CSC 580 Principles of Machine Learning

08 Kernel methods

Chicheng Zhang

Department of Computer Science

THE UNIVERSITY OF ARIZONA

*slides credit: built upon CSC 580 Fall 2021 lecture slides by Kwang-Sung Jun

Recall: Beyond linearity

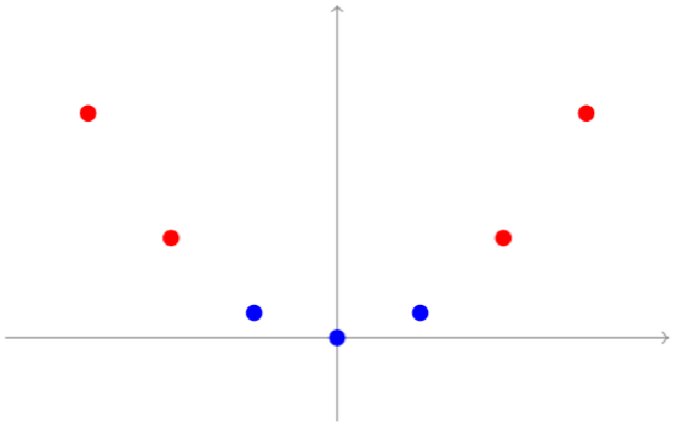
- Recall: for 1d problem, we embedded the feature: $x' = (x, 1) \in \mathbb{R}^2$

- Actually, the embedding trick is stronger.
  - $(x^2, x, 1)$: 2nd order polynomial
  - $(x^d, x^{d-1}, \dots, 1)$: d-th order polynomial (= degree d)

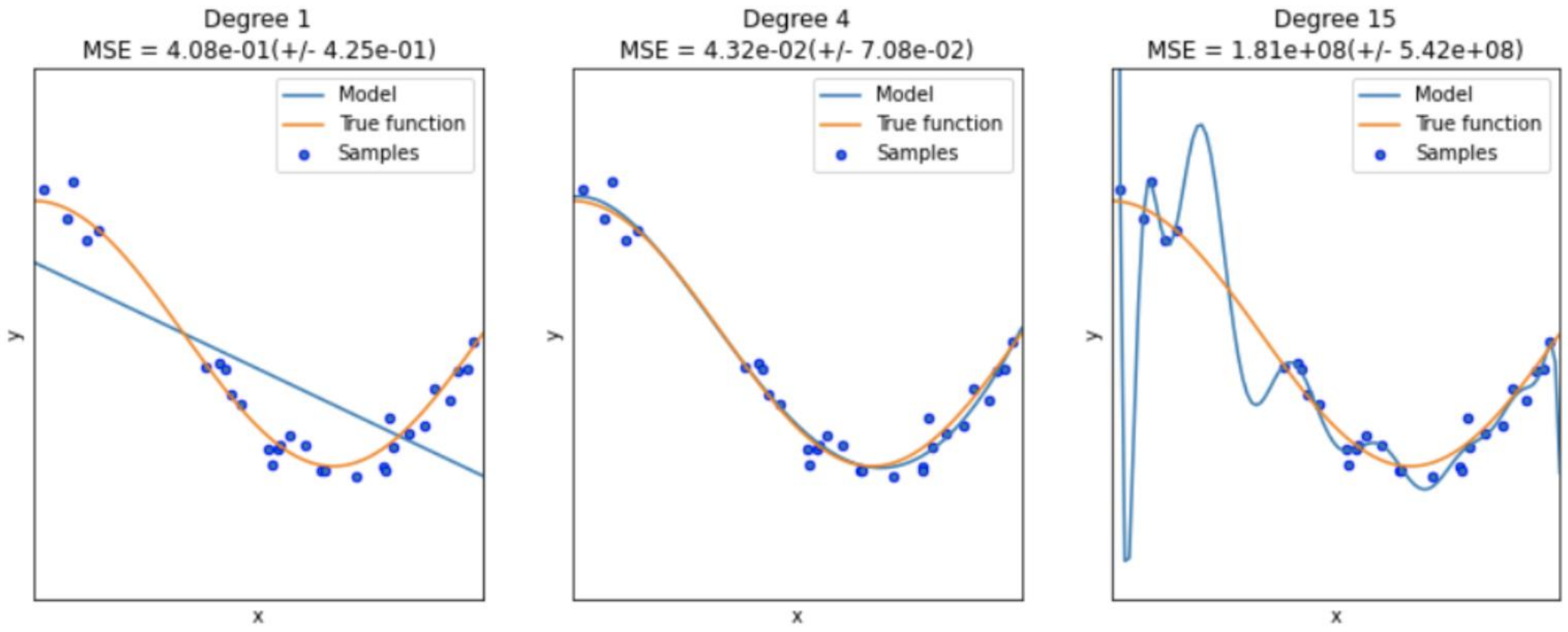
- Higher order => strictly larger class of function $\mathcal{F}$

- The following samples in $\mathbb{R}$ are not separable by halfspaces

- But, if we map $x \to (x, x^2)$ it is separable by halfspaces



Recall: Feature embedding trick



- overfitting vs underfitting
- bias-variance tradeoff.

$$\mathrm{err}(\hat{f}) \;=\; [\mathrm{err}(\hat{f}) - \min_{f^* \in \mathcal{F}} \mathrm{err}(f^*)] \;+\; \min_{f^* \in \mathcal{F}} \mathrm{err}(f^*)$$

from https://datascience.foundation/sciencewhitepaper/underfitting-and-overfitting-in-machine-learning

Kernel trick: high-level idea

- Given (possibly nonlinear) basis functions $\phi(x): \mathbb{R}^d \to \mathbb{R}^D$, where D is huge or infinite
- Would like to learn a model from class $\mathcal{F}_\phi = \{h: h(x) = \langle w, \phi(x) \rangle, \text{ for some } w \in \mathbb{R}^D\}$ with running time independent of D
- Computational Challenge:

  a naïve application of existing algorithms (e.g. SGD, Perceptron) has running time $\Omega(D)$
- Key structural assumption on ϕ:

  its induced *kernel function* $K(x, x') \coloneqq \langle \phi(x), \phi(x') \rangle$ can be evaluated in time independent of D
- How can we utilize this structure to address the challenge?

Kernel function: an example

- Let $|x| < 1$
- $\phi(x) = \left(1, x, x^2, x^3 \dots,\right) = (x^n)_{n=1}^{\infty}$
  - Impossible to write down explicitly
- Induced Kernel function:

$$K(x, y) \coloneqq \langle \phi(x), \phi(y) \rangle = \sum_{n=1}^{\infty} (x \cdot y)^n = \frac{1}{1 - xy}$$

  - Takes $O(1)$ time to calculate

How to use kernels with SVM

Primal

$$\min_{w,b} \|w\|^2 + C \sum_{i=1}^{n} \xi_i$$

$$s.t.\ \ y_i(w^\top \phi(x_i) + b) \geq 1 - \xi_i, \forall i$$

Dual

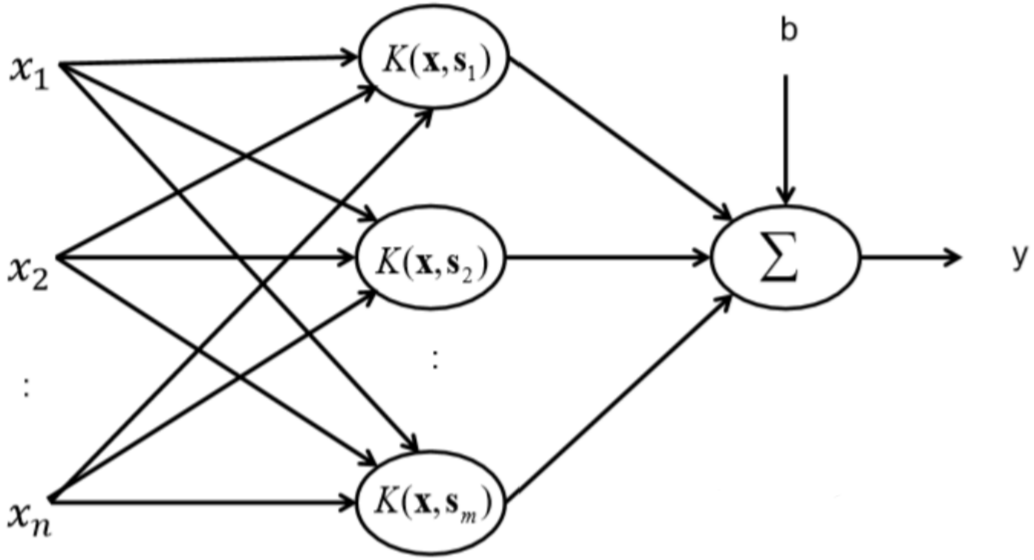
$$\max_{0 \leq \alpha \leq C} \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle \phi(x_i), \phi(x_j) \rangle$$

s.t. $\sum_i \alpha_i y_i = 0$

- To make predictions
  - Recall optimal $w^* = \sum_i \alpha_i^* y_i \phi(x_i)$
  - $\langle w^*, \phi(x) \rangle = \sum_i \alpha_i^* y_i \langle \phi(x_i), \phi(x) \rangle = \sum_i \alpha_i^* y_i K(x_i, x)$
  - Take $\text{sign}(\langle w^*, \phi(x) \rangle + b^*)$
  - Interpretation: α_i^*: weights, y_i: votes, $K(x_i, x_*)$: similarity

How to use kernels with SVM (cont'd)

$$h(x) = \text{sign}(\langle w^*, \phi(x)\rangle + b^*) = \text{sign}(\textstyle\sum_i \alpha_i^* y_i K(x_i, x) + b^*)$$



Cf. weighted k-NN

$$\hat{y}(x) = \arg\max_y \textstyle\sum_{i \in N(x_*)} w_i\, 1\{y_i = y\}$$
$$= \text{sign}(\textstyle\sum_{i \in N(x_*)} w_i\, y_i),$$

where e.g., $w_i = \exp\left(-\beta d(x_i, x)^2\right)$

- Summary
  - training: compute α_i^*'s and b^*
  - test: compute the kernel functions $K(x_i, x)$ and then take weighted combination $\sum_i \alpha_i^* y_i\, K(x_i, x) + b^*$

Polynomial kernels

- $K(x, x') = (1 + \langle x, x' \rangle)^k$ => Q: is this a valid kernel?

- E.g., if $x = (x_1, x_2)$ and $z = (z_1, z_2)$,

$$(1 + x_1 z_1 + x_2 z_2)^2 = 1 + x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1 z_1 + 2x_2 z_2 + 2x_1 x_2 z_2 z_1$$

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

where $\phi(x) = (x_1^2, x_2^2, \sqrt{2}\, x_1, \sqrt{2} x_2, \sqrt{2} x_1 x_2, 1)$

Polynomial kernels

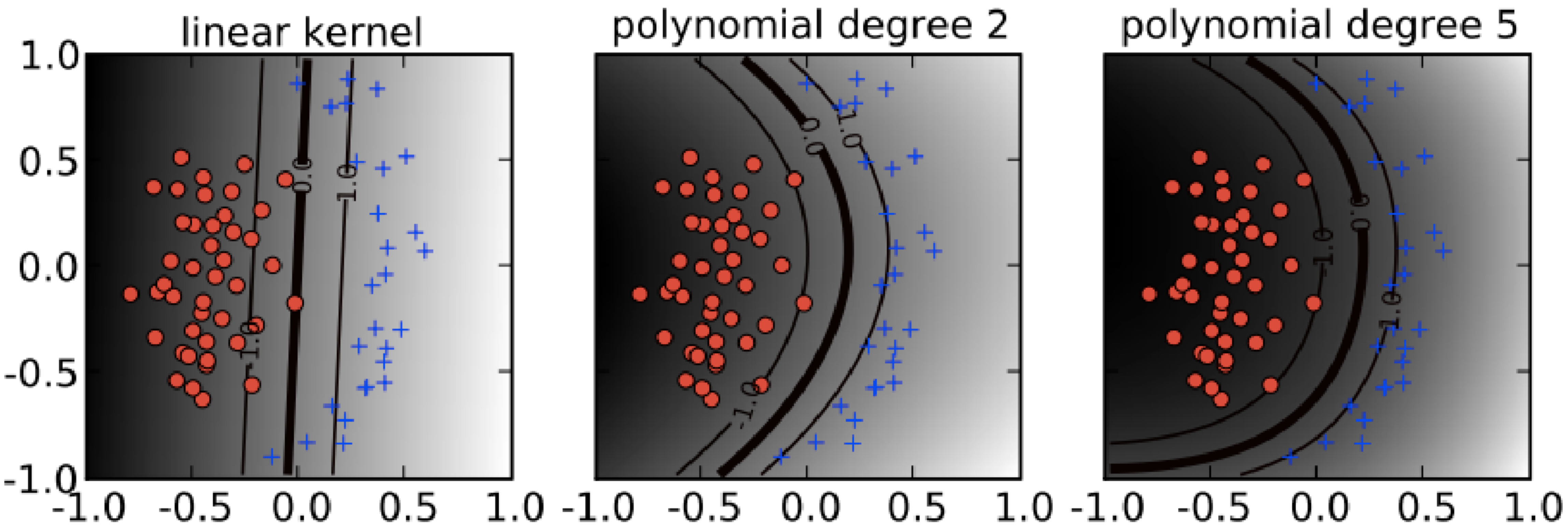


Figure from Ben-Hur & Weston,
Methods in Molecular Biology 2010

Gaussian/RBF kernels

- $K(x, x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$ (often parameterized as $\exp\left(-\gamma\|x - x'\|^2\right)$)
- How can we show that this is a valid kernel? => We should find $\phi(x)$ that results in $K(x, x') = \langle\phi(x), \phi(x')\rangle$

Assume $x \in R^1$ and $\gamma > 0$.

$$e^{-\gamma\|x_i-x_j\|^2} = e^{-\gamma(x_i-x_j)^2} = e^{-\gamma x_i^2+2\gamma x_i x_j-\gamma x_j^2}$$

$$=e^{-\gamma x_i^2-\gamma x_j^2}\left(1 + \frac{2\gamma x_i x_j}{1!} + \frac{(2\gamma x_i x_j)^2}{2!} + \frac{(2\gamma x_i x_j)^3}{3!} + \cdots\right)$$

$$=e^{-\gamma x_i^2-\gamma x_j^2}\left(1 \cdot 1+\sqrt{\frac{2\gamma}{1!}}x_i \cdot \sqrt{\frac{2\gamma}{1!}}x_j+\sqrt{\frac{(2\gamma)^2}{2!}}x_i^2 \cdot \sqrt{\frac{(2\gamma)^2}{2!}}x_j^2\right.$$

$$\left.+\sqrt{\frac{(2\gamma)^3}{3!}}x_i^3 \cdot \sqrt{\frac{(2\gamma)^3}{3!}}x_j^3 + \cdots\right) = \phi(x_i)^T\phi(x_j),$$

where

$$\phi(x) = e^{-\gamma x^2}\left[1, \sqrt{\frac{2\gamma}{1!}}x, \sqrt{\frac{(2\gamma)^2}{2!}}x^2, \sqrt{\frac{(2\gamma)^3}{3!}}x^3, \cdots\right]^T.$$

recall how we make **predictions**:

$w^\top x_* = \sum_i \alpha_i y_i x_i^\top x_* = \sum_i \alpha_i y_i K(x_i, x_*)$

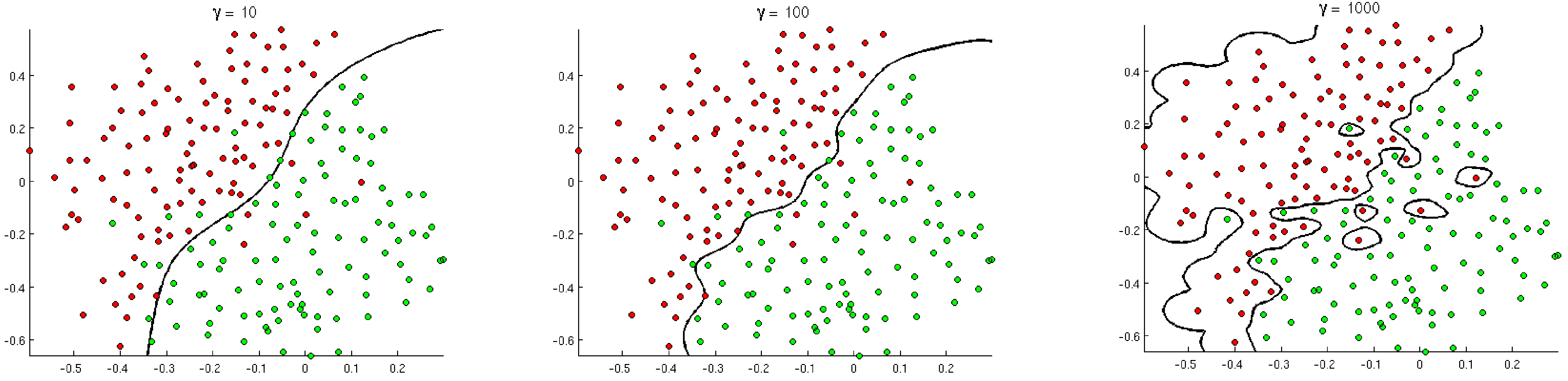
weighted k-NN

- $\arg\max_y \sum_{i\in N(x_*)} w_i\, 1\{y_i = y\}$
- e.g., $w_i = \exp\left(-\beta \cdot \left(d(x_i, x_*)\right)^2\right)$

(from https://www.csie.ntu.edu.tw/~cjlin/talks/kuleuven_svm.pdf)

Gaussian kernel

- $\gamma = \frac{1}{2\sigma^2}$



- Larger $\gamma \Rightarrow$ smaller $\sigma^2 \Rightarrow$ more likely to overfit
- A heuristic in practice: choose $\sigma = \mathrm{median}(\|x_i - x_j\|, i \neq j)$

How to recognize a valid kernel

- Two methods

  (1) Find an explicit feature representation $\phi(x)$

  (2) check Mercer's condition

cf. covariance matrix

- (Def) Let $K(x, x')$ be a kernel. Let $S = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^d$
  Then, $G := \left[K\left(x_i, x_j\right)\right]_{ij} \in \mathbb{R}^{n \times n}$ is called the **Gram matrix** of S

- (Thm) Let $K(x, x')$ be a symmetric function. Then,

$K(x, x') = \langle \phi(x), \phi(x') \rangle$ for some $\phi(x)$ $\iff$ K satisfies *Mercer's condition*: for any $\forall m \geq 1, \forall x_1, \ldots, x_m$, the **Gram matrix** of $S = \{x_1, \ldots, x_m\}$ is PSD.

Example

- Is $K(x, y) = \max(x, y)$ a valid kernel?
- After some failed trials of constructing ϕ, you may want to disprove that K is a kernel
- Suffices to show that K fails Mercer's condition, i.e. exists some dataset S whose Gram matrix is not PSD
- Guess $S = \{-1\} \Rightarrow G = (-1)$ not PSD

- What if we restrict the inputs $x, y \geq 0$?
- Guess $S = \{0, 2\} \ \Rightarrow G = \begin{pmatrix} K(0,0) & K(0,2) \\ K(2,0) & K(2,2) \end{pmatrix} = \begin{pmatrix} 0 & 2 \\ 2 & 2 \end{pmatrix}$
  - G is not PSD. Why?
  - Method 1: find v such that $v^\top G v < 0$
  - Method 2: check that some eigenvalues of G are < 0

Building Kernels from simpler ones

- Kernels are closed under
  - Positive scaling
  - sum/product
  - composition with a positive power series: $\sum_{i=1}^{\infty} a_i \left(K(x, x')\right)^i$, where $a_i \geq 0$ for all i

Building Kernels from simpler ones (cont'd)

kernel composition	mapping composition
$k(\boldsymbol{x},\boldsymbol{v}) = k_a(\boldsymbol{x},\boldsymbol{v}) + k_b(\boldsymbol{x},\boldsymbol{v})$	$\phi(\boldsymbol{x}) = \left(\phi_a(\boldsymbol{x}),\ \phi_b(\boldsymbol{x})\right)$
$k(\boldsymbol{x},\boldsymbol{v}) = \gamma\ k_a(\boldsymbol{x},\boldsymbol{v}),\ \gamma > 0$	$\phi(\boldsymbol{x}) = \sqrt{\gamma}\ \phi_a(\boldsymbol{x})$
$k(\boldsymbol{x},\boldsymbol{v}) = k_a(\boldsymbol{x},\boldsymbol{v}) k_b(\boldsymbol{x},\boldsymbol{v})$	$\phi_l(\boldsymbol{x}) = \phi_{ai}(\boldsymbol{x})\phi_{bj}(\boldsymbol{x})$
$k(\boldsymbol{x},\boldsymbol{v}) = \boldsymbol{x}^{\mathsf{T}} A \boldsymbol{v},\quad A$ is p.s.d.	$\phi(\boldsymbol{x}) = L^{\mathsf{T}}\boldsymbol{x},$ where $A = LL^{\mathsf{T}}$
$k(\boldsymbol{x},\boldsymbol{v}) = f(\boldsymbol{x}) f(\boldsymbol{v}) k_a(\boldsymbol{x},\boldsymbol{v})$	$\phi(\boldsymbol{x}) = f(\boldsymbol{x})\phi_a(\boldsymbol{x})$

Kernelized Perceptron algorithm

- How to combine the Perceptron algorithm with a nonlinear feature mapping $\phi: \mathcal{X} \to \mathbb{R}^D$?
- Recall the Perceptron algorithm:

Algorithm 29 PerceptronTrain(**D**, *MaxIter*)

$w \leftarrow 0$, $b \leftarrow 0$ // initialize weights and bias
for $iter = 1 \ldots MaxIter$ **do**
    for all $(x,y) \in$ **D** **do**
        $a \leftarrow w \cdot \phi(x) + b$ // compute activation for this example
        if $ya \leq 0$ **then**
            $w \leftarrow w + y\,\phi(x)$ // update weights
            $b \leftarrow b + y$ // update bias
        end if
    end for
end for
return w, b

- Suppose ϕ is associated with a kernel K
- Is it possible to implement this without ever explicitly computing ϕ?

Kernelized Perceptron algorithm

- Key observation: throughout the run of the Perceptron algorithm, w always lies in $\mathrm{span}(\phi(x_1), \dots, \phi(x_n))$, i.e.

$$w \text{ always has the form } \alpha_1\phi(x_1) + \cdots + \alpha_n\phi(x_n)$$

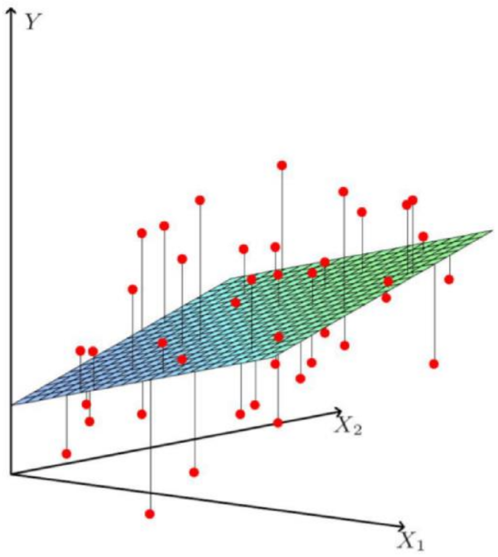
- Key algorithmic idea: instead of maintaining $w \in \mathbb{R}^D$, we maintain its linear combination coefficient $(\alpha_1, \dots, \alpha_n) \in \mathbb{R}^n$!

Algorithm 30 KernelizedPerceptronTrain($\mathbf{D}$, *MaxIter*)

$\boldsymbol{\alpha} \leftarrow \mathbf{0}$, $b \leftarrow 0$ // initialize coefficients and bias
for $iter = 1 \dots MaxIter$ **do**
  for all $(x_n, y_n) \in \mathbf{D}$ **do**
    $a \leftarrow \sum_m \alpha_n \boxed{\phi(x_m) \cdot \phi(x_n)} + b$ // compute activation for this example
    if $y_n a \leq 0$ **then**
$K(x_m, x_n)$
      $\alpha_n \leftarrow \alpha_n + y_n$ // update coefficients
      $b \leftarrow b + y$ // update bias
    end if
  end for
end for
return $\boldsymbol{\alpha}, b$

Kernelized ridge regression

- Recall ridge regression:, $\widehat{w} = \arg\min_{w} \|Xw\ - y\|^2 + \lambda \|w\|^2$



$$\begin{pmatrix} - & x_1 & - \\ & \dots & \\ - & x_n & - \end{pmatrix} \cdot w \approx \begin{pmatrix} y_1 \\ \dots \\ y_n \end{pmatrix}$$

$X: n \times d$ $\qquad$ $y: n \times 1$

- Woodbury matrix identity (matrix inversion lemma)

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U\left(C^{-1} + VA^{-1}U\right)^{-1} VA^{-1}.$$

- (Thm) $\widehat{w}$ can be alternatively written as $\widehat{w} = X^\top(\lambda I_n + \boxed{X\, X^\top})^{-1} y$

Gram matrix

(proof)

- starting point: recall $\widehat{w} = (\lambda I + \boxed{X^\top X})^{-1} X^\top y$ => apply the lemma above to $(\lambda I + X^\top X)^{-1}$

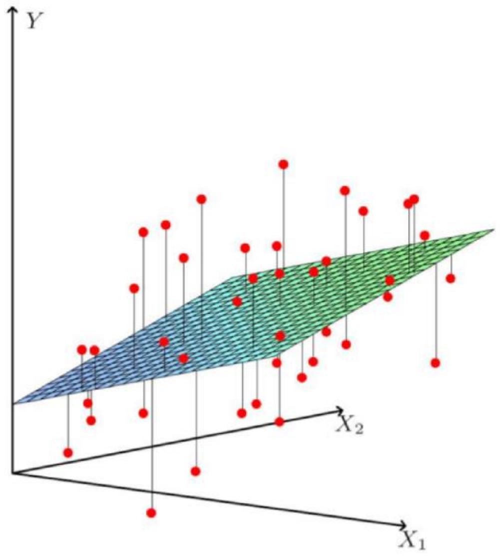
(scaled) covariance matrix

- tip: when you get stuck, try the special case of $d = 1$ to get a sense.

Kernelized ridge regression

$$\begin{pmatrix} -\ \phi(x_1)\ - \\ \dots \\ -\ \phi(x_n)\ - \end{pmatrix} \cdot w \approx \begin{pmatrix} y_1 \\ \dots \\ y_n \end{pmatrix}$$

$\Phi: n \times d$ $\quad$ $y: n \times 1$



- $\hat{w} = \Phi^\top(\lambda I + \Phi\Phi^\top)^{-1} y$
- Recall $\Phi\Phi^\top = \left(K(x_i, x_j)\right)_{i,j}$ is the Gram matrix
- **<u>Prediction for x:</u>**

$$\begin{aligned} \langle \hat{w}, \phi(x) \rangle &= \phi(x)^\top \hat{w} \\ &= \phi(x)^\top \Phi^\top (\lambda I + \Phi\Phi^\top)^{-1} y \\ &= (K(x, x_1), \dots, K(x, x_n)) \cdot \alpha, \text{ where } \alpha = (\lambda I + \Phi\Phi^\top)^{-1} y \end{aligned}$$

- Again, avoids explicit representation of $\phi(x_i)$'s

Time complexity of (kernelized) Perceptron & ridge regression

- ϕ: feature map from d dimensions to p dimensions
- n: training set size
- k: the number of operations to evaluate $K(x, x')$

- Test stage:

	Without kernel trick	With Kernel trick
Ridge regression	$O(p)$	$O(nk)$
Perceptron	$O(p)$	$O(nk)$

- Training stage:

	Without kernel trick	With Kernel Trick
Ridge regression	$O(np^2 + p^3)$	$O(n^2k + n^3)$
Perceptron	$O(\#\text{iters} \times p)$	$O(\#\text{iters} \times nk)$

Next lecture (10/3)

- Unsupervised learning

- Assigned reading: CIML 3.4 (Review) 11.3