Support Vector Machines

Nishant Mehta

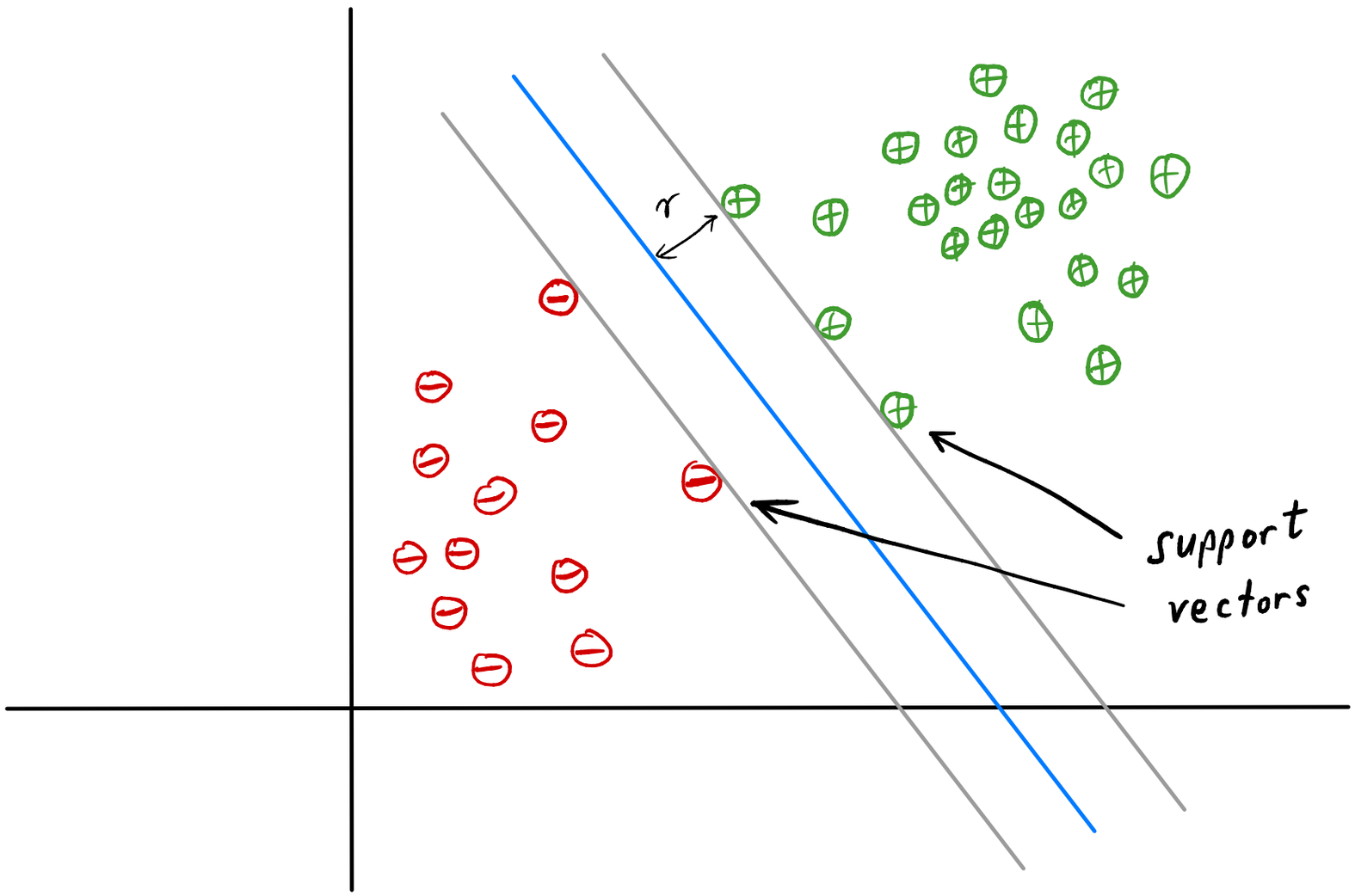
Lectures 8 and 9

Hard-margin SVM

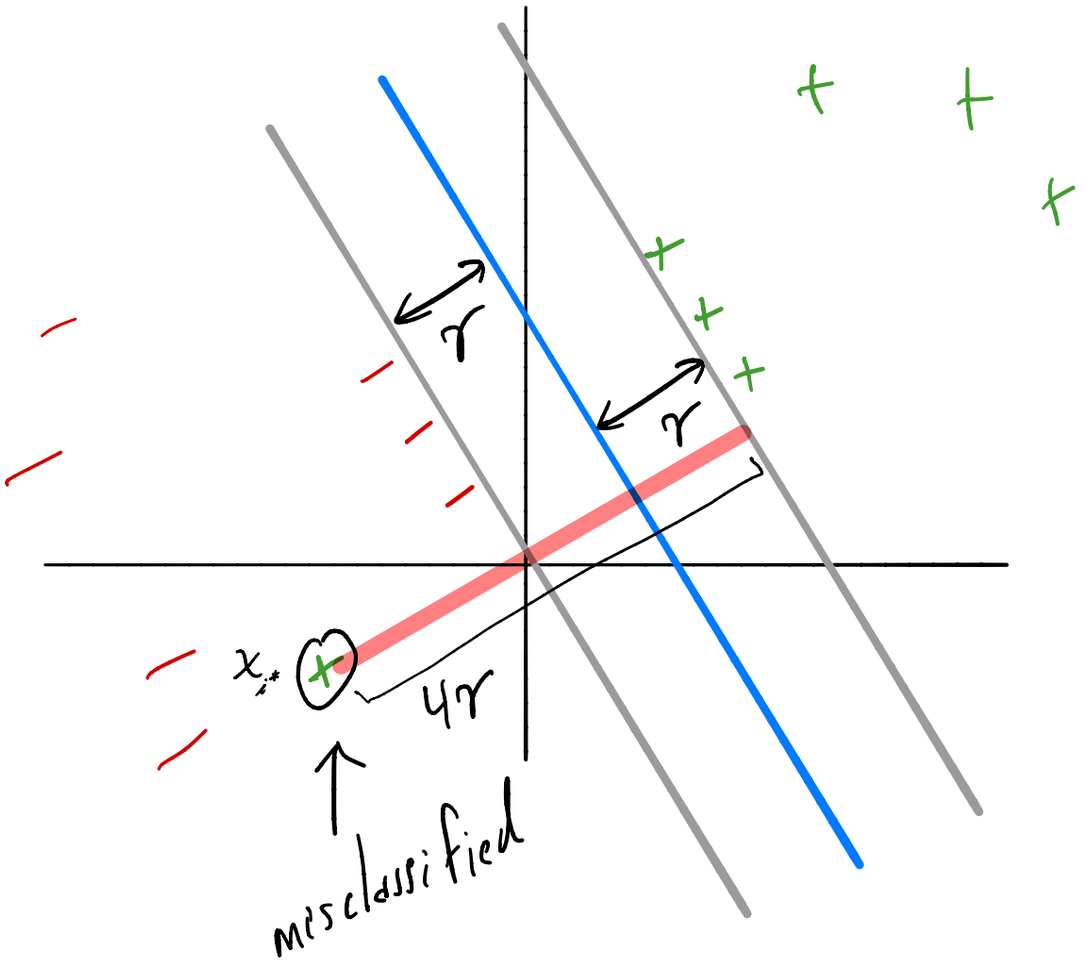
margin $\gamma = \frac{1}{\|w\|}$

Hard margin SVM problem

$$\begin{aligned} \underset{w,b}{\text{minimize}} \quad & \|w\|^2 \\ \text{subject to} \quad & y_i\big(\langle w, x_i\rangle + b\big) \geq 1, \ \ i = 1, \ldots, n. \end{aligned}$$



Soft-margin SVM



What if data isn't linearly separable?

Or, most of the data is separable with large margin, and some only with very low margin?

Soft-margin SVM problem

$C > 0$ regularization parameter

$$\begin{aligned} \underset{w \in \mathbb{R}^d,\, b \in \mathbb{R},\, \xi \in \mathbb{R}^n_+}{\text{minimize}} \quad & \|w\|^2 + C \sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & y_i\big(\langle w, x_i \rangle + b\big) \geq 1 - \xi_i, \;\; i = 1, \ldots, n \end{aligned}$$

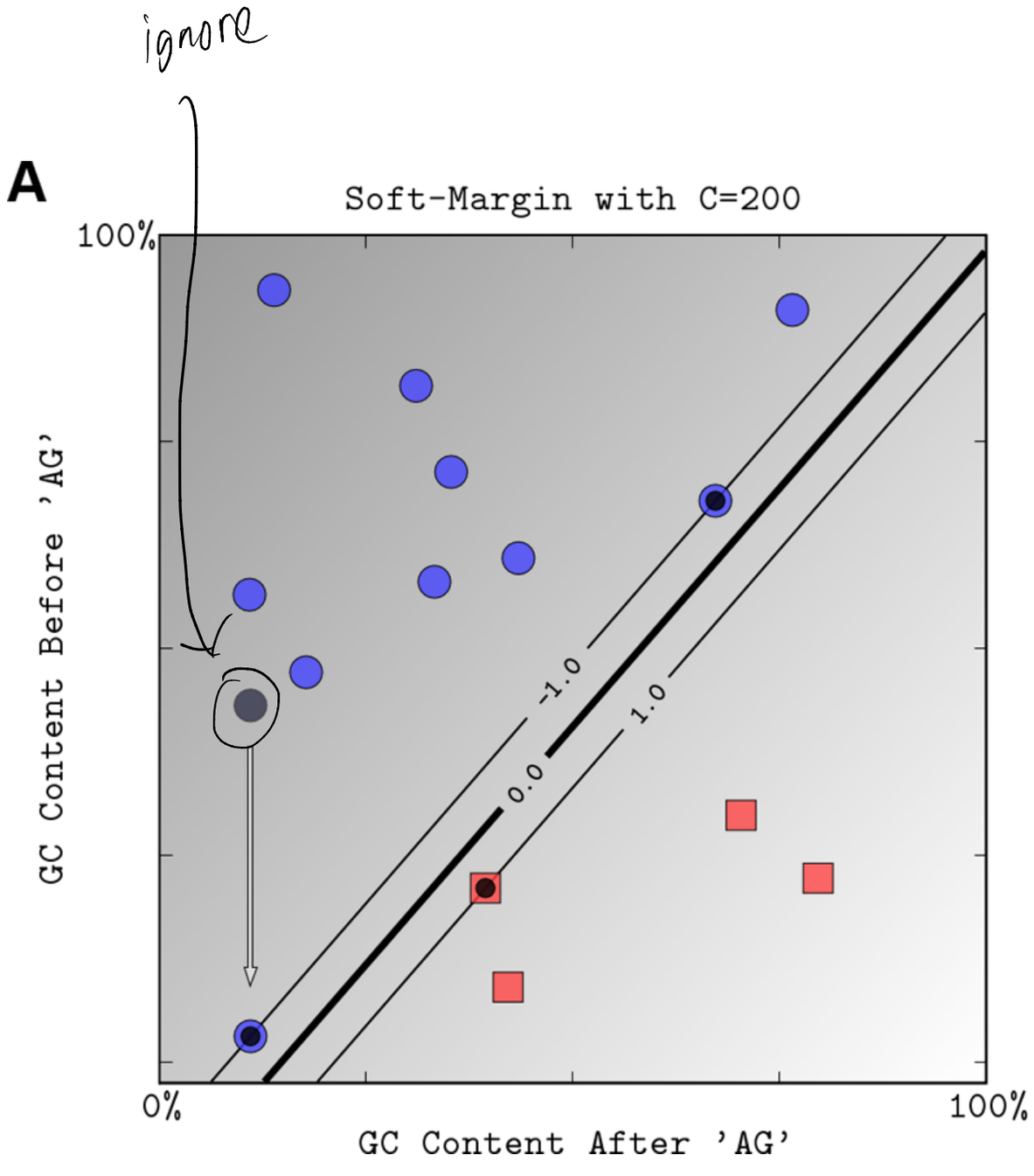
nonnegative vector in $\mathbb{R}^n$

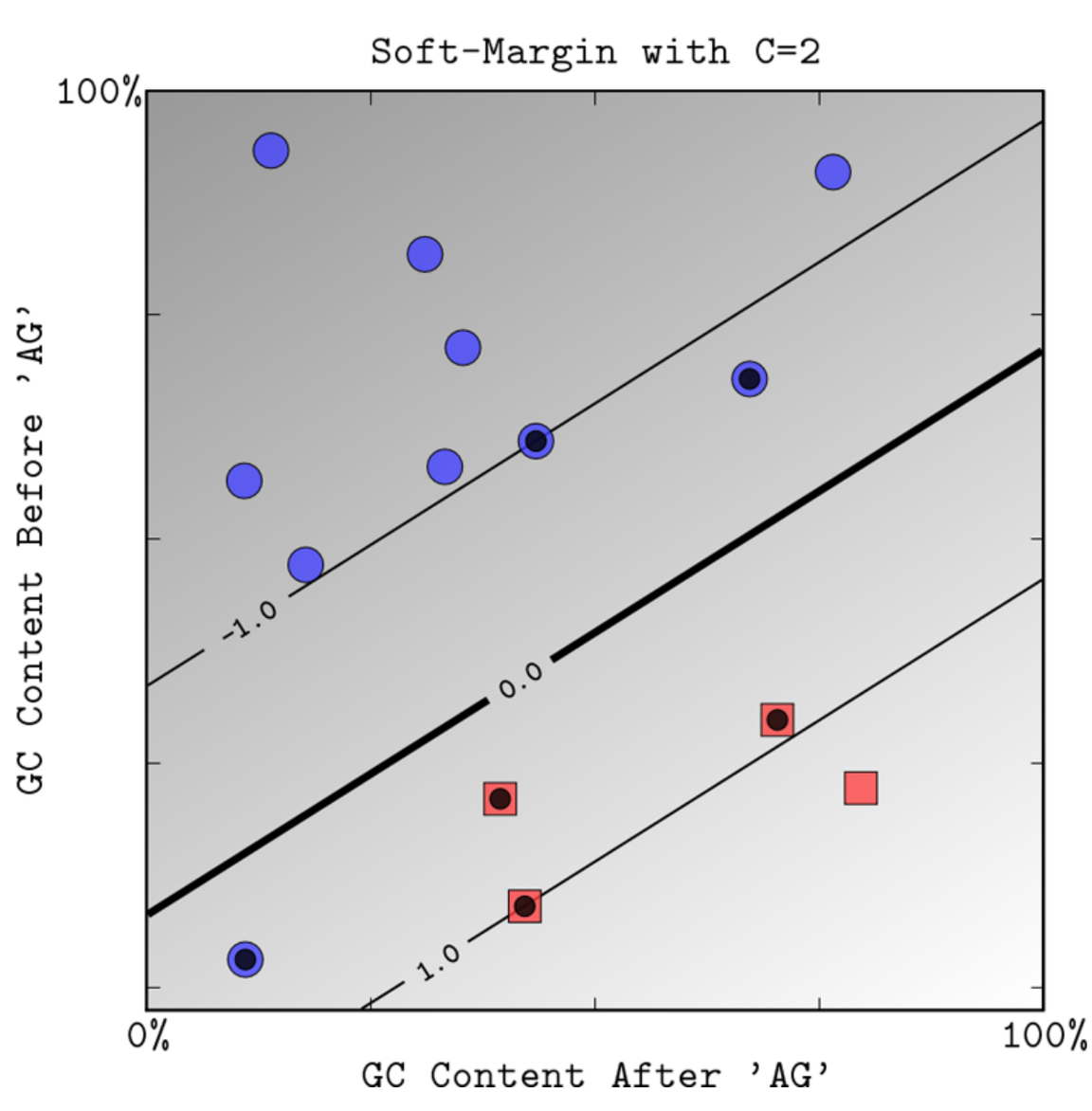
Examples: $\xi_i = 0$ $\quad y_i(\langle w, x_i\rangle + b) \geq 1$ margin of (w,b) on (x_i, y_i) is at least $\frac{1}{\|w\|}$

$\xi_i = \frac{2}{3}$ $\quad y_i(\langle w, x_i\rangle + b) = 1 - \xi_i = 1 - \frac{2}{3} = \frac{1}{3}$

$$\Rightarrow \frac{y_i(\langle w, x_i\rangle + b)}{\|w\|} = \frac{1/3}{\|w\|} = \frac{1-\xi_i}{\|w\|}$$

Varying C (linear kernel)

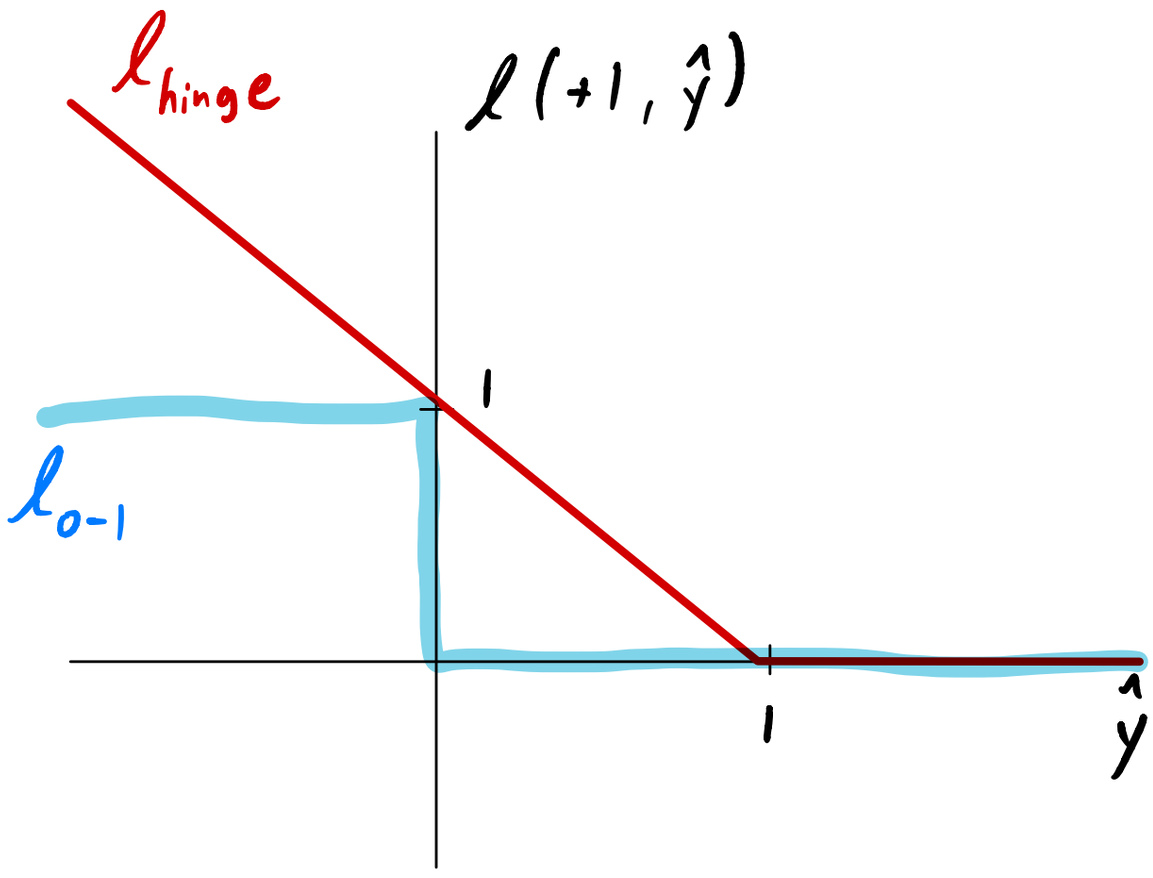




From “Support Vector Machines and Kernels for Computational Biology” (Ben-Hur et al., 2008)

Soft-margin SVM - Hinge Loss

$$\underset{w\in\mathbb{R}^n, b\in\mathbb{R}}{\text{minimize}} \quad \|w\|^2 + C\sum_{i=1}^{n} \max\Big\{0, 1 - y_i\big(\langle w, x_i\rangle + b\big)\Big\}$$



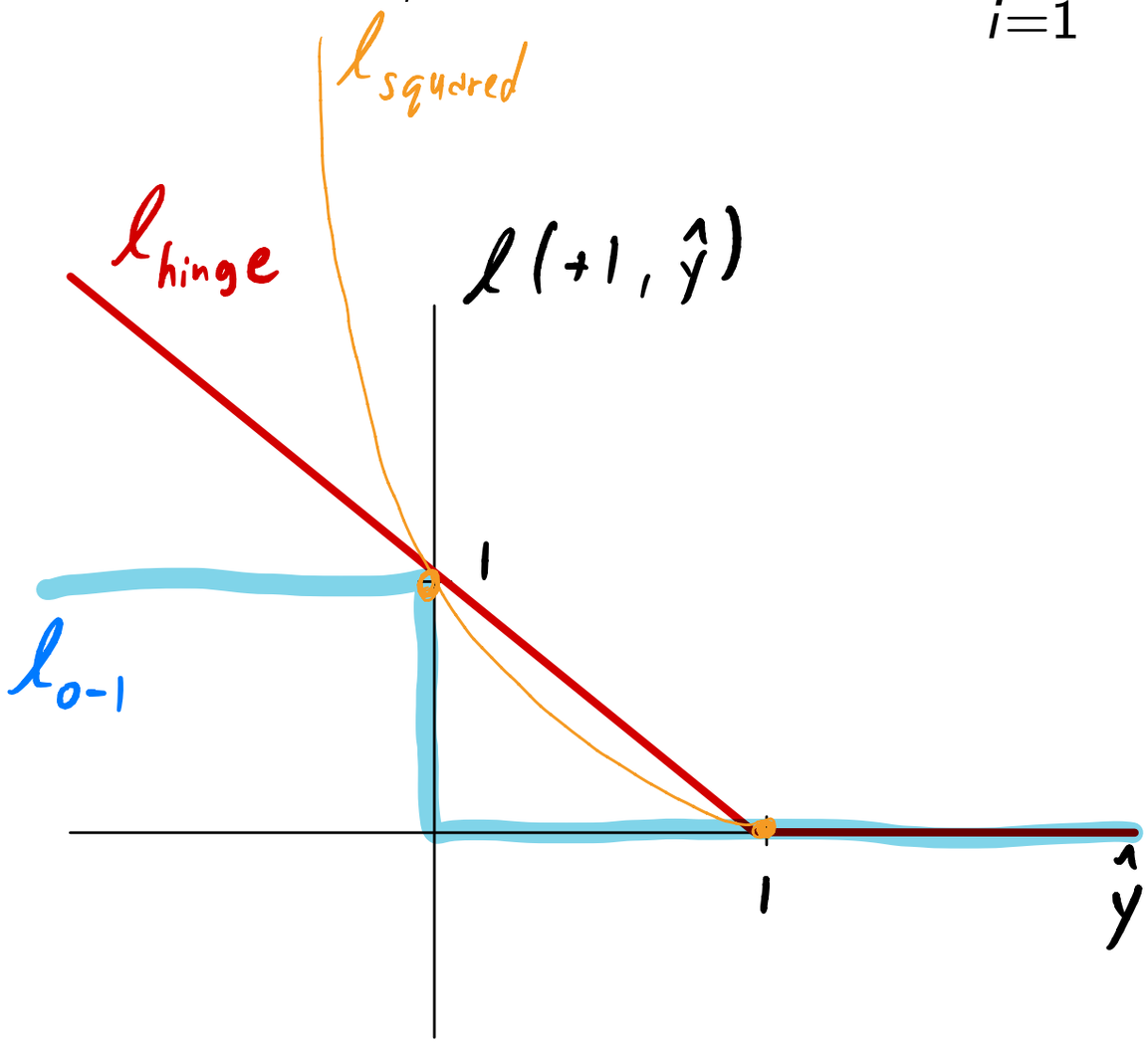
hinge loss

$$\ell_{\text{hinge}}(y, \hat{y}) = \max\{0, 1 - y\hat{y}\}$$

$$\ell_{hinge}(1, \hat{y}) = \max\{0, 1 - \hat{y}\}$$

Soft-margin SVM - Hinge Loss

$$\underset{w\in\mathbb{R}^n, b\in\mathbb{R}}{\text{minimize}} \quad \|w\|^2 + C\sum_{i=1}^{n} \max\Big\{0, 1 - y_i\big(\underbrace{\langle w, x_i\rangle + b}_{\hat{y}_i}\big)\Big\}$$



hinge loss

$$\ell_{\text{hinge}}(y, \hat{y}) = \max\{0, 1 - y\hat{y}\}$$

$$\underset{w\in\mathbb{R}^n, b\in\mathbb{R}}{\text{minimize}} \quad \|w\|^2 + C\sum_{i=1}^{n} \ell_{\text{hinge}}\big(y_i, f_{w,b}(x_i)\big)$$

SVM - Regularization viewpoint

SVM can be viewed as minimizing regularized training error under hinge loss

$$\underset{w\in\mathbb{R}^n, b\in\mathbb{R}}{\text{minimize}} \; \frac{1}{C}\|w\|^2 + C\sum_{i=1}^{n} \ell_{\text{hinge}}\left(y_i, f_{w,b}(x_i)\right)$$

$\lambda \geq 0$

Equivalent

$$\lambda = \frac{1}{C}$$

$$\underset{w\in\mathbb{R}^n, b\in\mathbb{R}}{\text{minimize}} \; \sum_{i=1}^{n} \ell_{\text{hinge}}\left(y_i, f_{w,b}(x_i)\right) + \lambda\|w\|^2$$

λ = regularization parameter

SVM dual problem

$$\underset{\alpha \in \mathbb{R}^n}{\text{maximize}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle$$

$$\text{subject to} \quad \sum_{i=1}^{n} y_i \alpha_i = 0$$

$$0 \leq \alpha_i \leq C, \ i = 1, \ldots, n$$

How to get w and b from this?

$$w = \sum_{i=1}^{n} y_i \alpha_i x_i$$

$$b = y_i - \sum_{j=1}^{n} y_j \alpha_j \langle x_i, x_j \rangle \quad \text{for any } i \text{ satisfying } 0 < \alpha_i < C$$

How to predict? $$f_{w,b}(x_{\text{test}}) = \langle w, x_{\text{test}} \rangle + b = \sum_{i=1}^{n} y_i \alpha_i \langle x_i, x_{\text{test}} \rangle + b$$

SVM dual problem - Inner products only

$$\underset{\alpha \in \mathbb{R}^n}{\text{maximize}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle$$

$$\text{subject to} \quad \sum_{i=1}^{n} y_i \alpha_i = 0$$

$$0 \leq \alpha_i \leq C, \ i = 1, \ldots, n$$

How to predict? $f_{w,b}(x_{\text{test}}) = \langle w, x_{\text{test}} \rangle + b = \sum_{i=1}^{n} y_i \alpha_i \langle x_i, x_{\text{test}} \rangle + b$

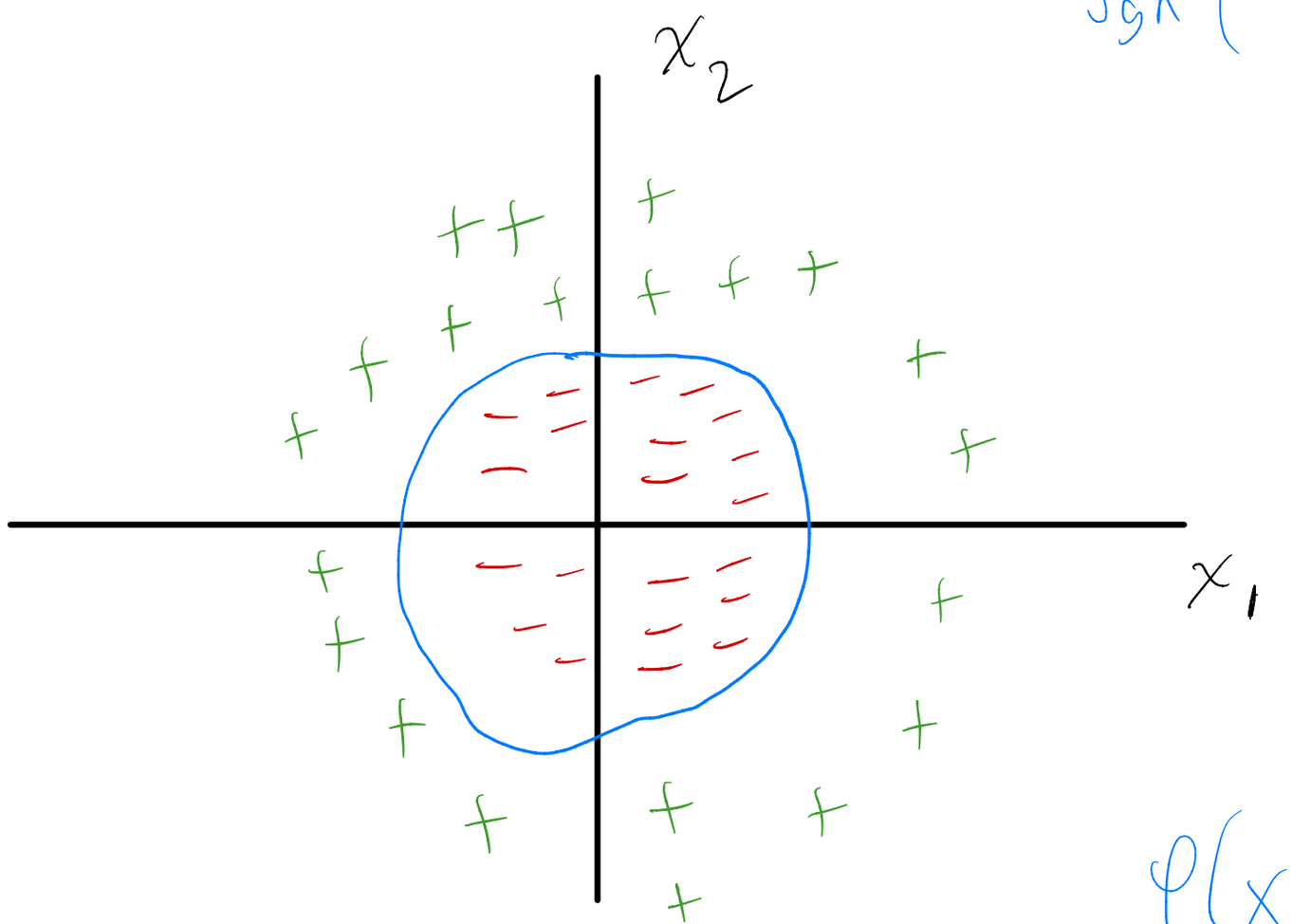
Dual SVM only needs inner products between input examples!

How can we achieve nonlinear classifiers?

$$sgn\left(1 \cdot x_1^2 + 1 \cdot x_2^2 - 1\right)$$

$$= sgn\left(\left\langle w, \begin{pmatrix} x_1^2 \\ x_2^2 \end{pmatrix} \right\rangle + b\right)$$

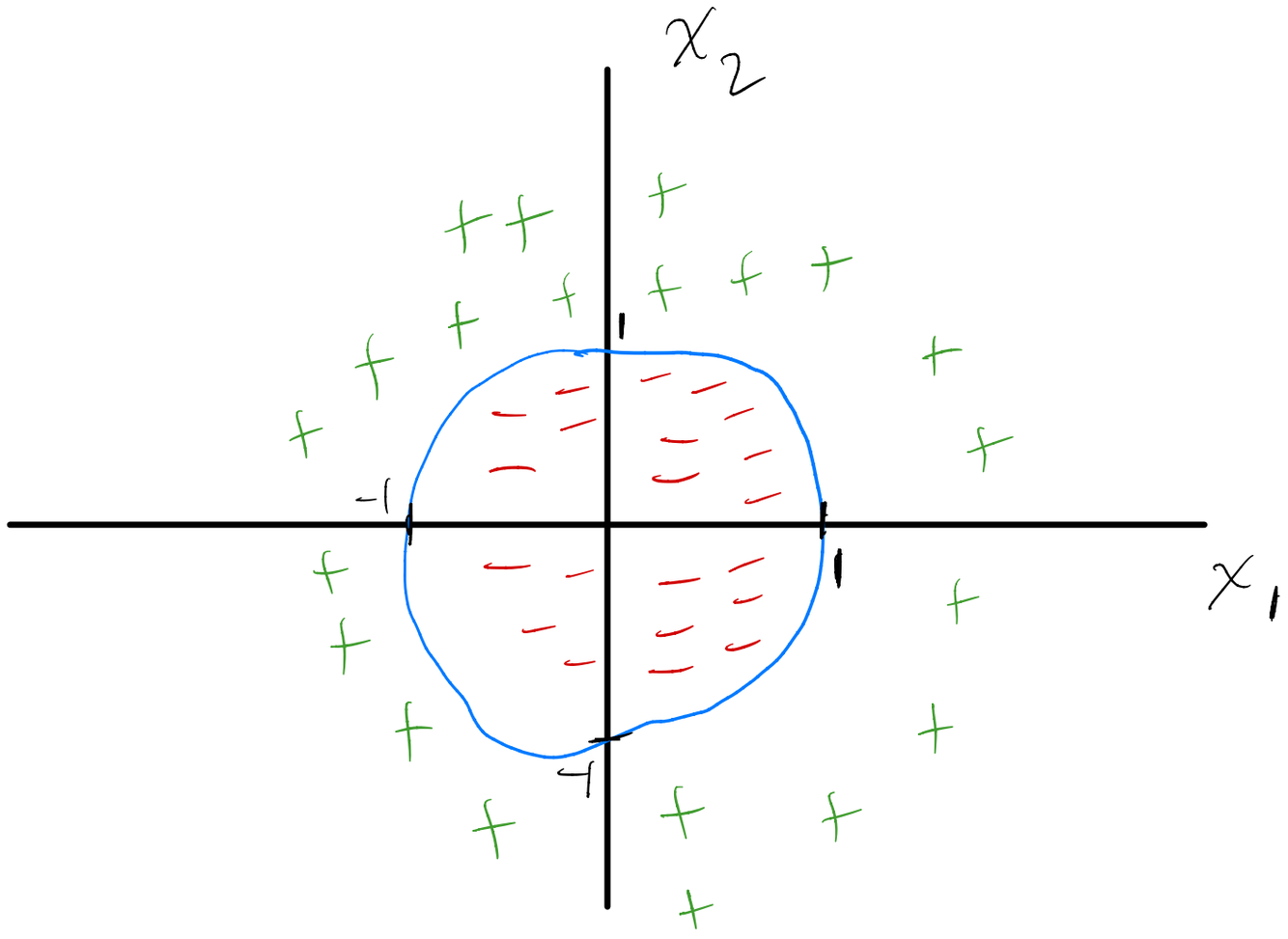
$$w = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad b = -1$$



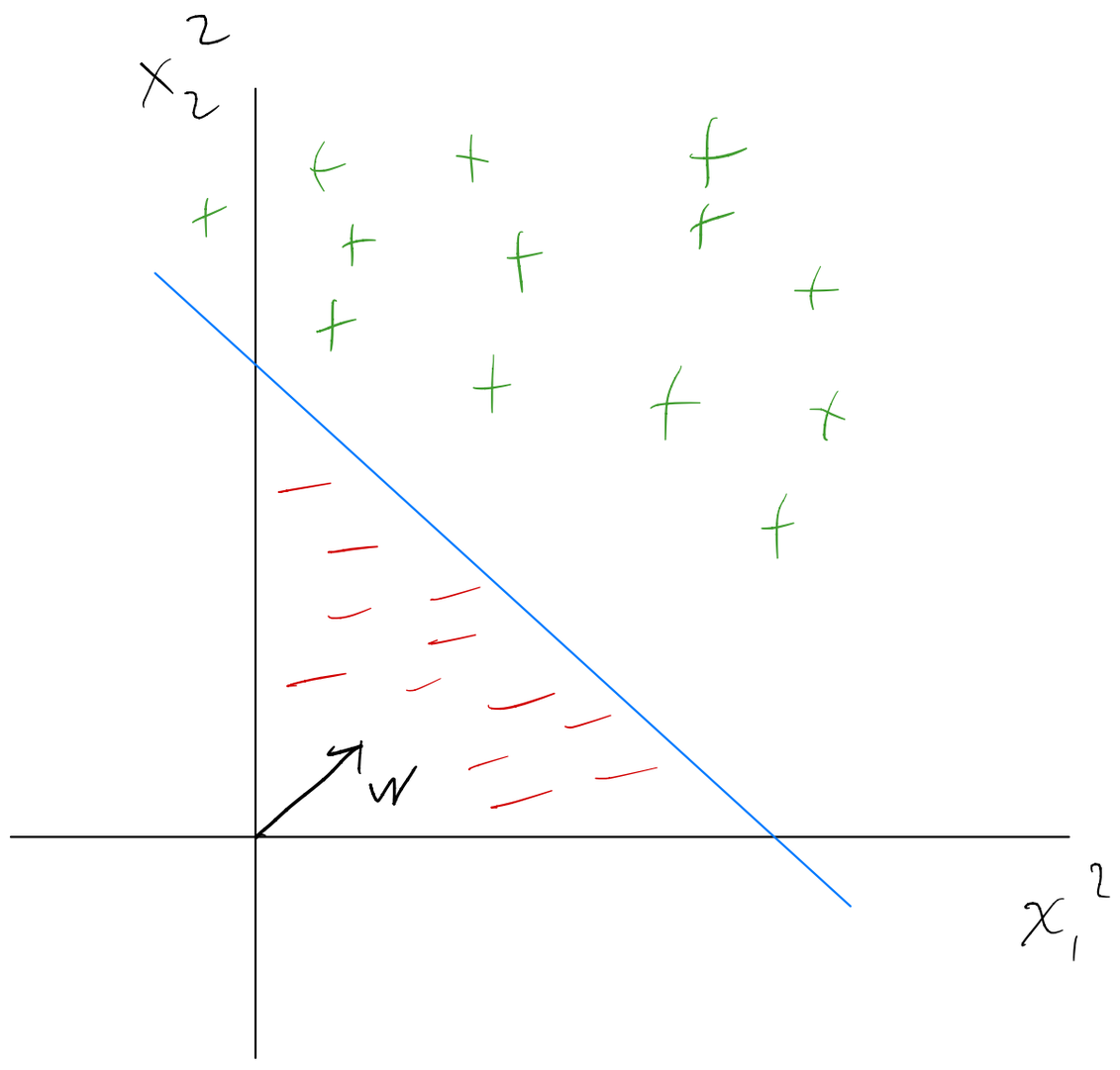
$$\varphi(x) = \begin{pmatrix} x_1^2 \\ x_2^2 \end{pmatrix}$$

Idea: feature map

Classification in original space

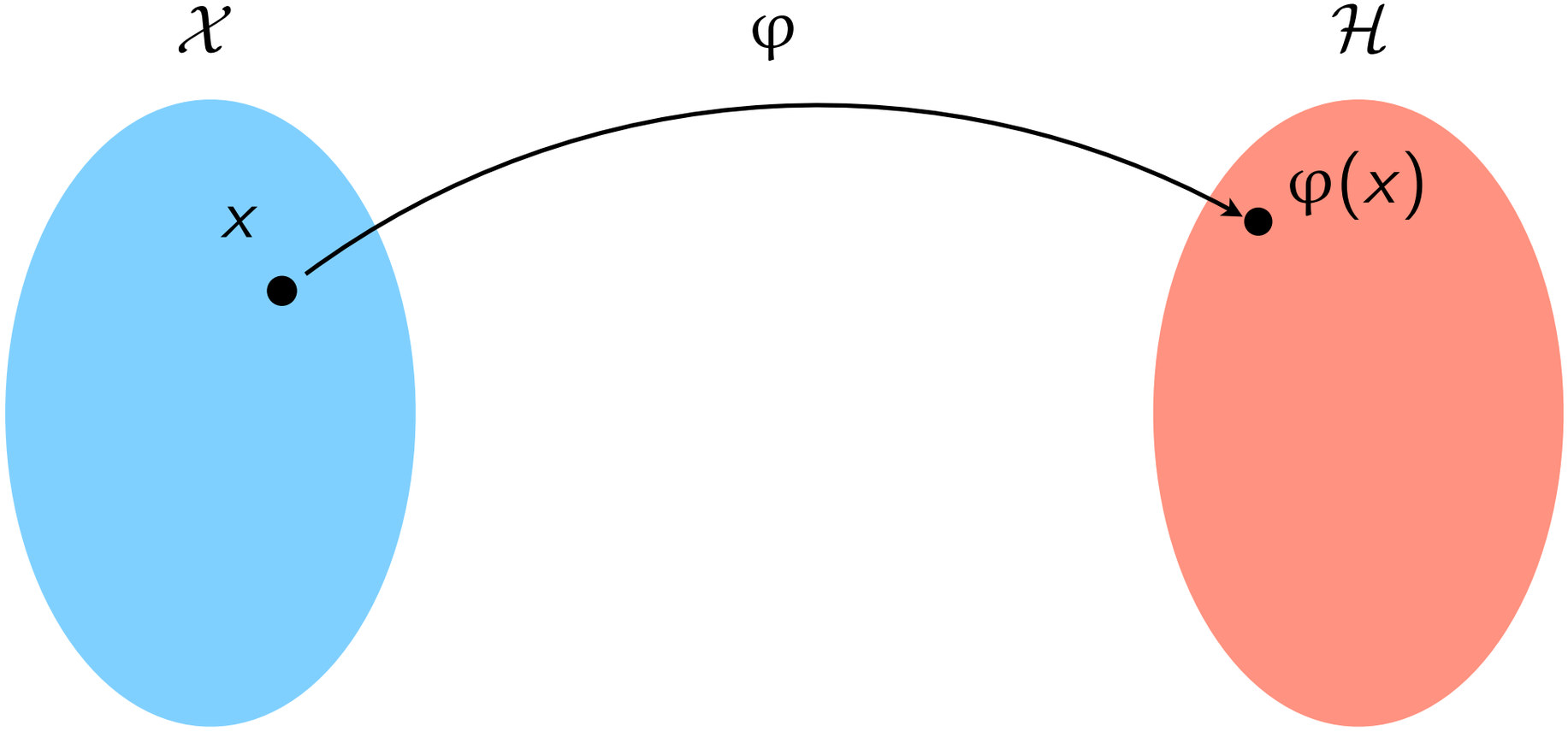


Classification in feature space



Idea: feature map

Use a feature map: $\varphi(x) : \mathcal{X} \to \mathcal{H}$



Kernel trick

Question: Can we compute inner product between input examples x and z in feature space without explicitly computing $\varphi(x)$ and $\varphi(z)$?

In many cases, yes! We use a *kernel function*:

$$k(x, z) = \langle \varphi(x), \varphi(\mathbf{z}) \rangle$$

Equal to inner product… but we won't compute it this way!

Example 1: Warm-up exercise

(dimension $d = 2$)

original space ($\mathcal{X}$): $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$

(dimension $= 3$)

feature space ($\mathcal{H}$): $\varphi(x) = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}\, x_1 x_2 \end{pmatrix}$

kernel function

$$k(x, z) = \langle \varphi(x), \varphi(z) \rangle = \left\langle \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}\, x_1 x_2 \end{pmatrix}, \begin{pmatrix} z_1^2 \\ z_2^2 \\ \sqrt{2}\, z_1 z_2 \end{pmatrix} \right\rangle$$

$$= x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 z_1 x_2 z_2$$

$$= (x_1 z_1 + x_2 z_2)^2 = \langle x, z \rangle^2 = k(x, z)$$

actual way to compute

Example 2: Polynomial kernel, one dimension

inner product $\langle x, z \rangle = xz$

The *polynomial kernel* (one dimension):

$$k(x, z) = (xz + a)^r = (xz)^r + \binom{r}{1}(xz)^{r-1}a + \ldots + \binom{r}{r-1}(xz)a^{r-1} + a^r$$

hyperparameter

What is the feature space?

$$= x^r z^r + \sqrt{\binom{r}{1} \cdot a}\, x^{r-1} \sqrt{\binom{r}{1} a}\, z^{r-1} + \ldots + \sqrt{a^r} \cdot \sqrt{a^r}$$

$$\varphi(x) = \begin{pmatrix} x^r \\ \sqrt{\binom{r}{1} \cdot a}\, x^{r-1} \\ \sqrt{\binom{r}{2} a^2}\, x^{r-2} \\ \vdots \\ \sqrt{\binom{r}{r} a^r}\, x^0 \end{pmatrix}$$

Example 3: Polynomial kernel, general dimension

$d \geq 1$

The *polynomial kernel* (general dimension):

$$k(x, z) = \left(\langle x, z \rangle + a\right)^r$$

$\varphi(x)$ has one feature for each monomial up to degree r

How many features are there in the feature space?

Example 3: Polynomial kernel, general dimension

The *polynomial kernel*:

$$k(x, z) = \left(\langle x, z\rangle + a\right)^r$$

$$\varphi(x) = \begin{pmatrix} x_1^r \\ x_2^{r-3} x_4^2 x_6 \\ x_4^3 \\ \vdots \end{pmatrix}$$

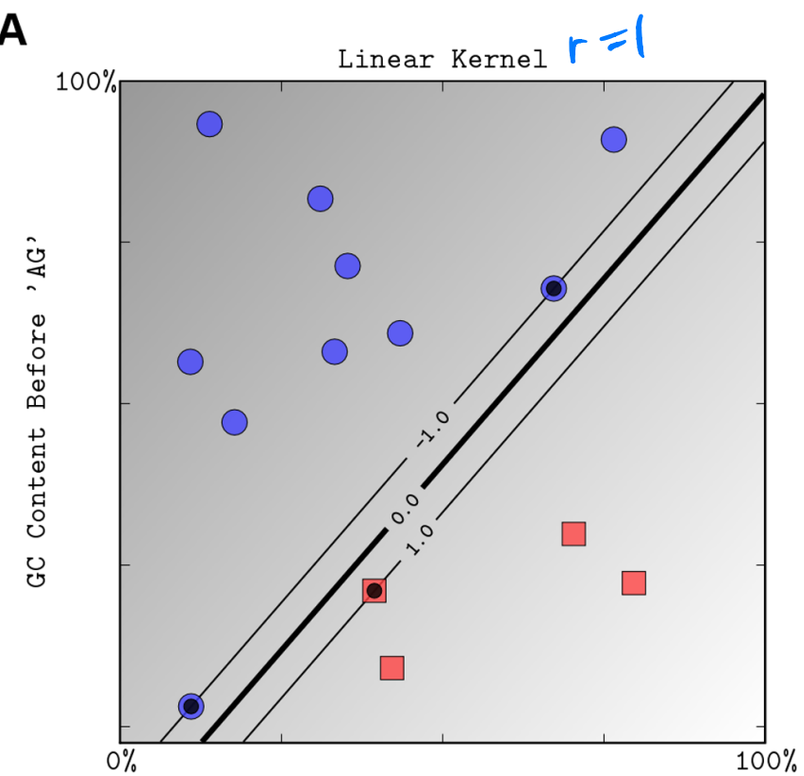
$\varphi(x)$ has one feature for each monomial up to degree r

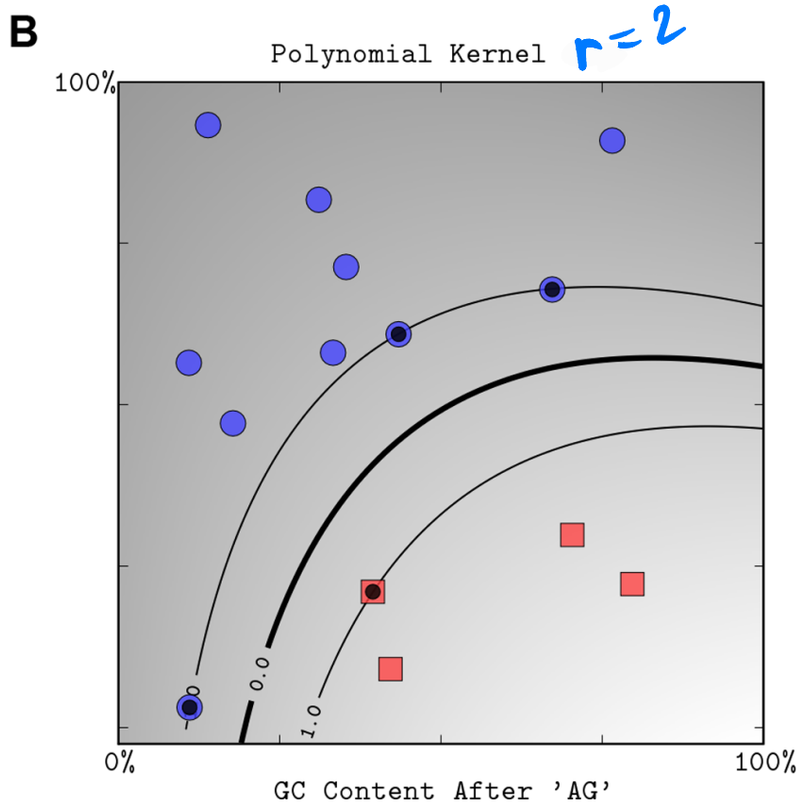
How many features are there in the feature space?

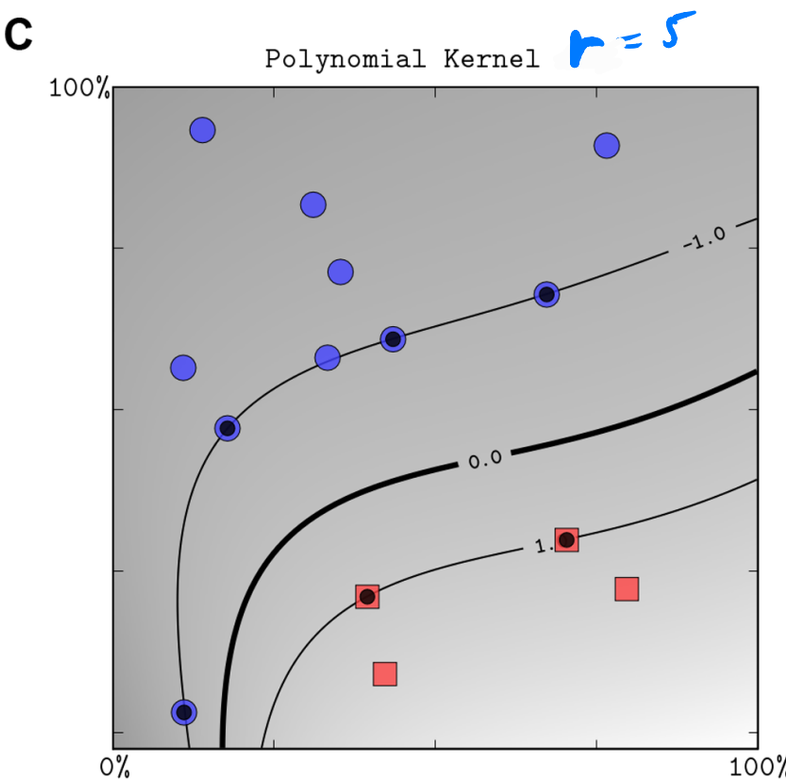
$$\binom{r+d}{d} = \binom{r+d}{r} \leq \text{order } d^r$$

But the kernel can be computed in only $O(d)$

Polynomial kernels of increasing degree







Gaussian kernel

The *Gaussian kernel* is based on the distance between two examples

$$k(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right)$$

bandwidth parameter

The Gaussian kernel is a type of similarity measure,
taking values between 0 and 1

What is the corresponding feature map $\varphi(x)$?

Gaussian kernel

The *Gaussian kernel* is based on the distance between two examples

$$k(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right)$$

bandwidth parameter

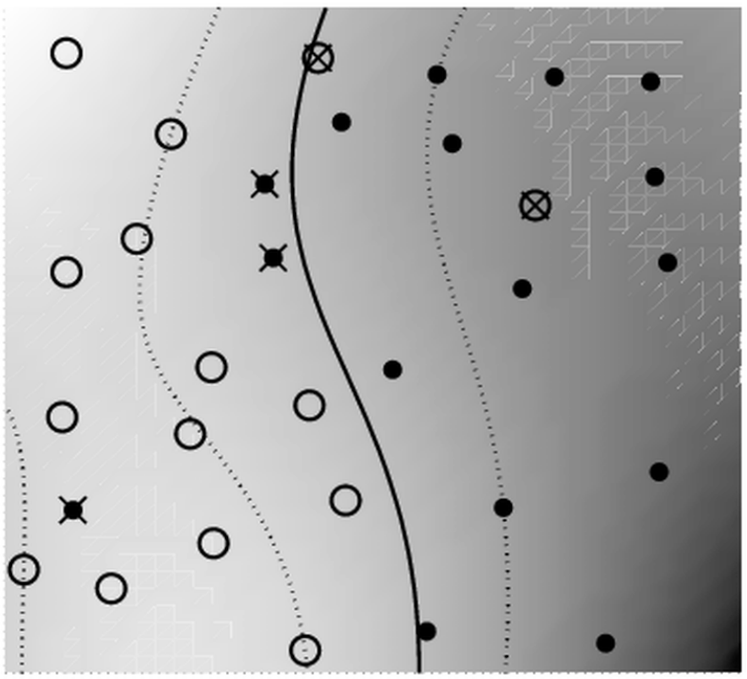
The Gaussian kernel is a type of similarity measure,
taking values between 0 and 1

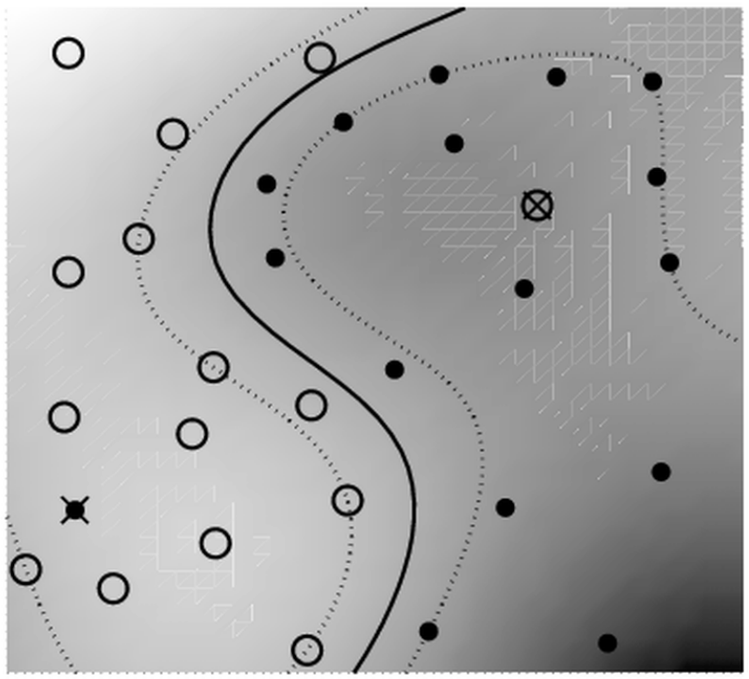
What is the corresponding feature map $\varphi(x)$?

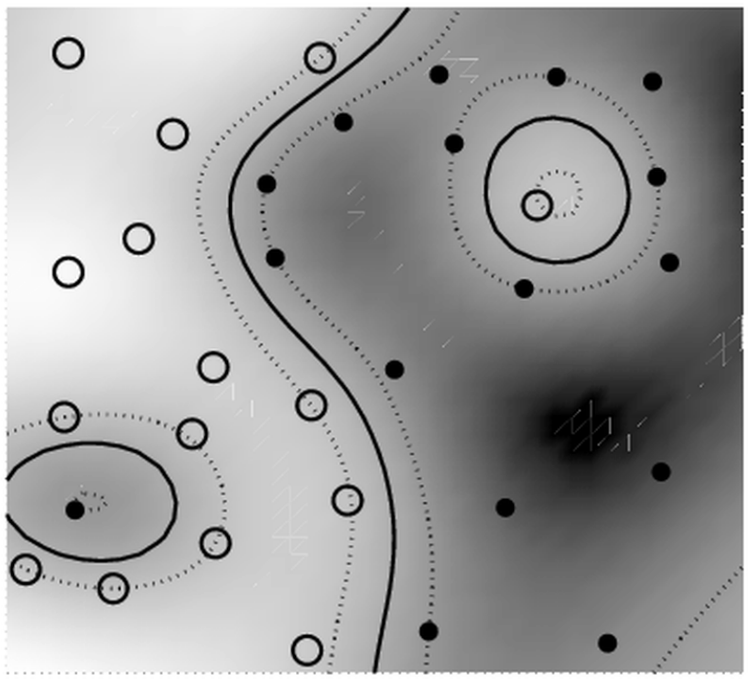
It's infinite dimensional!

Varying Gaussian kernel bandwidth (C kept constant)

Decreasing kernel bandwidth







From the book “Learning with Kernels” (Schölkopf and Smola, 2001)

The problem with kernel machines

- Kernel methods (like “kernelized SVMs” = SVMs using a kernel function) are nice in that they allow for nonlinear predictions

- But computation is a major disadvantage:
  - Training is expensive: quadratic program with n variables
  - Computing predictions is expensive: $O(nd)$ per test point

- Idea: approximate the feature map φ via some other feature map ψ and explicitly map the data to the feature space corresponding to ψ

  (for this to be helpful, the dimension of the new feature space should not be too high)

Random Fourier Features (RFF)

there exists a feature map such that the kernel computes an inner product in the corresponding feature space

- Given: a *positive definite* kernel that satisfies *shift invariance*

$$k(x, z) = k(x - z)$$

it turns out that p is a probability distribution

- Let $\hat{p}$ be the Fourier transform of $k(\cdot)$

- Draw D samples $\omega_1, \ldots, \omega_D$ independently from p

- Use feature map:

$$\psi(x) = \sqrt{\frac{1}{D}} \begin{pmatrix} \cos(\langle \omega_1, x \rangle) \\ \vdots \\ \cos(\langle \omega_D, x \rangle) \\ \sin(\langle \omega_1, x \rangle) \\ \vdots \\ \sin(\langle \omega_D, x \rangle) \end{pmatrix}$$

Random Fourier Features (RFF)

- What is special about Random Fourier Features?
- As $D \to \infty$, it holds that $\langle \psi(x), \psi(z) \rangle \to k(x, z)$
- Another cool thing: we can view increasing D as increasing complexity (higher D means richer feature space)

Example: RFF with Gaussian Kernel

- Consider Gaussian kernel (with unit bandwidth parameter) expressed as:

$$k(\Delta) = \exp\left(-\frac{\|\Delta\|_2^2}{2}\right)$$

- It can be shown that the probability distribution p is

$$p(\omega) = \frac{\exp\left(-\frac{\|\omega\|_2^2}{2}\right)}{(2\pi)^{d/2}}$$

RFF is interesting in its own right

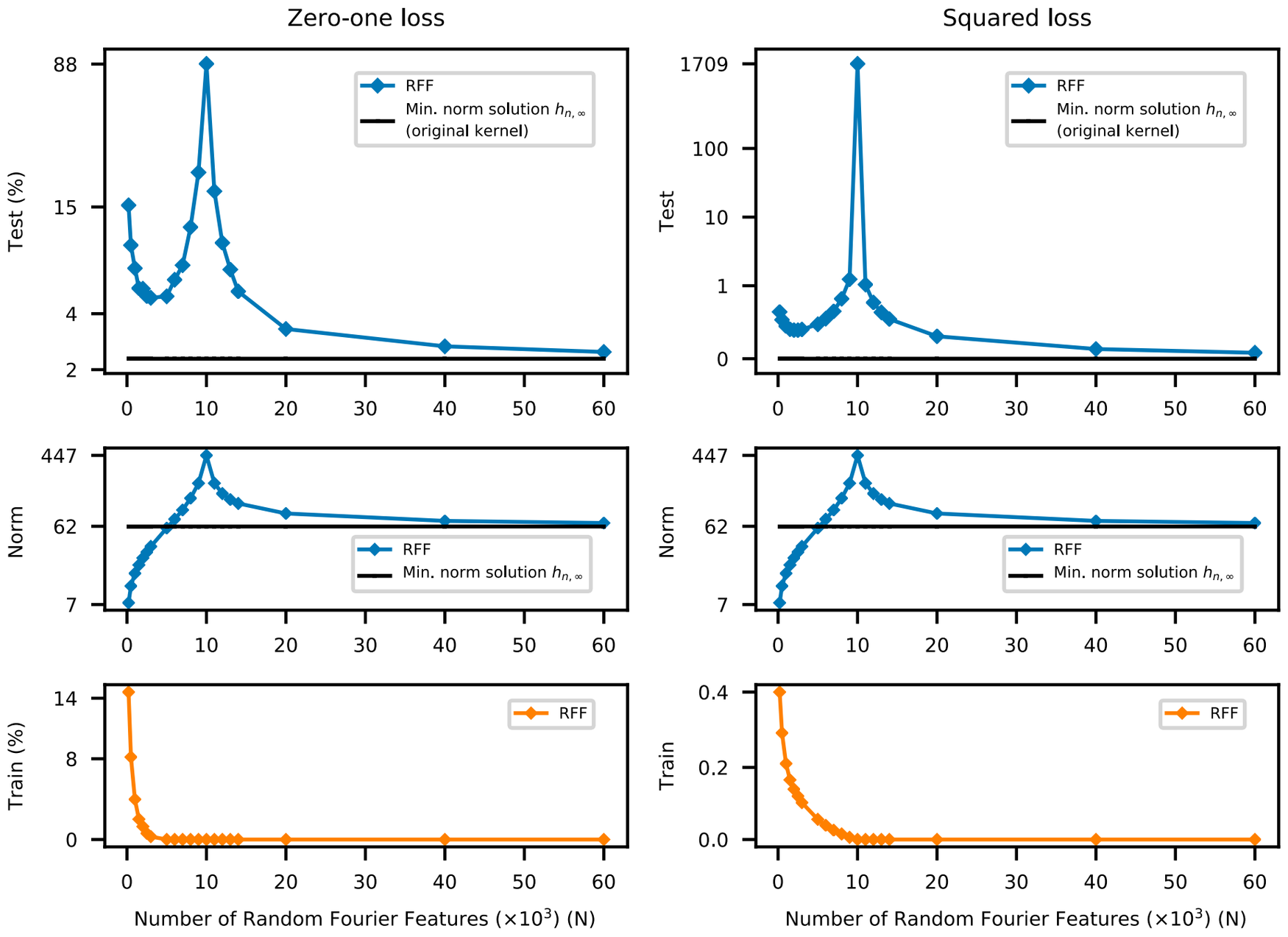


Fig. 2. Double-descent risk curve for the RFF model on MNIST. Shown are test risks (log scale), coefficient ℓ_2 norms (log scale), and training risks of the RFF model predictors $h_{n,N}$ learned on a subset of MNIST ($n = 10^4$, 10 classes). The interpolation threshold is achieved at $N = 10^4$.

from “Reconciling modern machine-learning practice and the classical bias–variance trade-off”
Belkin, Hsu, Ma, and Mandal, PNAS 2019.