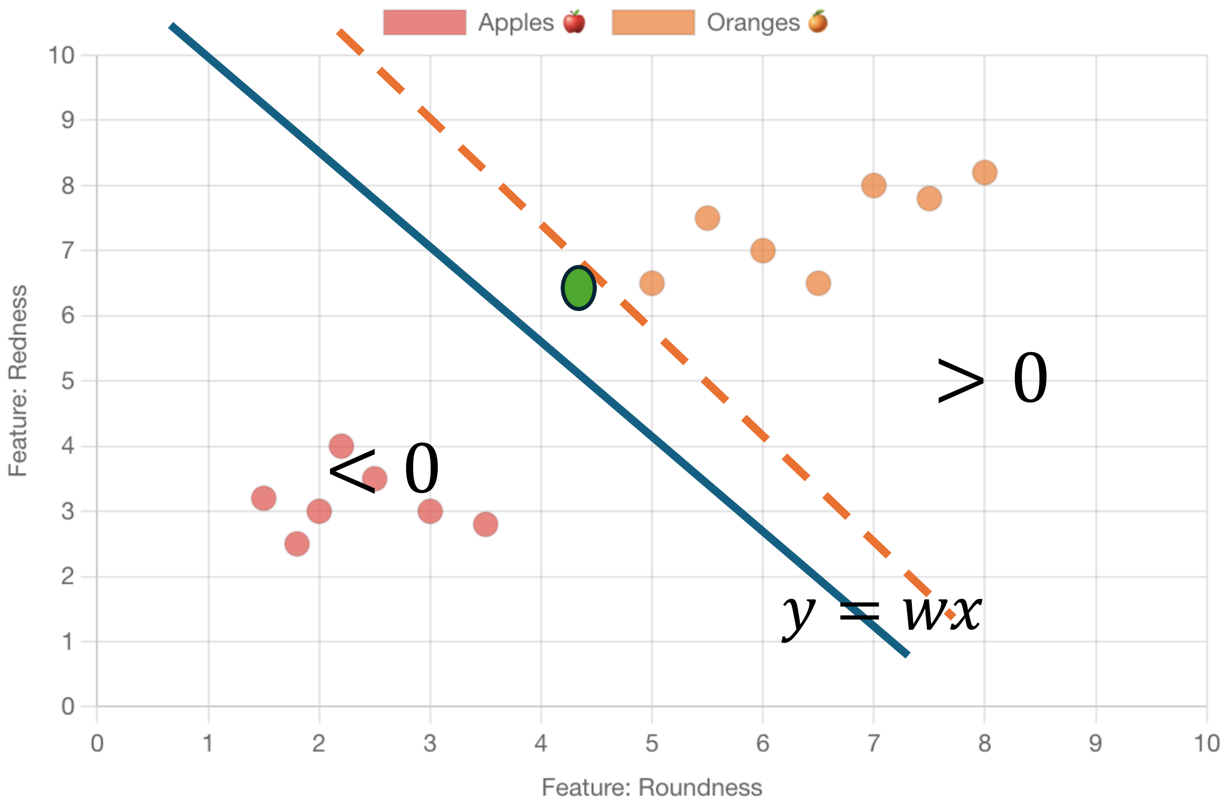
Can we make linear classifier better?

This chart plots our fruit based on two features: "redness" and "roundness". Apples (🍎) and oranges (🍊) form distinct clusters. Our goal is to find a line that separates them. You can then add a new fruit by clicking on the chart to see how it gets classified!



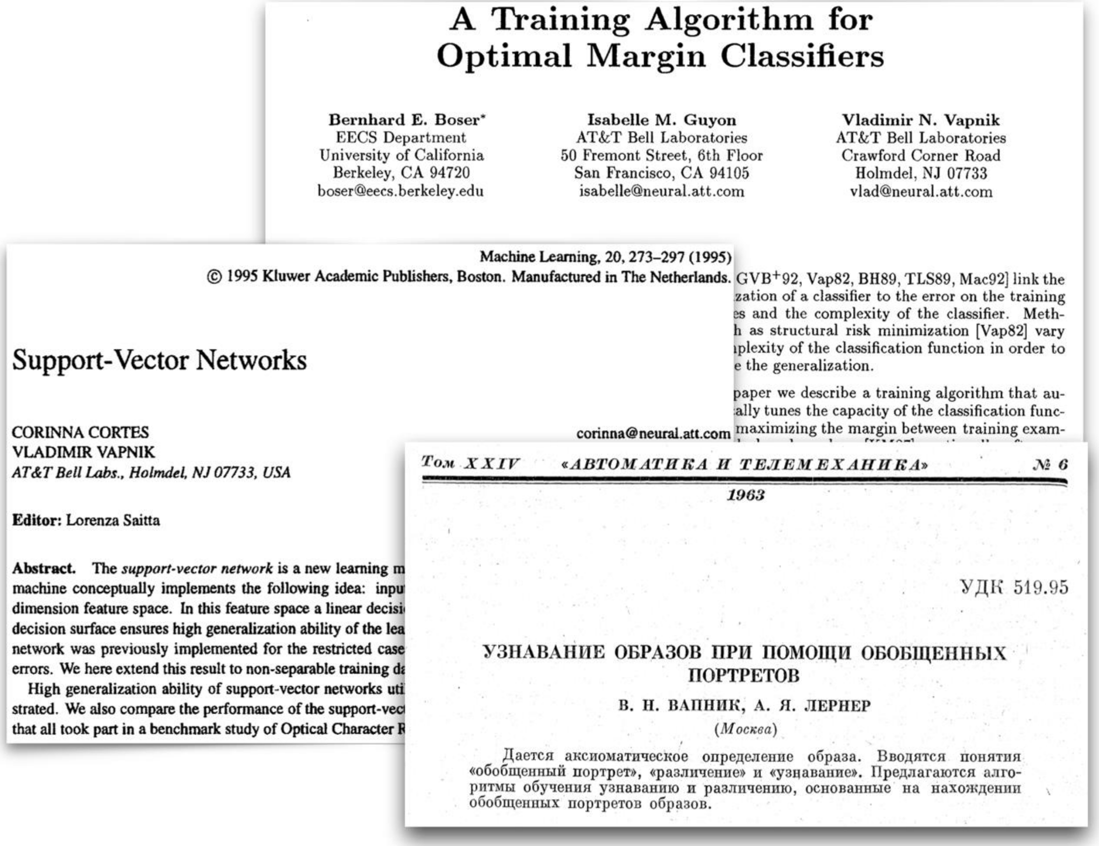
Can we make linear classifier better?

This chart plots our fruit based on two features: "redness" and "roundness". Apples (🍎) and oranges (🍊) form distinct clusters. Our goal is to find a line that separates them. You can then add a new fruit by clicking on the chart to see how it gets classified!



Support Vector Machine (SVM)

Vapnik’s invention



A Training Algorithm for Optimal Margin Classifiers

Bernhard E. Boser*
EECS Department
University of California
Berkeley, CA 94720
boser@eecs.berkeley.edu

Isabelle M. Guyon
AT&T Bell Laboratories
50 Fremont Street, 6th Floor
San Francisco, CA 94105
isabelle@neural.att.com

Vladimir N. Vapnik
AT&T Bell Laboratories
Crawford Corner Road
Holmdel, NJ 07733
vlad@neural.att.com

GVB+92, Vap82, BH89, TLS89, Mac92] link the zation of a classifier to the error on the training es and the complexity of the classifier. Meth- h as structural risk minimization [Vap82] vary plexity of the classification function in order to e the generalization.

paper we describe a training algorithm that au- ally tunes the capacity of the classification func- maximizing the margin between training exam-

Machine Learning, 20, 273–297 (1995)
© 1995 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Support-Vector Networks

CORINNA CORTES corinna@neural.att.com
VLADIMIR VAPNIK
AT&T Bell Labs., Holmdel, NJ 07733, USA

Editor: Lorenza Saitta

Abstract. The *support-vector network* is a new learning m machine conceptually implements the following idea: inpu dimension feature space. In this feature space a linear decisi decision surface ensures high generalization ability of the lea network was previously implemented for the restricted case errors. We here extend this result to non-separable training d High generalization ability of support-vector networks uti strated. We also compare the performance of the support-vec that all took part in a benchmark study of Optical Character R

Том XXIV «АВТОМАТИКА И ТЕЛЕМЕХАНИКА» № 6
1963

УДК 519.95

УЗНАВАНИЕ ОБРАЗОВ ПРИ ПОМОЩИ ОБОБЩЕННЫХ ПОРТРЕТОВ

В. Н. ВАПНИК, А. Я. ЛЕРНЕР

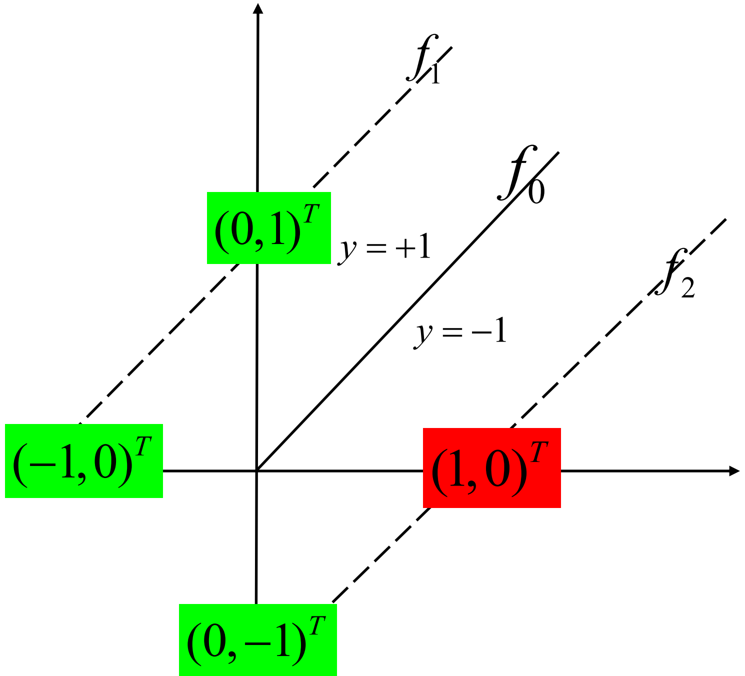
(*Москва*)

Дается аксиоматическое определение образа. Вводятся понятия «обобщенный портрет», «различение» и «узнавание». Предлагаются алгоритмы обучения узнаванию и различению, основанные на нахождении обобщенных портретов образов.

SVM

$$f(x) = w^T \mathbf{x} + b$$

$$y = \operatorname{sgn}\left(f\left(x\right)\right) = \begin{matrix} f(x) = w^T \mathbf{x}_i + b > 0 & \text{for } y = +1 \\ f(x) = w^T \mathbf{x}_i + b < 0 & \text{for } y = -1 \end{matrix}$$



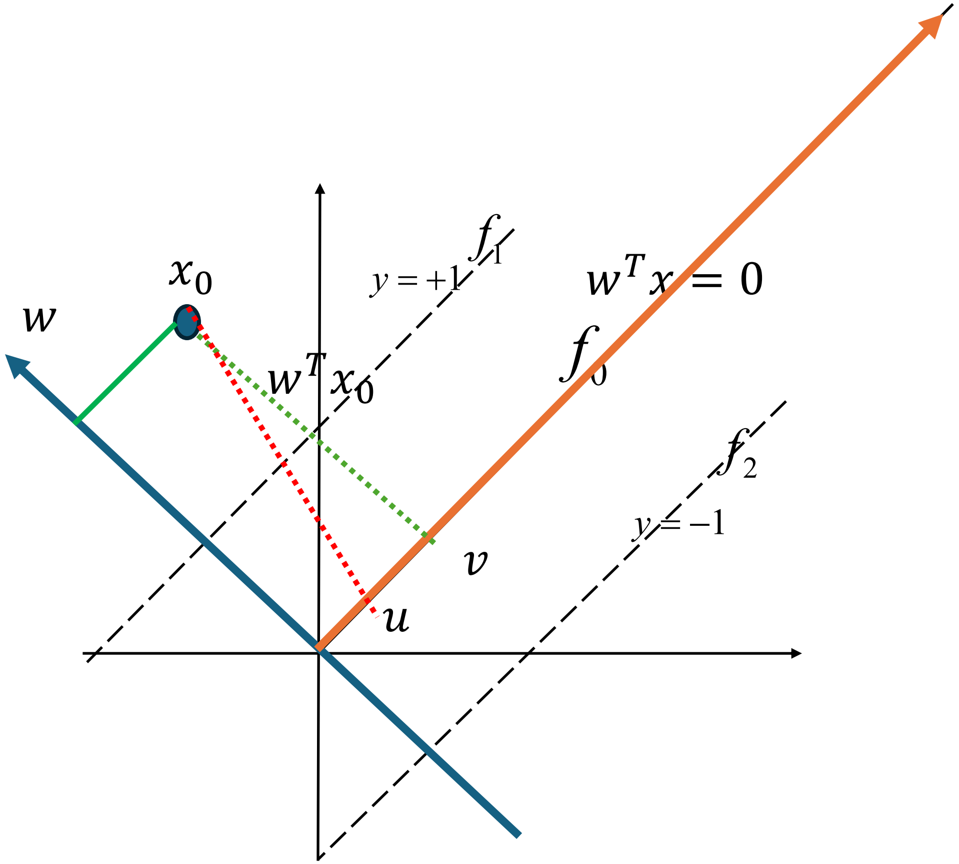
$$w^T = (-1,1); b = 0$$

$$f_0(x) = (-1,1)\mathbf{x} = 0$$

$$f_1(x) = (-1,1)\mathbf{x} - 1 = 0$$

$$f_2(x) = (-1,1)\mathbf{x} + 1 = 0$$

Shortest distance



$$|w|_2 = 1$$

$$\min_{w^Tu=0} |x_0 - u|^2 = |v-u|^2 + w^Tx_0 \geq w^Tx_0$$

Max Margin

The best hyperplane is the one that is as far as possible from the data points of both classes

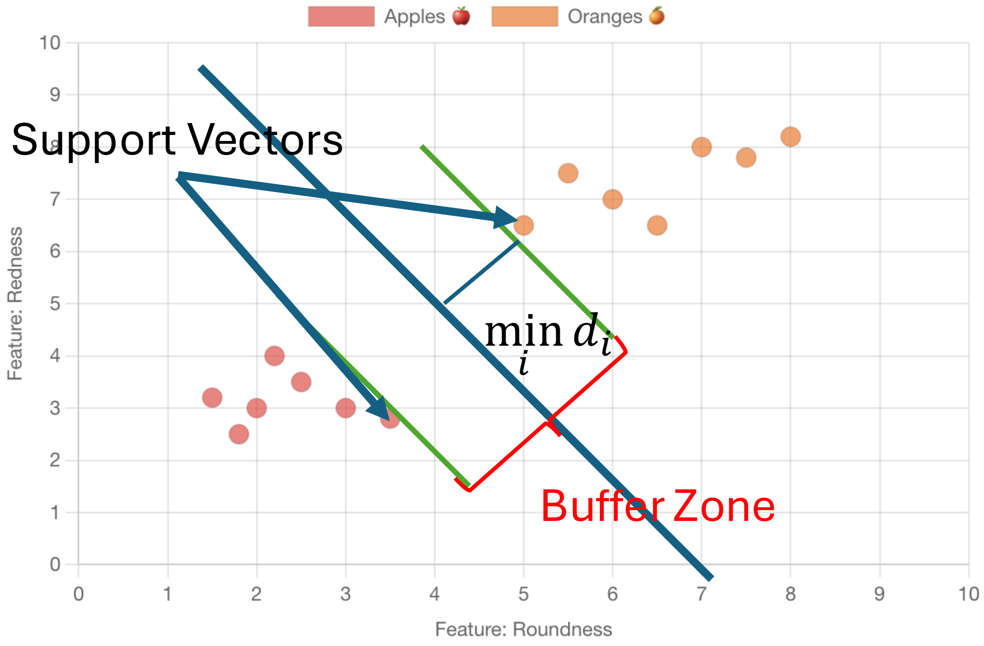
But how to formulate to find such hyperplane?

The distance of each point x_i to any hyperplane w can be written as: $d_i = y_i \cdot (w^T\ x_i)$

$$w_{MM} = argmax_{|w|_2=1} \min_i d_i$$

It maximize the minimum distance to any point

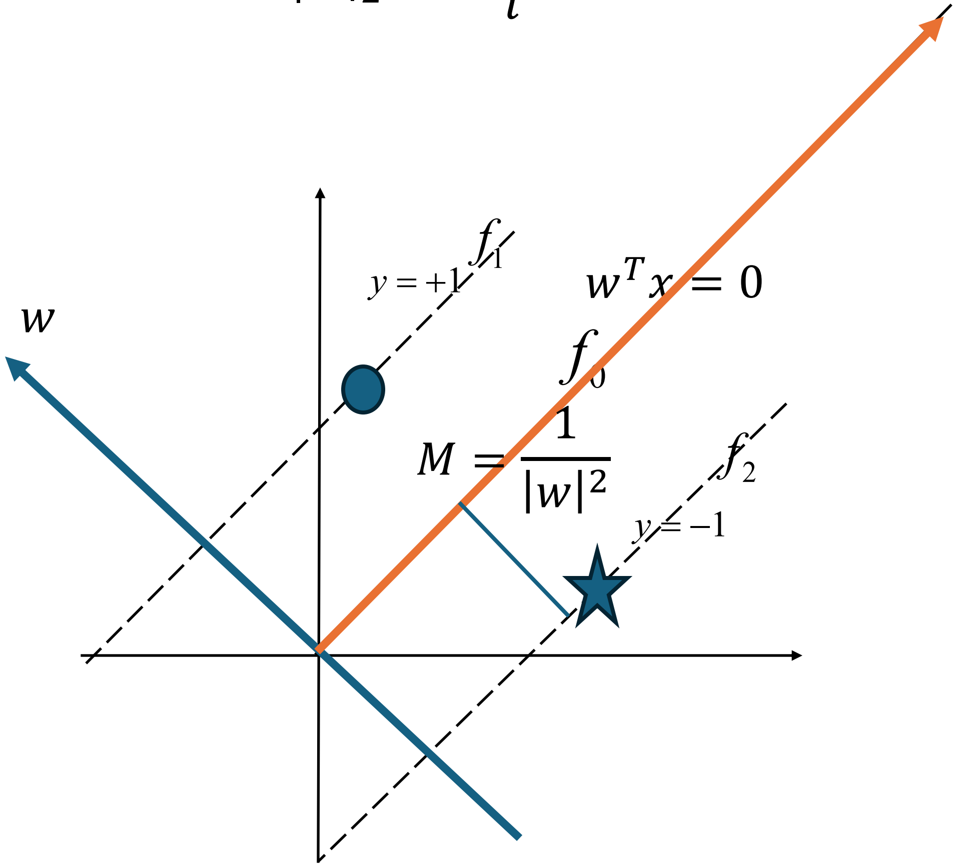
This chart plots our fruit based on two features: "redness" and "roundness". Apples (🍎) and oranges (🍊) form distinct clusters. Our goal is to find a line that separates them. You can then add a new fruit by clicking on the chart to see how it gets classified!



Reformulate

Assume that we have a hyperplane such that it can classify all the points perfectly

$$max_{|w|_2=1} \min_i y_i \cdot (w^T \; x_i)$$



$$\max M, \qquad s.t. \, y_i \cdot \left(< \frac{w}{|w|^2}, x > \right) \geq M$$

$$M = \frac{1}{|w|^2}$$

$$\max \frac{1}{|w|^2}, \qquad s.t. \, y_i \cdot \left(< \frac{w}{|w|^2}, x > \right) \geq \frac{1}{|w|^2}$$

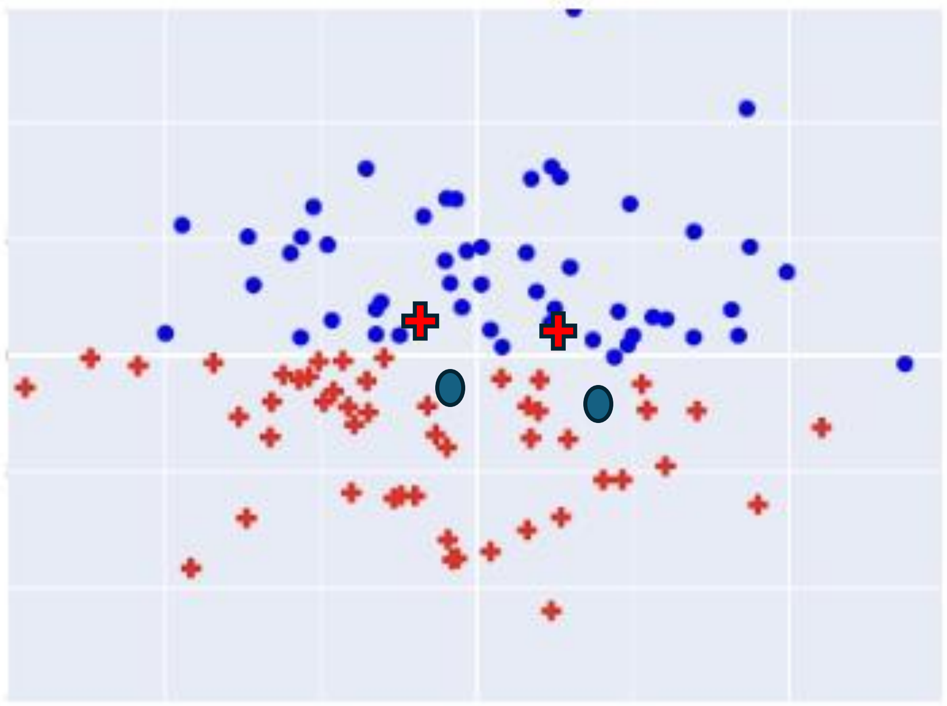
$$\min |w|^2, \qquad s.t. \, y_i \cdot (w^T x_i) \geq 1$$

The real case

Hard-SVM $\min|w|^2, \quad s.t.\, y_i \cdot (w^T x_i) \geq 1$

Soft-SVM $\min|w|^2 + \lambda \sum_i^n \xi_i\,,$

$s.t.\, y_i \cdot (w^T x_i) \geq 1 - \xi_i, \quad \xi_i > 0$



Intuition: maximize the margin while allowing some constraints to be violated

How: introduce slack variables ξ_i, relax the constrain

Soft SVM

$$\min|w|^2 + \lambda \sum_{i}^{n} \xi_i \,,$$
$$s.t.\, y_i \cdot (w^T x_i) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

For all the point x_i, if it does not violate the rule, the tolerance $\xi_i \ = 0$

For all the point x_i, if it in between the margin, the tolerance $0 < \xi_i \ < 1$

For all the point x_i, if it in cross the margin, the tolerance $\xi_i \ > 1$

Violate the rule

$$\xi_i \ = \max(0, 1 - y_i \cdot (w^T x_i))$$

Ignore the ones that in the opposite direction, but try to minimize the distance that the points lying in between the hyperplane and margins

Soft SVM

$$\mathrm{m}in|w|^2 + \lambda \sum_{i}^{n} \xi_i\,, \qquad s.t.\, y_i \cdot (w^T x_i) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

⬇

$$\mathrm{min}|w|^2 + \lambda \sum_{i}^{n} \xi_i - \sum_{i}^{n} \alpha_i\ [y_i \cdot (w^T x_i) - 1 + \xi_i] - \sum \mu_i \xi_i$$

How to solve it?

Setting: $(X, Y) \sim D$, where $y \in \{-1, 1\}$

Goal: find a classifier $f: R^n \ \rightarrow \{-1,1\}$, that minimizes the risk

$$L(f) = E_D \ \ (1_{Y \neq f(x)})$$

$$\min_{w} \ \frac{1}{2}|w|^2 + \lambda \sum_{i}^{n} \xi_i - \sum_{i}^{n} \alpha_i \ [y_i \cdot (w^T x_i) - 1 + \xi_i] - \sum \mu_i \xi_i$$

Setting up the Lagrangian

We use Lagrange multipliers $\alpha_i \geq 0$ and $\mu_i \geq 0$.

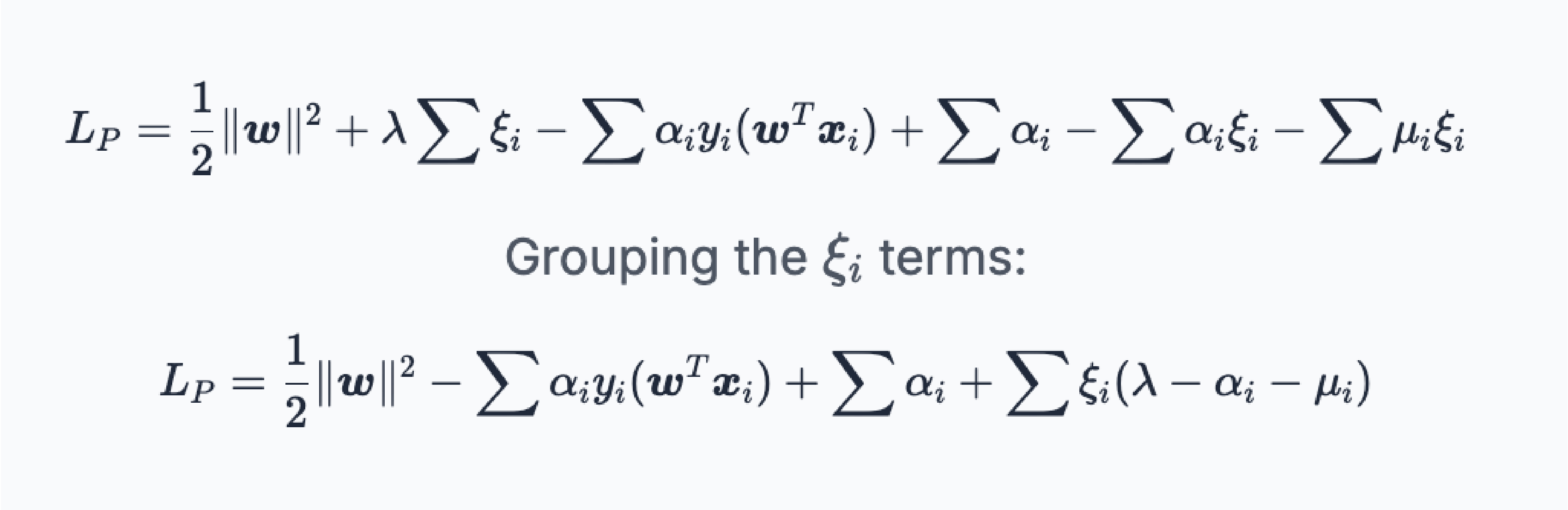
$$L_P = \frac{1}{2}\|\boldsymbol{w}\|^2 + \lambda \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i [y_i(\boldsymbol{w}^T \boldsymbol{x}_i) - 1 + \xi_i] - \sum_{i=1}^{N} \mu_i \xi_i$$

Our goal is to minimize L with respect to the primal variables ($\boldsymbol{w}$, $\boldsymbol{\xi}$).

1. $\frac{\partial L_P}{\partial \boldsymbol{w}} = 0 \implies \boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i$
2. $\frac{\partial L_P}{\partial \xi_i} = 0 \implies \lambda - \alpha_i - \mu_i = 0$

$\alpha_i \in [0, \lambda]$

Rewrite the primal problem



$$L_P = \frac{1}{2}\|\boldsymbol{w}\|^2 + \lambda \sum \xi_i - \sum \alpha_i y_i(\boldsymbol{w}^T \boldsymbol{x}_i) + \sum \alpha_i - \sum \alpha_i \xi_i - \sum \mu_i \xi_i$$

Grouping the ξ_i terms:

$$L_P = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum \alpha_i y_i(\boldsymbol{w}^T \boldsymbol{x}_i) + \sum \alpha_i + \sum \xi_i(\lambda - \alpha_i - \mu_i)$$

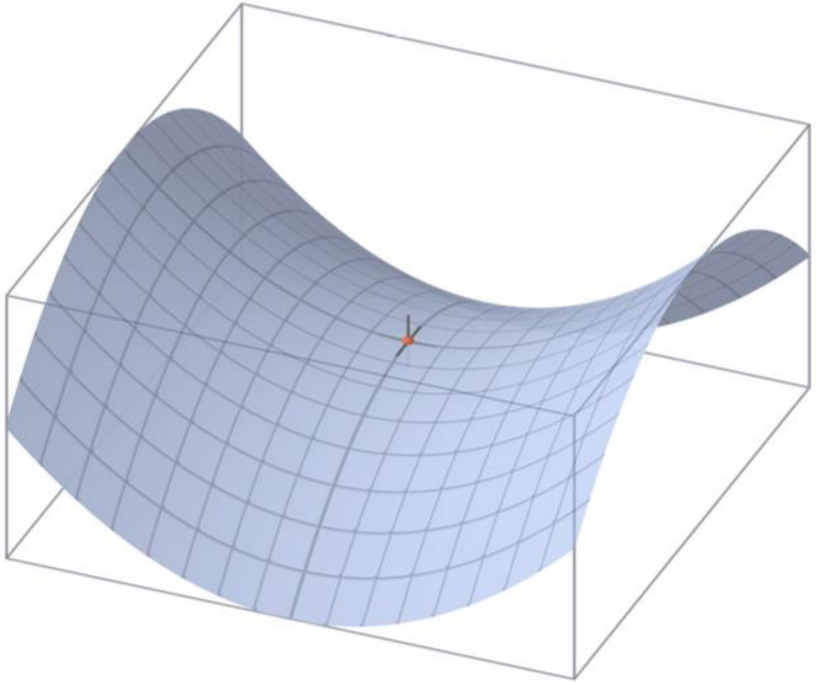
Duality

$$L_P = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum \alpha_i y_i(\boldsymbol{w}^T\boldsymbol{x}_i) + \sum \alpha_i + \sum \xi_i\underbrace{(\lambda - \alpha_i - \mu_i)}_{\text{This is zero!}}$$

$$G(w,\alpha) = \min_{w}\max_{\alpha}\frac{1}{2}||w||^2 - \sum\left(y_i \cdot \left(\boldsymbol{w^T x_i}\right) - 1\right)\alpha_i$$

$$\max_{\alpha}\min_{w} G(w,\alpha) \leq \min_{w}\max_{\alpha} G(w,\alpha)$$

$$\max_{\alpha} W(\alpha) \leq \min_{w} L_P(w)$$



Because the function G is convex with respect to w and concave with respect to α

Rewrite the primal problem

$$L_P = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum \alpha_i y_i (\boldsymbol{w}^T \boldsymbol{x}_i) + \sum \alpha_i + \sum \xi_i \underbrace{(\lambda - \alpha_i - \mu_i)}_{\text{This is zero!}}$$

$$\sum \alpha_i y_i (\boldsymbol{w}^T \boldsymbol{x}_i) = \boldsymbol{w}^T \left(\sum \alpha_i y_i \boldsymbol{x}_i\right)$$

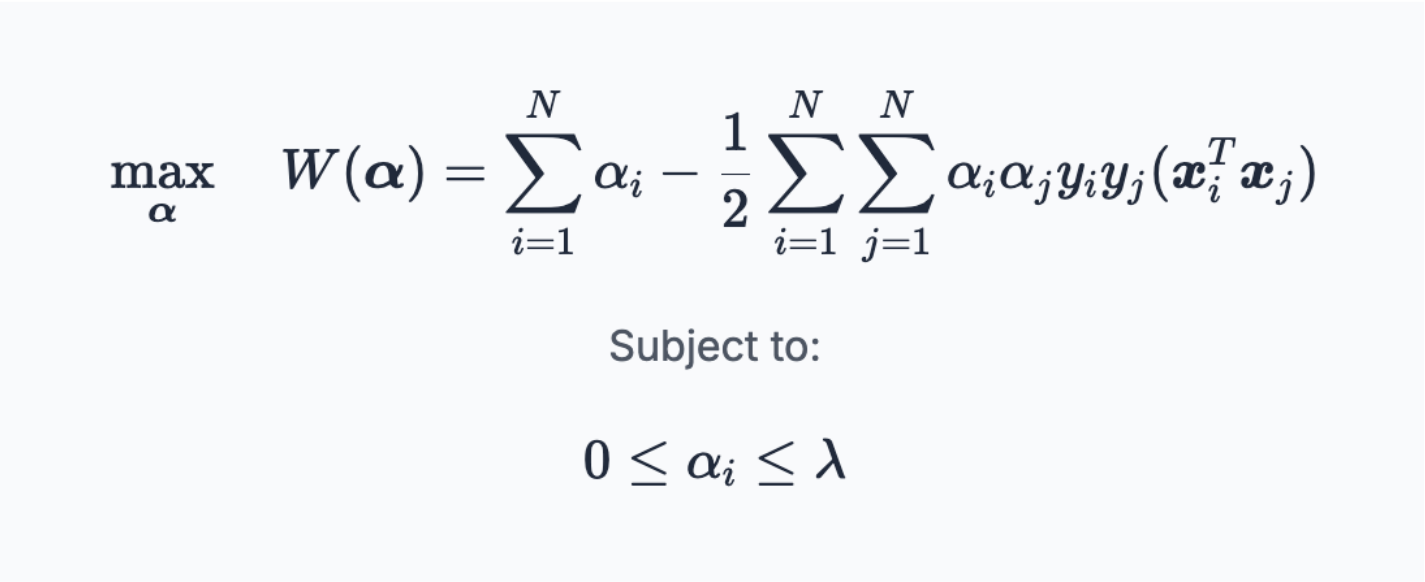
Recall: $\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i$

$$L_P = \tfrac{1}{2}\|\boldsymbol{w}\|^2 - \|\boldsymbol{w}\|^2 + \sum \alpha_i = \sum \alpha_i - \tfrac{1}{2}\|\boldsymbol{w}\|^2$$

$$\begin{aligned}\|\boldsymbol{w}\|^2 &= \left(\sum_i \alpha_i y_i \boldsymbol{x}_i\right)^T \left(\sum_j \alpha_j y_j \boldsymbol{x}_j\right)\\ &= \sum_i \sum_j \alpha_i \alpha_j y_i y_j (\boldsymbol{x}_i^T \boldsymbol{x}_j)\end{aligned}$$

The Dual Problem

Substituting back gives the dual, which is easier to solve.



$$\max_{\boldsymbol{\alpha}} \quad W(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\boldsymbol{x}_i^T \boldsymbol{x}_j)$$

Subject to:

$$0 \leq \alpha_i \leq \lambda$$

$$W(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^{\boldsymbol{T}} \mathbf{1} - \frac{1}{2} \boldsymbol{\alpha}^{\boldsymbol{T}} \boldsymbol{Y} \boldsymbol{X} \boldsymbol{X}^{\boldsymbol{T}} \boldsymbol{Y} \boldsymbol{\alpha}$$

$\boldsymbol{\alpha}^{\boldsymbol{T}}$ is a very sparse vector

Differentiable Concave function

$X \in R^{n \times d}, XX^T \in R^{n \times n}$, no dependency on d for Kernel Matrix

Can we solve it analytically?

1. Constrained Optimization problem
2. Inequality Constraints: box constraints can be tricky
3. High Dimensionality: Reverse matrix will be O(N^3)
4. Quadratic Programming: Well studied, no simple solution

The final solution

$$[\text{Data}, \lambda] \rightarrow \boxed{\text{QP Solver (e.g., SMO)}} \rightarrow \boldsymbol{\alpha}^* = [\alpha_1^*, \ldots, \alpha_N^*]$$

Once we have this resulting vector $\boldsymbol{\alpha}^*$, we use it to get $\boldsymbol{w}^*$:

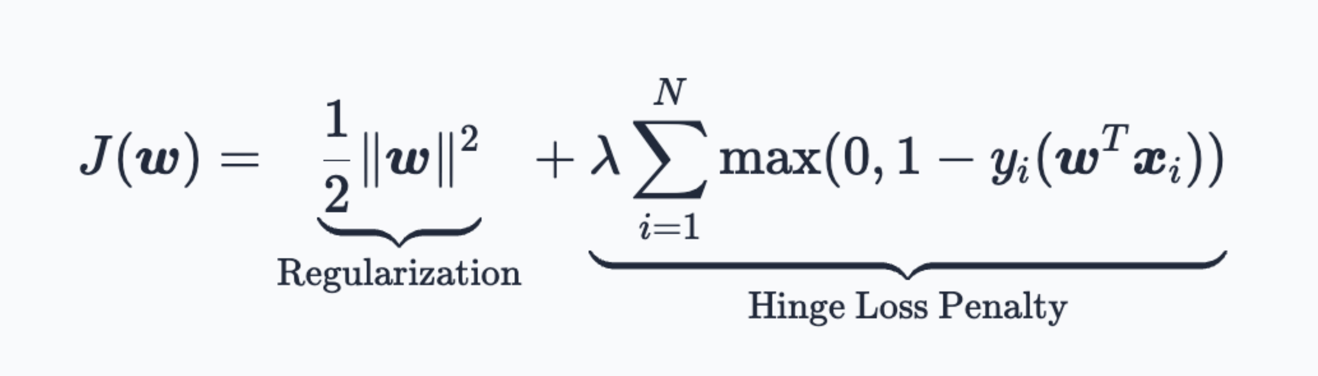
$$\boldsymbol{w}^* = \sum_{i=1}^{N} \alpha_i^* y_i \boldsymbol{x}_i$$

Can you solve it by using the optimization methods we learned in last section?

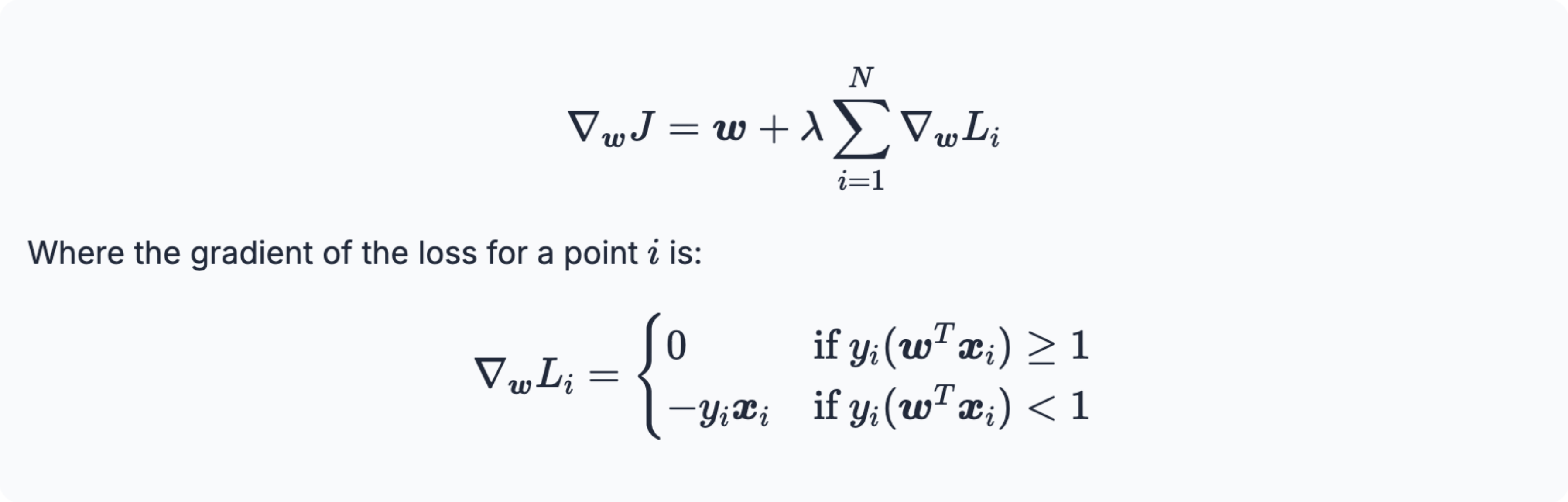
Only the support vectors (where $\alpha_i^* > 0$) contribute to the sum!

Let’s re-interpret it with GD:

$$\min |w|^2 + \lambda \sum^{n} \xi_i , \qquad s.t.\, y_i \cdot (w^T x_i) \geq 1 - \xi_i, \quad \xi_i \geq 0$$



$$J(\boldsymbol{w}) = \underbrace{\frac{1}{2}\|\boldsymbol{w}\|^2}_{\text{Regularization}} + \underbrace{\lambda \sum_{i=1}^{N} \max(0, 1 - y_i(\boldsymbol{w}^T \boldsymbol{x}_i))}_{\text{Hinge Loss Penalty}}$$



$$\nabla_{\boldsymbol{w}} J = \boldsymbol{w} + \lambda \sum_{i=1}^{N} \nabla_{\boldsymbol{w}} L_i$$

Where the gradient of the loss for a point i is:

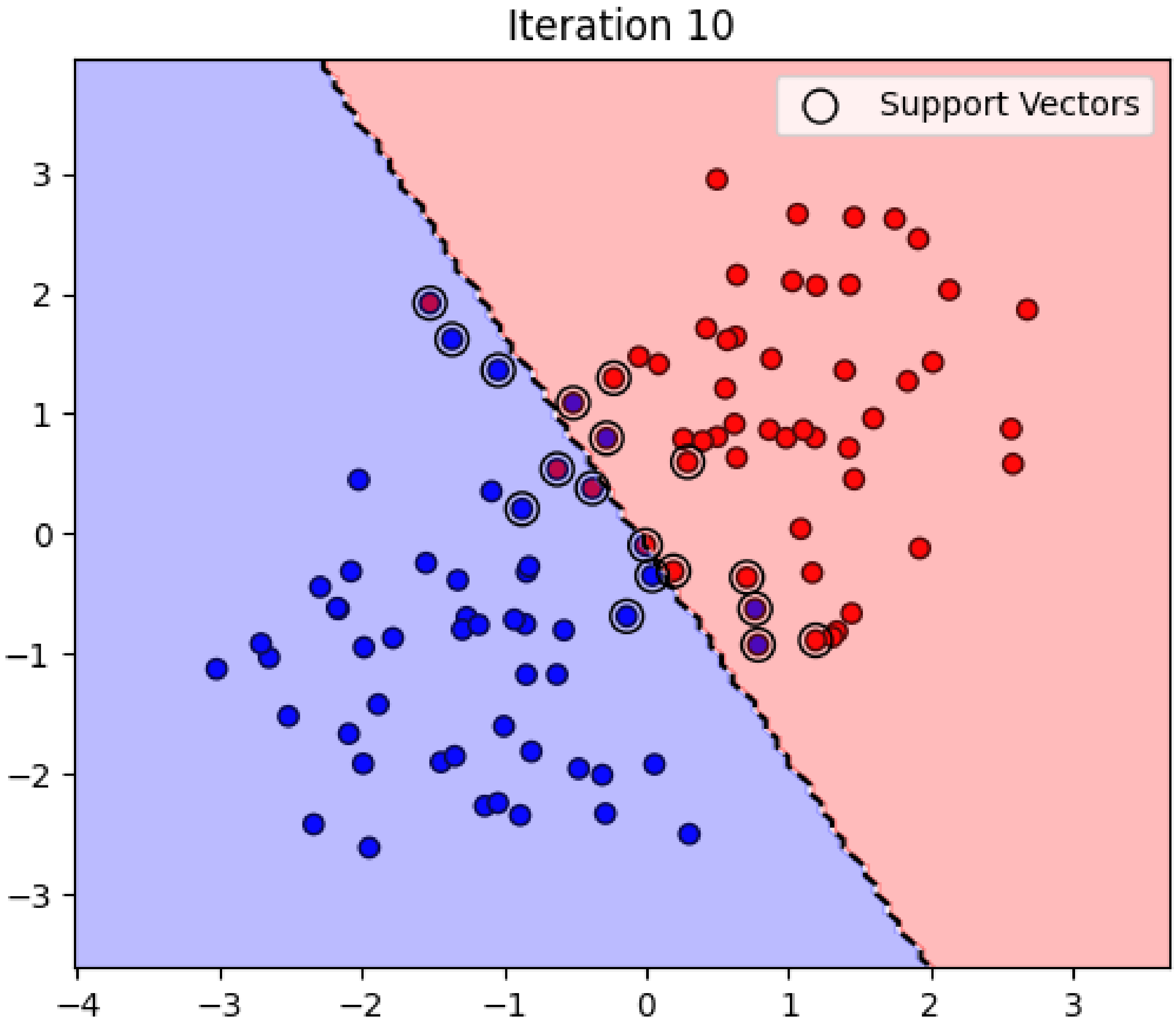
$$\nabla_{\boldsymbol{w}} L_i = \begin{cases} 0 & \text{if } y_i(\boldsymbol{w}^T \boldsymbol{x}_i) \geq 1 \\ -y_i \boldsymbol{x}_i & \text{if } y_i(\boldsymbol{w}^T \boldsymbol{x}_i) < 1 \end{cases}$$

Formulation

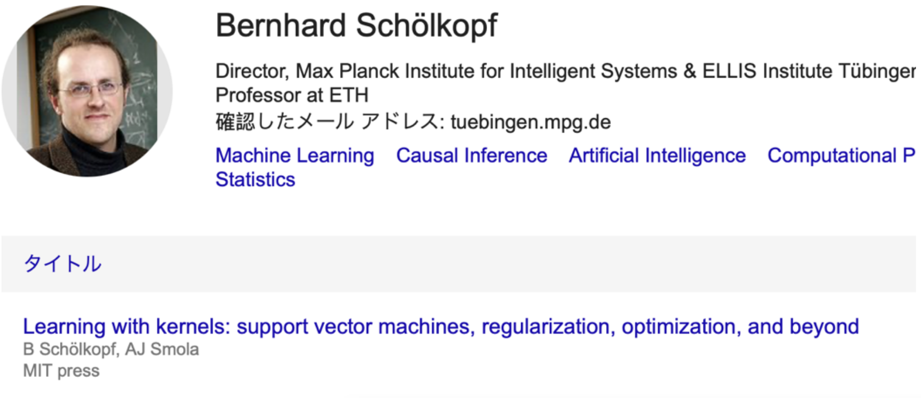
To classify a new (augmented

$$f(z) =$$

The prediction depends only on the do



Kernel Methods



Bernhard Schölkopf

Director, Max Planck Institute for Intelligent Systems & ELLIS Institute Tübinger
Professor at ETH
確認したメール アドレス: tuebingen.mpg.de

Learning with kernels: support vector machines, regularization, optimization, and beyond
B Schölkopf, AJ Smola
MIT press

A tutorial on support vector regre
AJ Smola, B Schölkopf
Statistics and computing 14 (3), 199-22

Nonlinear component analysis a
B Schölkopf, A Smola, KR Müller
Neural computation 10 (5), 1299-1319

Kernel principal component a
B Schölkopf, A Smola, KR Müller
International conference on artificial

Support vector method for no
B Schölkopf, RC Williamson, A Smo
Advances in neural information proc

Deep sets
M Zaheer, S Kottur, S Ravanbakhsh
Advances in neural information proc

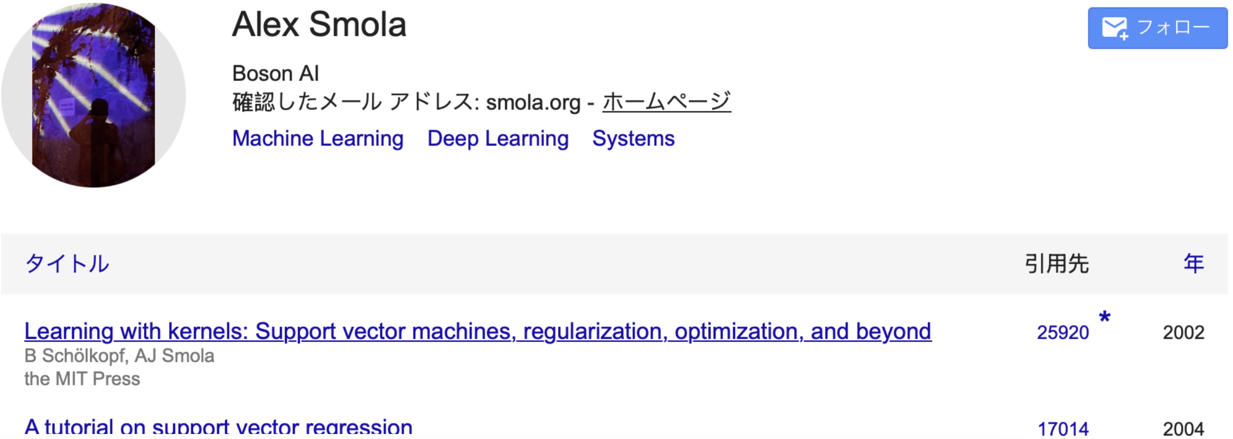
Kernel methods in machine le
T Hofmann, B Schölkopf, AJ Smola

A kernel method for the two-s
A Gretton, K Borgwardt, M Rasch, B
Advances in neural information proc

Communication efficient distri
M Li, DG Andersen, A Smola, K Yu
Advances in neural information proc

Advances in kernel methods:
B Schölkopf, CJC Burges, AJ Smola
MIT press

A generalized representer the
B Schölkopf, R Herbrich, AJ Smola
International conference on computational learning theory, 416-426



Alex Smola

Boson AI
確認したメール アドレス: smola.org - ホームページ

タイトル	引用先	年
Learning with kernels: Support vector machines, regularization, optimization, and beyond B Schölkopf, AJ Smola the MIT Press	25920 *	2002
A tutorial on support vector regression	17014	2004
	11432	1998

📅 **February 2-14, 2003** 📍 Canberra, Australia

📅 **February 11-22, 2002** 📍 Canberra, Australia

municated by John Platt

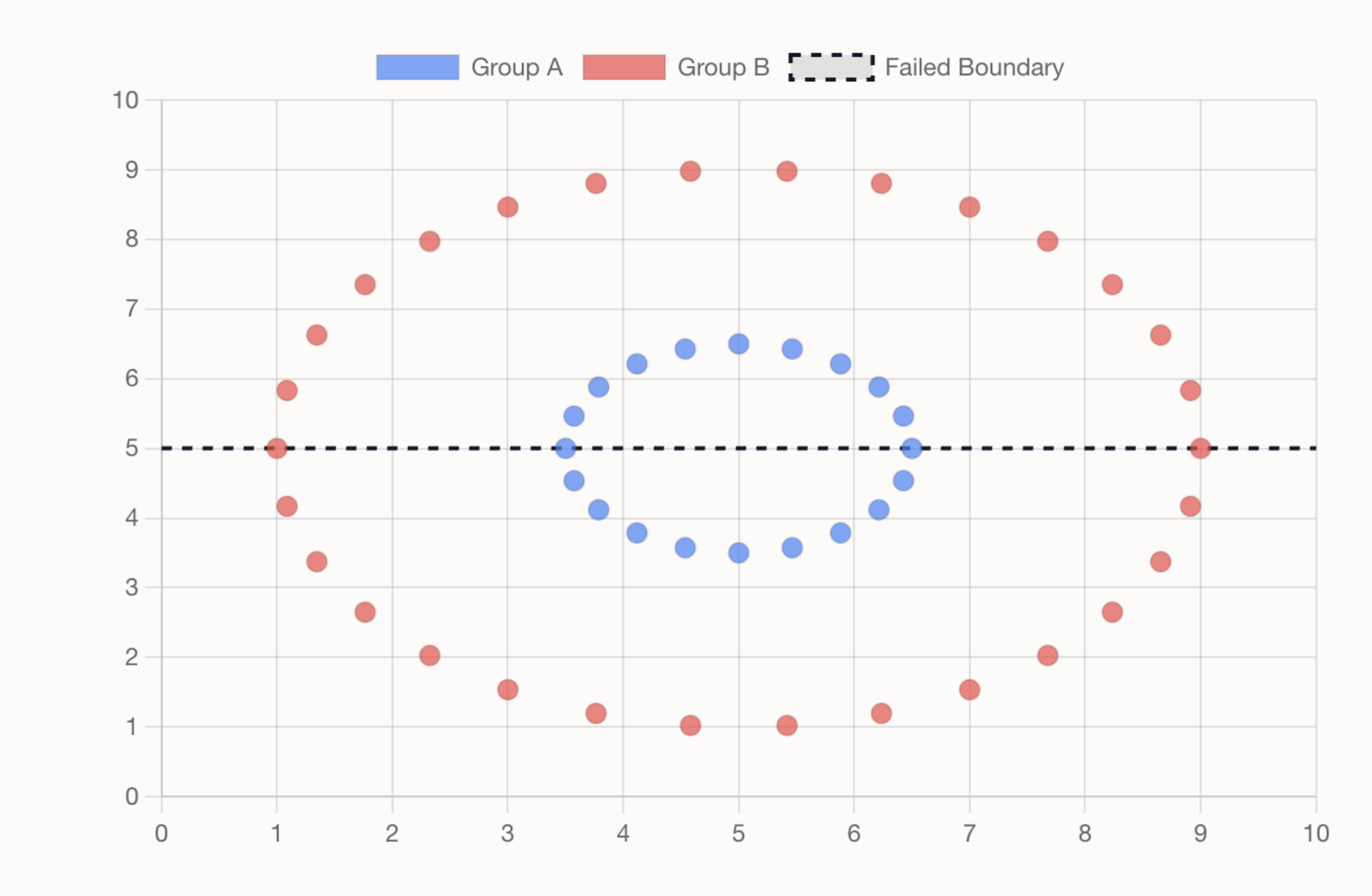
Engineering, Australian

anberra 0200, Australia

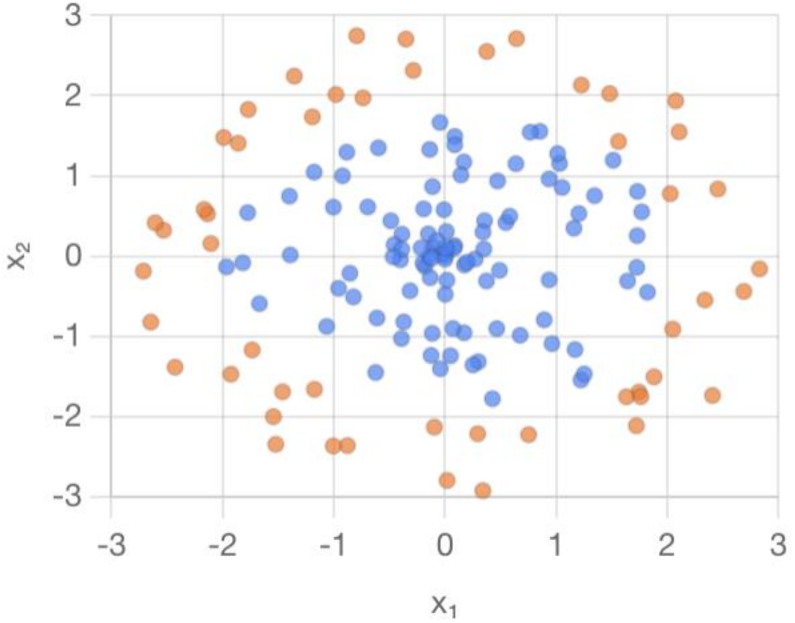
RSISE, Australian National University, Canberra 0200, Australia

MLSS contact: Bernhard Schölkopf, Alex Smola

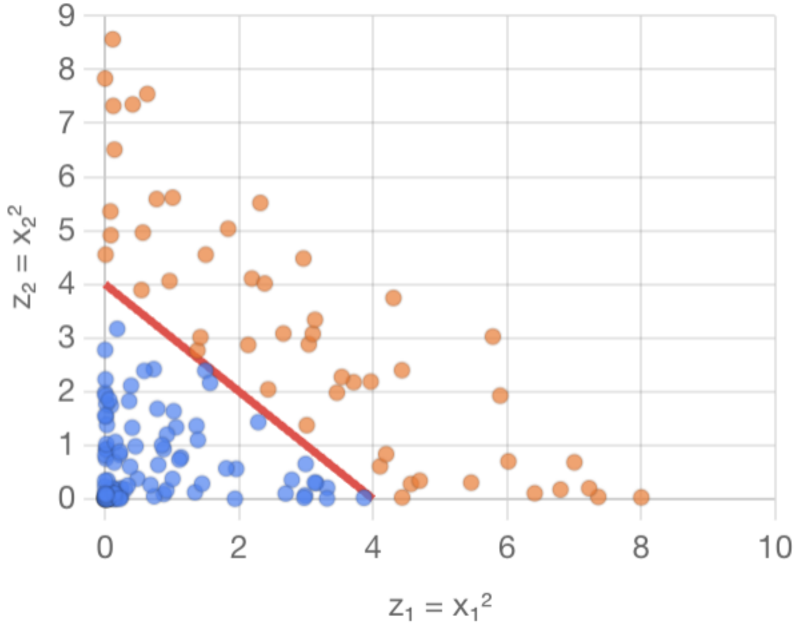
Non-linear case



Feature extraction

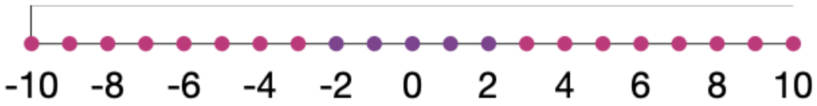


Original 2D Space (x_1, x_2)

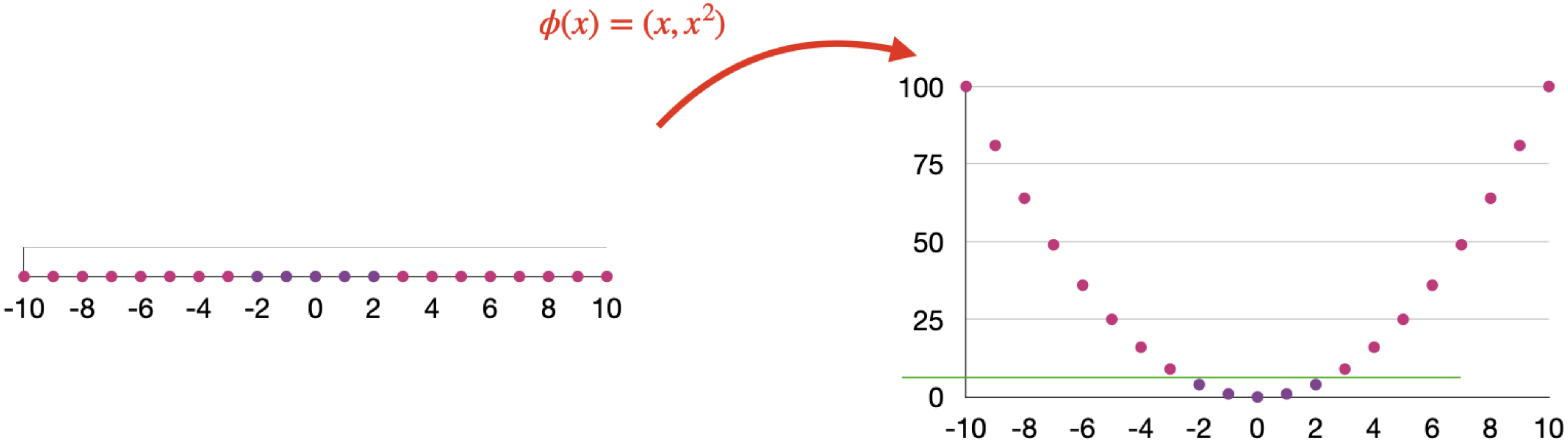


Transformed 2D Space (z_1, z_2)

Kernel tricks

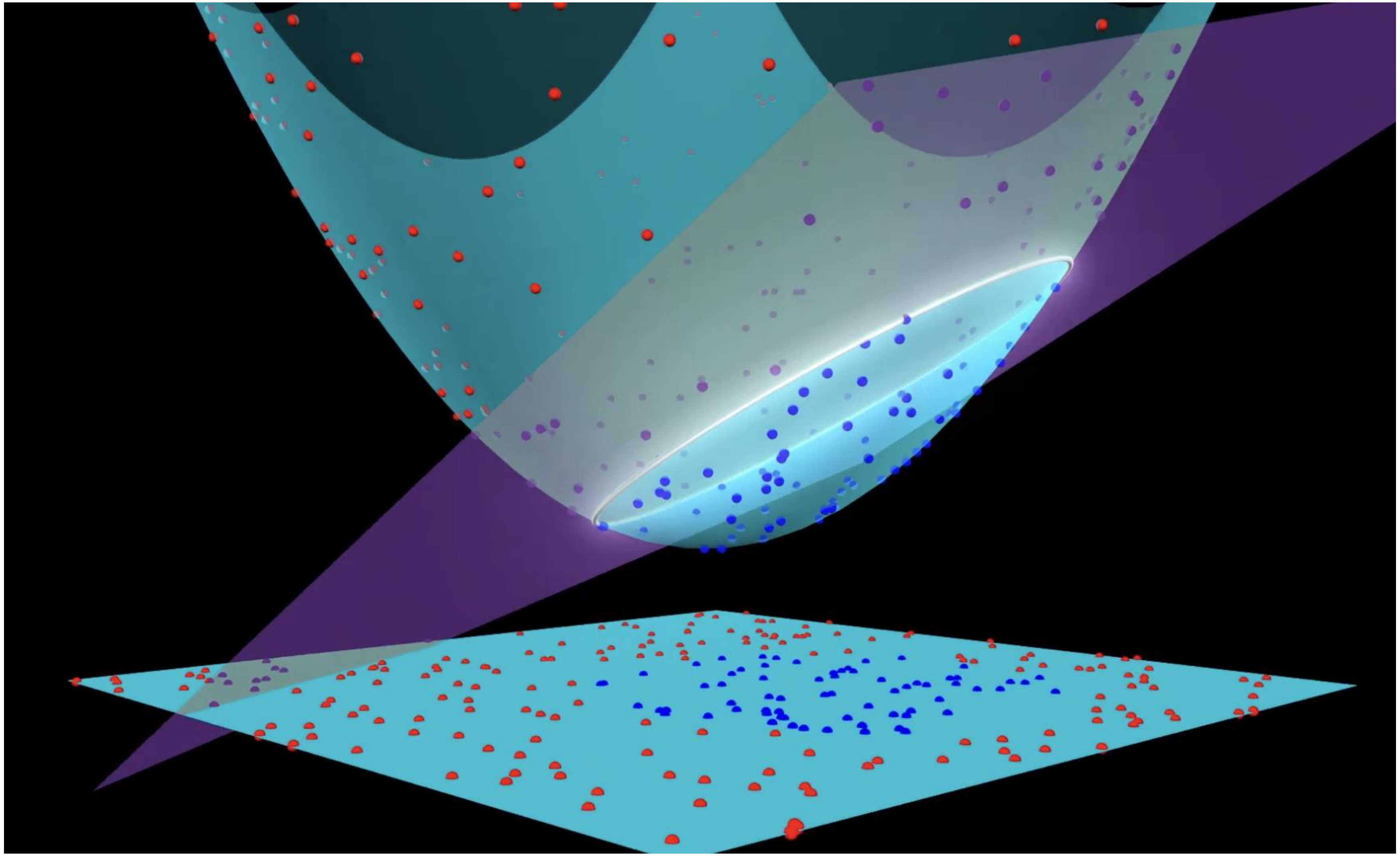


Kernel tricks



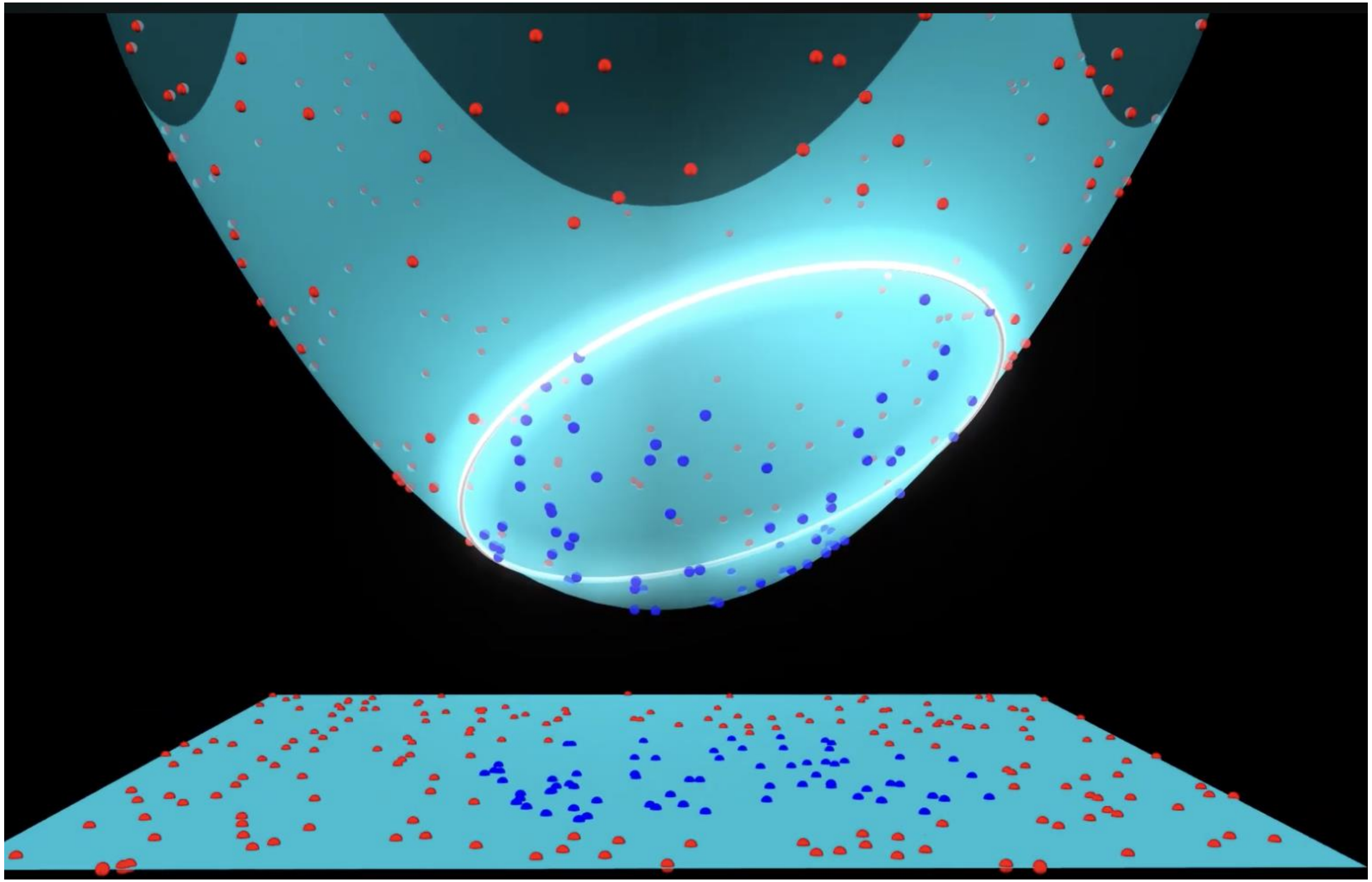
$\phi(x) = (x, x^2)$

Polynomial kernels



https://www.youtube.com/watch?v=OdlNM96sHio

Polynomial kernels



https://www.youtube.com/watch?v=OdlNM96sHio

What is kernel

$$\Phi(\cdot): R^n \rightarrow R^m, where\ m > n$$

Polynomial Kernel: $\boldsymbol{x} = [x_1, x_2], \Phi(\boldsymbol{x}) = [x_1^2, \sqrt{2}x_1x_2, x_2^2]$

$$\Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{z}) = (x_1^2z_1^2) + (x_2^2z_2^2) + 2x_1x_2z_1z_2$$

A circle in 2D becomes a plane in this 3D space. But calculating $\phi(\boldsymbol{x})$ for millions of points in high dimensions is computationally impossible when m is extremely large

Coincidence?

$$\Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{z}) = (x_1^2 z_1^2) + (x_2^2 z_2^2) + 2x_1 x_2 z_1 z_2$$

$$\mathrm{K}(\mathbf{x}, \mathbf{z}) = (\boldsymbol{x} \cdot \boldsymbol{z})^2 = (x_1 z_1 + x_2 z_2)^2$$

They are identical! We computed the 3D dot product using only 2D operations.

Kernel: 3 multiplications
Product on kernel space: 3 addition and 3 multiplication
Kernel tricks: 2 multiplications + 1 addition + 1 mutiplication

Kernel tricks

A kernel function $\boldsymbol{k}: \boldsymbol{X} \times \boldsymbol{X} \rightarrow R$

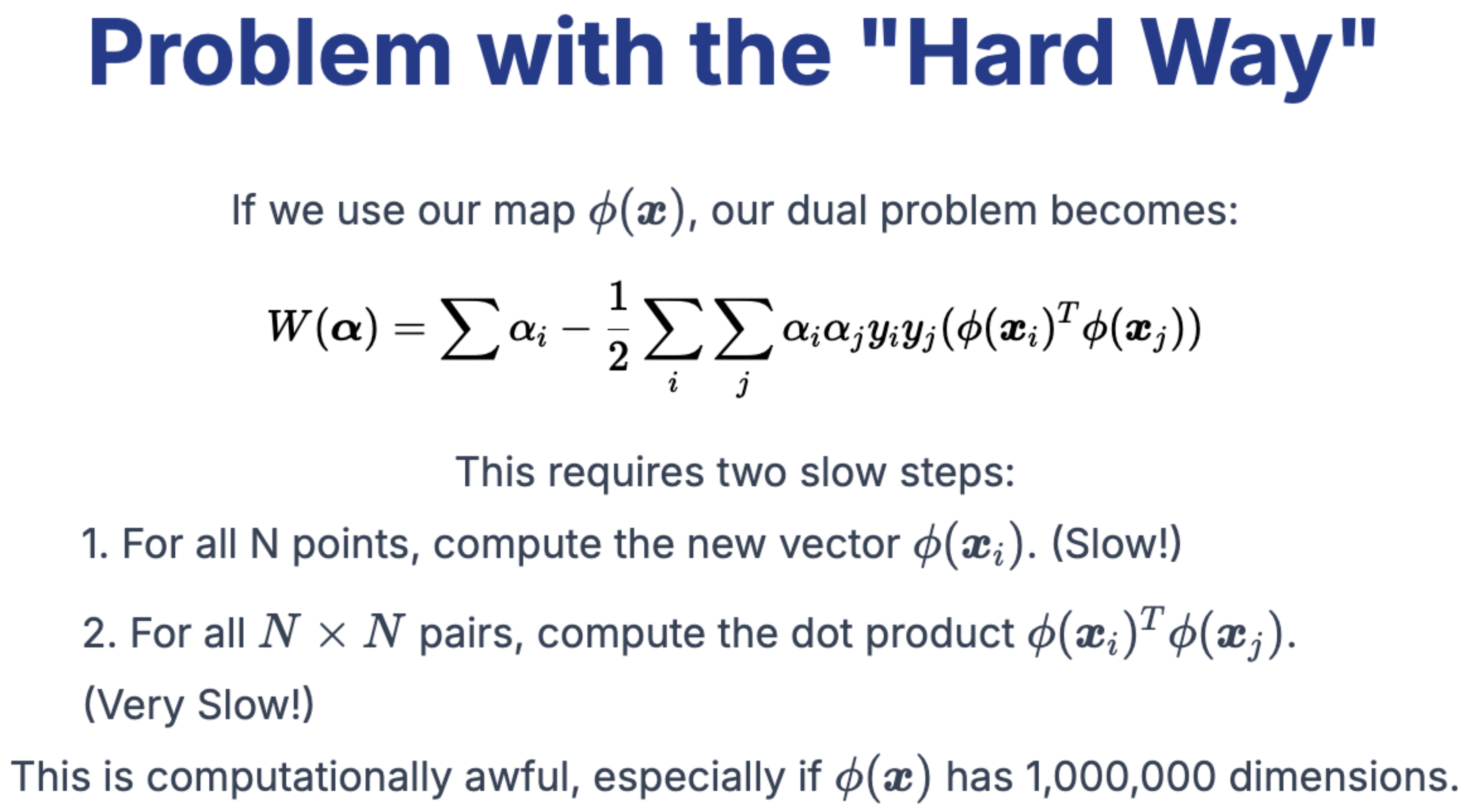
$$\mathbf{K} = \mathbf{X}\mathbf{X}^\top = \begin{pmatrix} x_1^\top x_1 & x_1^\top x_2 & \cdots & x_1^\top x_N \\ x_2^\top x_1 & x_2^\top x_2 & \cdots & x_2^\top x_N \\ \vdots & \vdots & \ddots & \vdots \\ x_N^\top x_1 & x_N^\top x_2 & \cdots & x_N^\top x_N \end{pmatrix} = (x_i^\top x_j)_{i,j} \in \mathbb{R}^{N\times N}$$

Hard way: $\Phi(\boldsymbol{x_i}) \cdot \Phi(\boldsymbol{z_j})$

Easy way: $k(x_i, x_j)$

Purpose: enable computation of linear classifiers in high-dimensional space without performing computations in this high-dimensional space directly.

Recall the SVM



Problem with the "Hard Way"

If we use our map $\phi(\boldsymbol{x})$, our dual problem becomes:

$$W(\boldsymbol{\alpha}) = \sum \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j (\phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j))$$

This requires two slow steps:

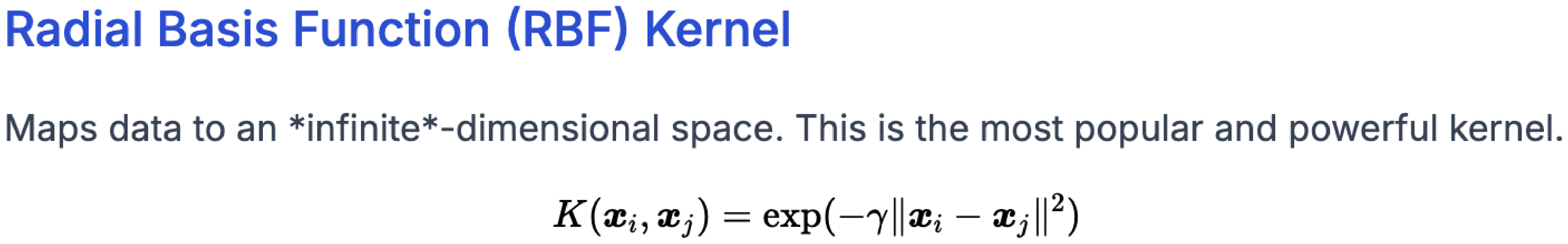
1. For all N points, compute the new vector $\phi(\boldsymbol{x}_i)$. (Slow!)
2. For all $N \times N$ pairs, compute the dot product $\phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$. (Very Slow!)

This is computationally awful, especially if $\phi(\boldsymbol{x})$ has 1,000,000 dimensions.

Other kernels

We have only considered the polynomial kernel $\mathrm{K}(\mathbf{x}, \mathbf{z}) = (\boldsymbol{x} \cdot \boldsymbol{z})^2$

What if we would like to increase the dimensionality from m to infinite?



Radial Basis Function (RBF) Kernel

Maps data to an *infinite*-dimensional space. This is the most popular and powerful kernel.

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2)$$

Euclidean v.s. Hilbert Space

Aspect	Euclidean Space	Hilbert Space
Dimensionality	Finite	Possibly infinite
Elements	Vectors of real numbers	Functions, sequences, or feature mappings
Inner product	$< x, y > = \sum x_i y_i$	$< f, g >_H$, can involve integrals or kernels
Distance	(|x - y|_2)	(|f - g|_\mathcal{H})
Completeness	Automatically complete	Must be complete to be a Hilbert space
Example	3D geometry	Function spaces, RKHS in kernel methods

A **Hilbert space** is a more general concept — it's **an inner product space** that is also complete (i.e., all **Cauchy sequences** converge within it).
You can think of a Hilbert space as a **possibly infinite-dimensional generalization** of Euclidean space.

RBF kernel

$$k(x,z) = \exp\bigl(-\gamma(x^2 - 2xz + z^2)\bigr) = \exp(-\gamma x^2)\exp(-\gamma z^2)\exp(2\gamma xz)$$

$$\exp(2\gamma xz) = \frac{\sum(2r)^n x^n z^n}{n!}$$

multinomial theorem $(xz)^n = \sum_{|\alpha|=n} \frac{n!}{\alpha!} x^\alpha z^\alpha$ where $\alpha = (\alpha_1, \cdots \alpha_d), |\alpha| = \sum \alpha_i = n, \alpha! = \prod \alpha_i$

$$k(x,z) = \exp(-\gamma x^2)\exp(-\gamma z^2) \overset{+\infty}{\sum}_{n=0} \sum_{|\alpha|=n} \frac{n!}{\alpha!} x^\alpha z^\alpha$$

$$k(x,z) = \sum_{|\alpha|=n} \left[\exp(-\gamma x^2)\sqrt{\frac{(2\gamma)^{|\alpha|}}{\alpha!}}\, x^\alpha\right] \left[\exp(-\gamma z^2)\sqrt{\frac{(2\gamma)^{|\alpha|}}{\alpha!}}\, z^\alpha\right.$$

$$\Phi_\alpha(x) = \exp(-\gamma x^2)\sqrt{\frac{(2\gamma)^{|\alpha|}}{\alpha!}}\, x^\alpha, \qquad k(x,z) = \sum_{\alpha}^{+\infty} \Phi_\alpha(x)\Phi_\alpha(z)$$

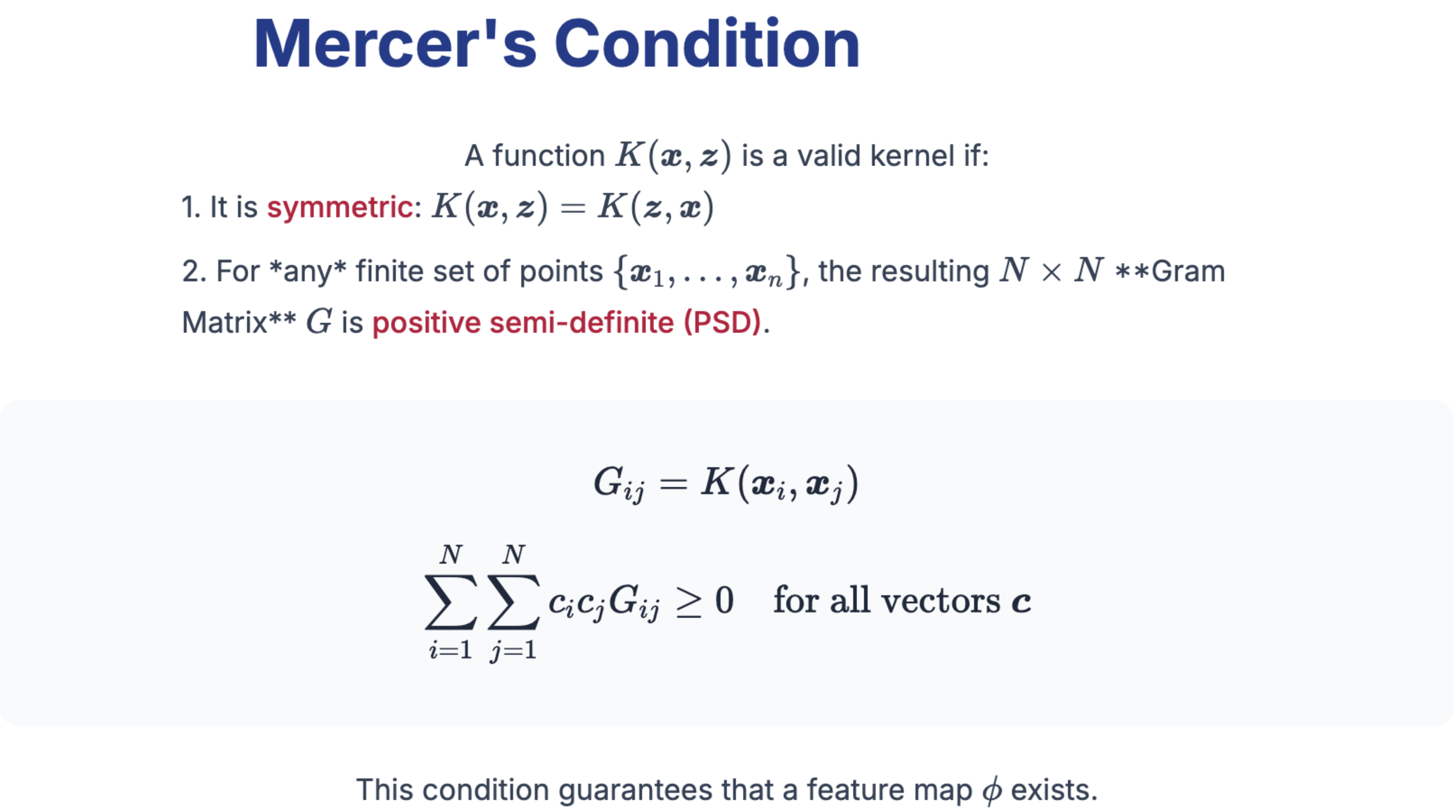
https://en.wikipedia.org/wiki/Multinomial_theorem

Are they valid kernel?

Select all bivariate functions $k(x, z)$ that are not valid kernel functions

(a) $x^\top M\, z$ (b) $e^{-||x-z||_2}$ (c) $\sin(x)\cos(z)$ (d) $\min\ \{x, z\}$

How to create your own kernel?



Mercer's Condition

A function $K(\boldsymbol{x}, \boldsymbol{z})$ is a valid kernel if:

1. It is symmetric: $K(\boldsymbol{x}, \boldsymbol{z}) = K(\boldsymbol{z}, \boldsymbol{x})$

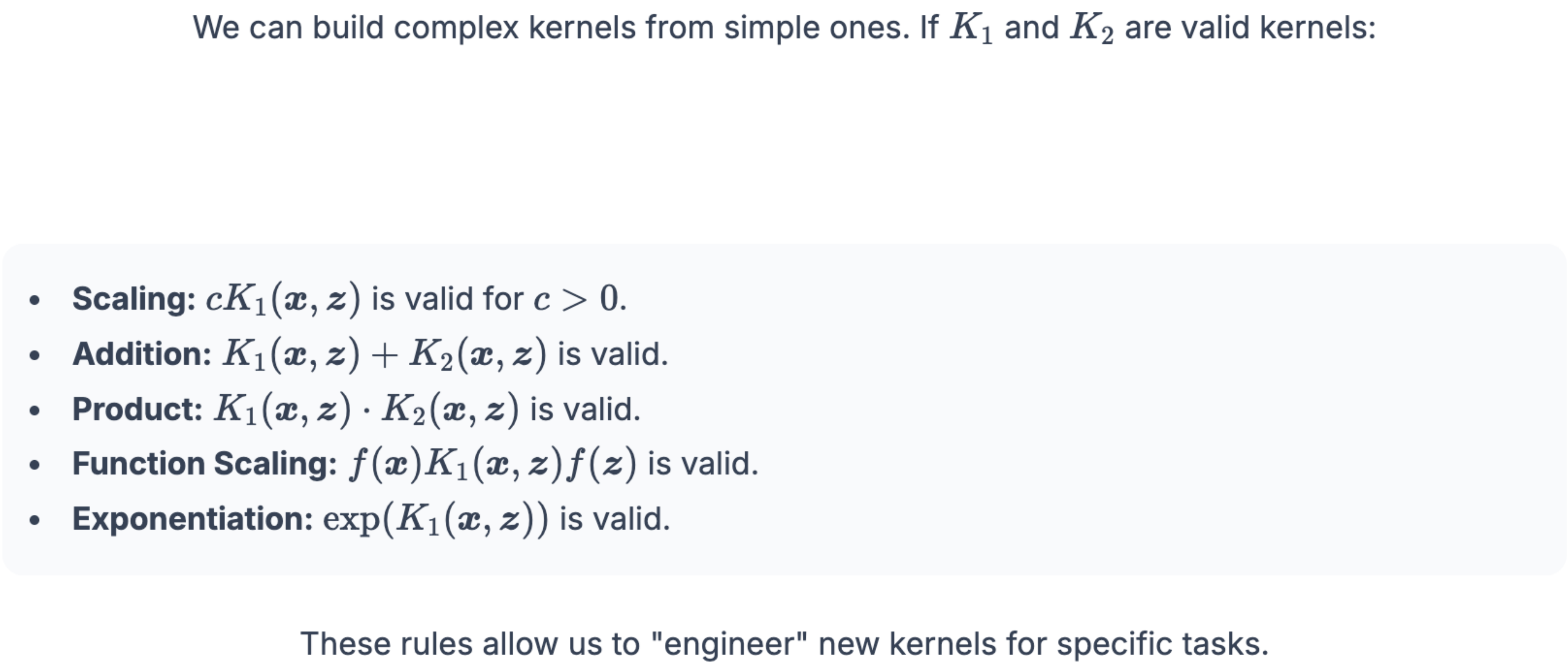
2. For *any* finite set of points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, the resulting $N \times N$ **Gram Matrix** G is positive semi-definite (PSD).

$$G_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

$$\sum_{i=1}^{N} \sum_{j=1}^{N} c_i c_j G_{ij} \geq 0 \quad \text{for all vectors } \boldsymbol{c}$$

This condition guarantees that a feature map ϕ exists.

Some properties



We can build complex kernels from simple ones. If K_1 and K_2 are valid kernels:

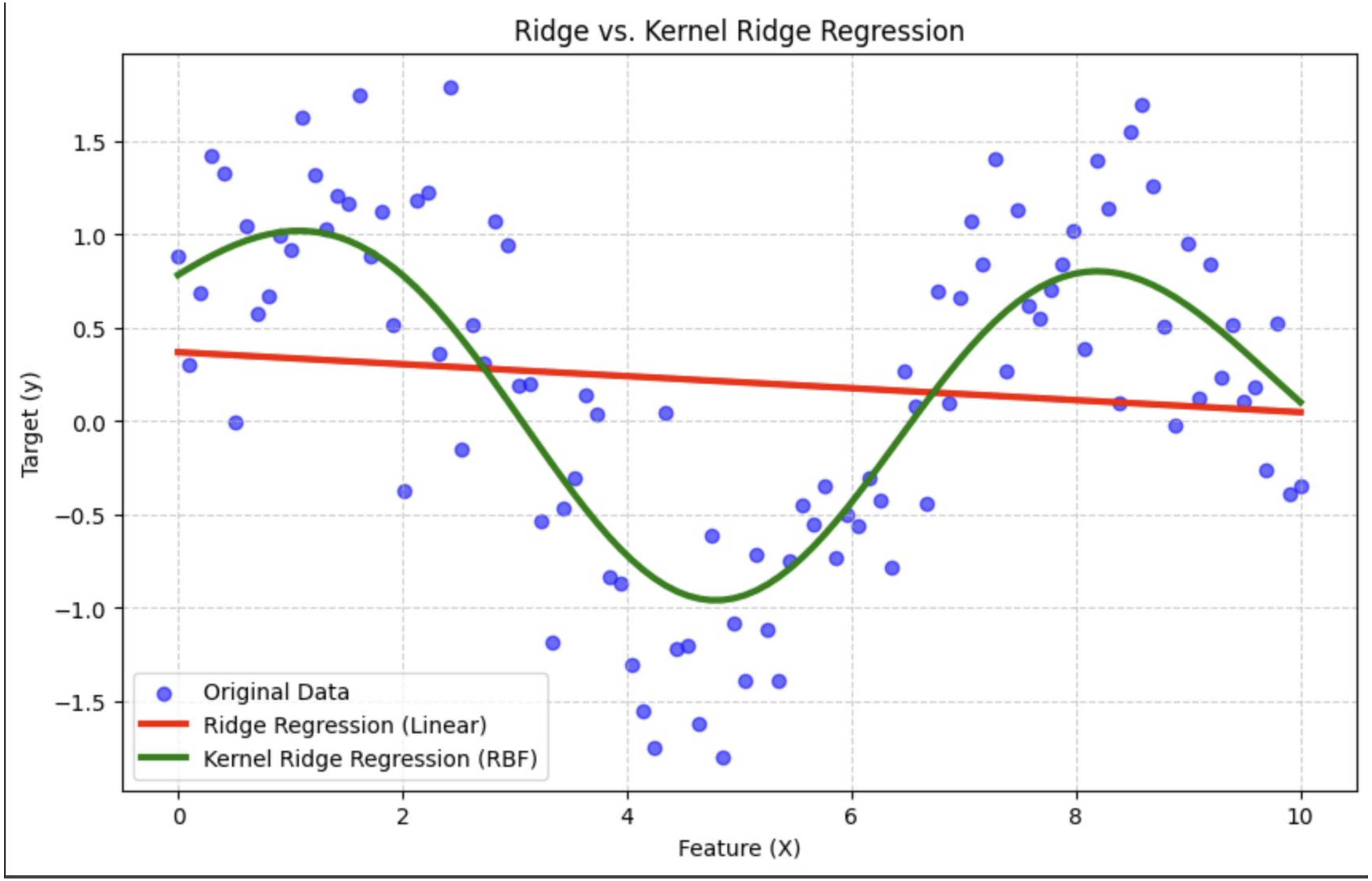
- **Scaling:** $cK_1(\boldsymbol{x}, \boldsymbol{z})$ is valid for $c > 0$.
- **Addition:** $K_1(\boldsymbol{x}, \boldsymbol{z}) + K_2(\boldsymbol{x}, \boldsymbol{z})$ is valid.
- **Product:** $K_1(\boldsymbol{x}, \boldsymbol{z}) \cdot K_2(\boldsymbol{x}, \boldsymbol{z})$ is valid.
- **Function Scaling:** $f(\boldsymbol{x})K_1(\boldsymbol{x}, \boldsymbol{z})f(\boldsymbol{z})$ is valid.
- **Exponentiation:** $\exp(K_1(\boldsymbol{x}, \boldsymbol{z}))$ is valid.

These rules allow us to "engineer" new kernels for specific tasks.

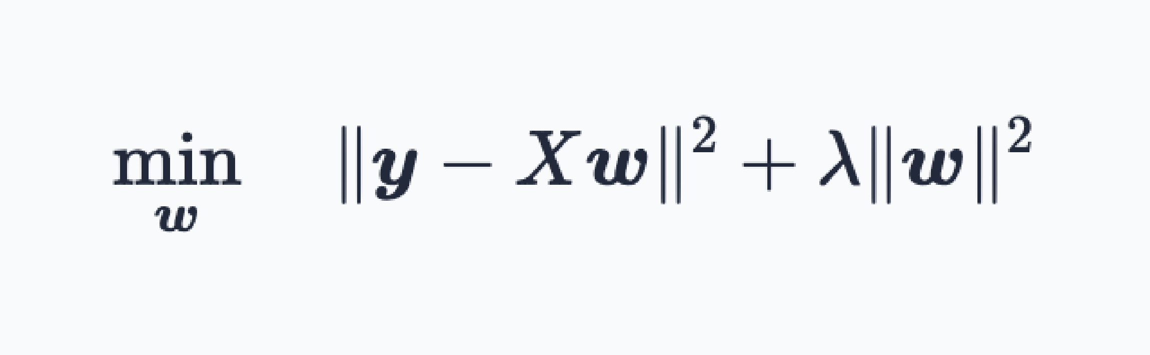
Intuitively why do we need SPD and symmetric?

Viewpoint	Why PSD is required
Geometric	Guarantees the kernel behaves like an inner product in some space.
Algebraic	Ensures Gram matrices have nonnegative eigenvalues (convex optimization).
Statistical	Covariance (or similarity) can’t be negative in total — ensures consistency.
Intuitive	Prevents “negative similarity” and allows meaningful distances and clustering.

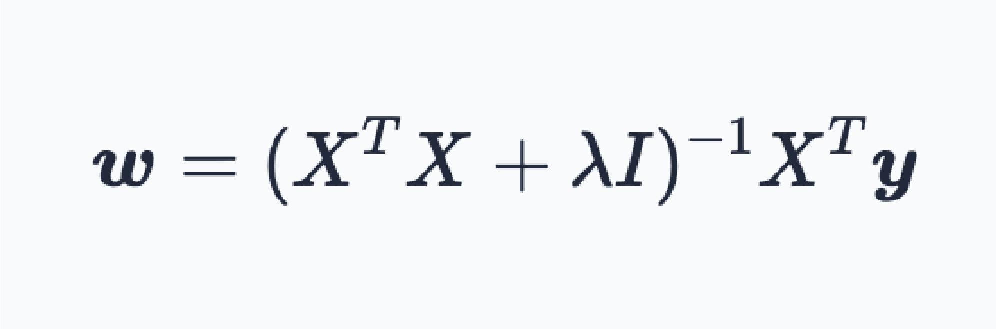
Kernel with ridge regression



Kernel with ridge regression



$$\min_{\boldsymbol{w}} \quad \|\boldsymbol{y} - X\boldsymbol{w}\|^2 + \lambda\|\boldsymbol{w}\|^2$$



$$\boldsymbol{w} = (X^T X + \lambda I)^{-1} X^T \boldsymbol{y}$$