Support Vector Machines Part 2

Yingyu Liang
Computer Sciences 760
Fall 2017

http://pages.cs.wisc.edu/~yliang/cs760/

Some of the slides in these lectures have been adapted/borrowed from materials developed by Mark Craven, David Page, Jude Shavlik, Tom Mitchell, Nina Balcan, Matt Gormley, Elad Hazan, Tom Dietterich, and Pedro Domingos.

Goals for the lecture

you should understand the following concepts

- soft margin SVM
- support vector regression
- the kernel trick
- polynomial kernel
- Gaussian/RBF kernel
- valid kernels and Mercer’s theorem
- kernels and neural networks

Variants: soft-margin and SVR

Hard-margin SVM

- Optimization (Quadratic Programming):

$$\min_{w,b} \frac{1}{2}||w||^2$$

$$y_i(w^T x_i + b) \geq 1, \forall i$$

Soft-margin SVM [Cortes & Vapnik, *Machine Learning* 1995]

- if the training instances are not linearly separable, the previous formulation will fail
- we can adjust our approach by using *slack variables* (denoted by ζ_i) to tolerate errors

$$\min_{w,b,\zeta_i} \frac{1}{2}||w||^2 + C\sum_i \zeta_i$$

$$y_i(w^T x_i + b) \geq 1 - \zeta_i, \zeta_i \geq 0, \forall i$$

- C determines the relative importance of maximizing margin vs. minimizing slack

The effect of C in soft-margin SVM

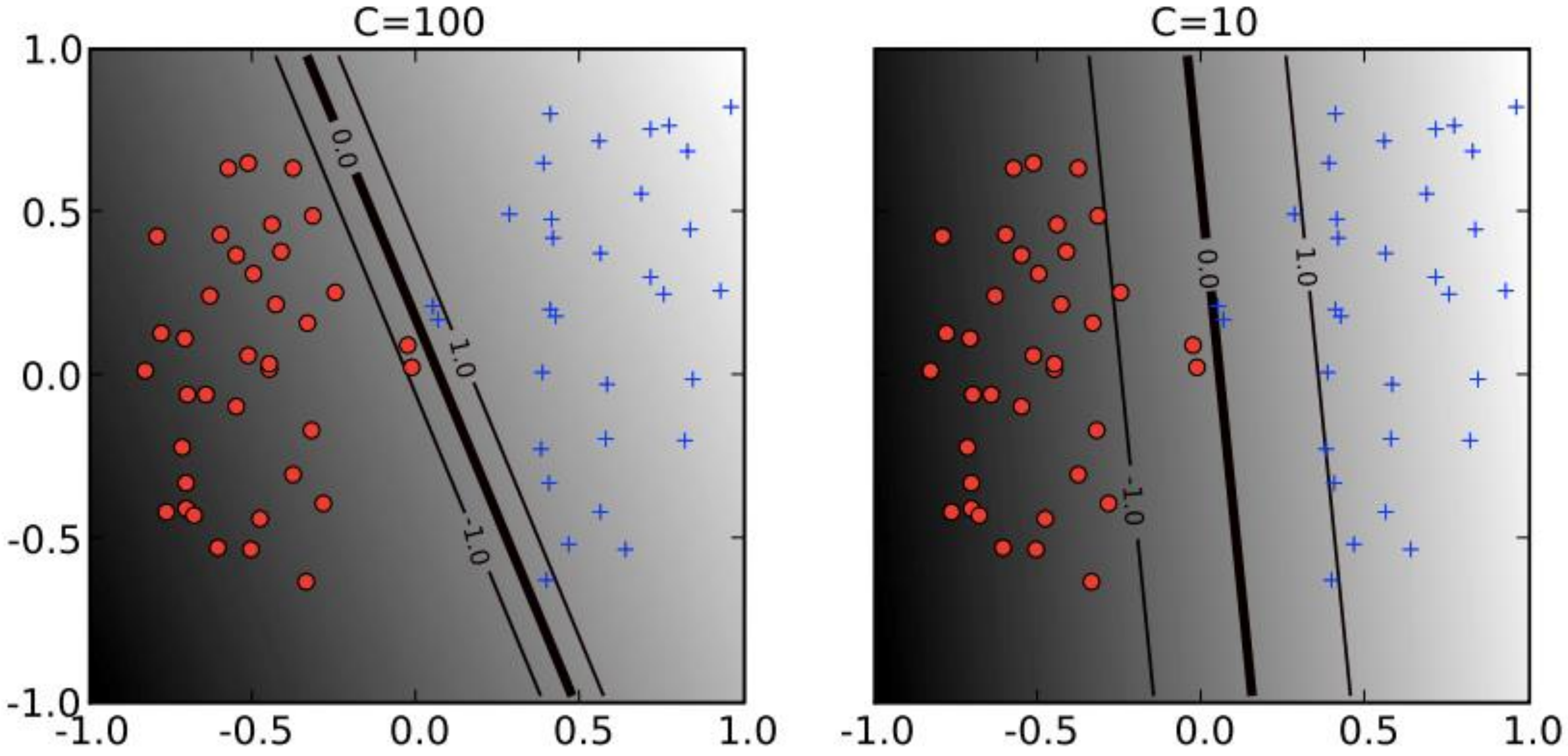
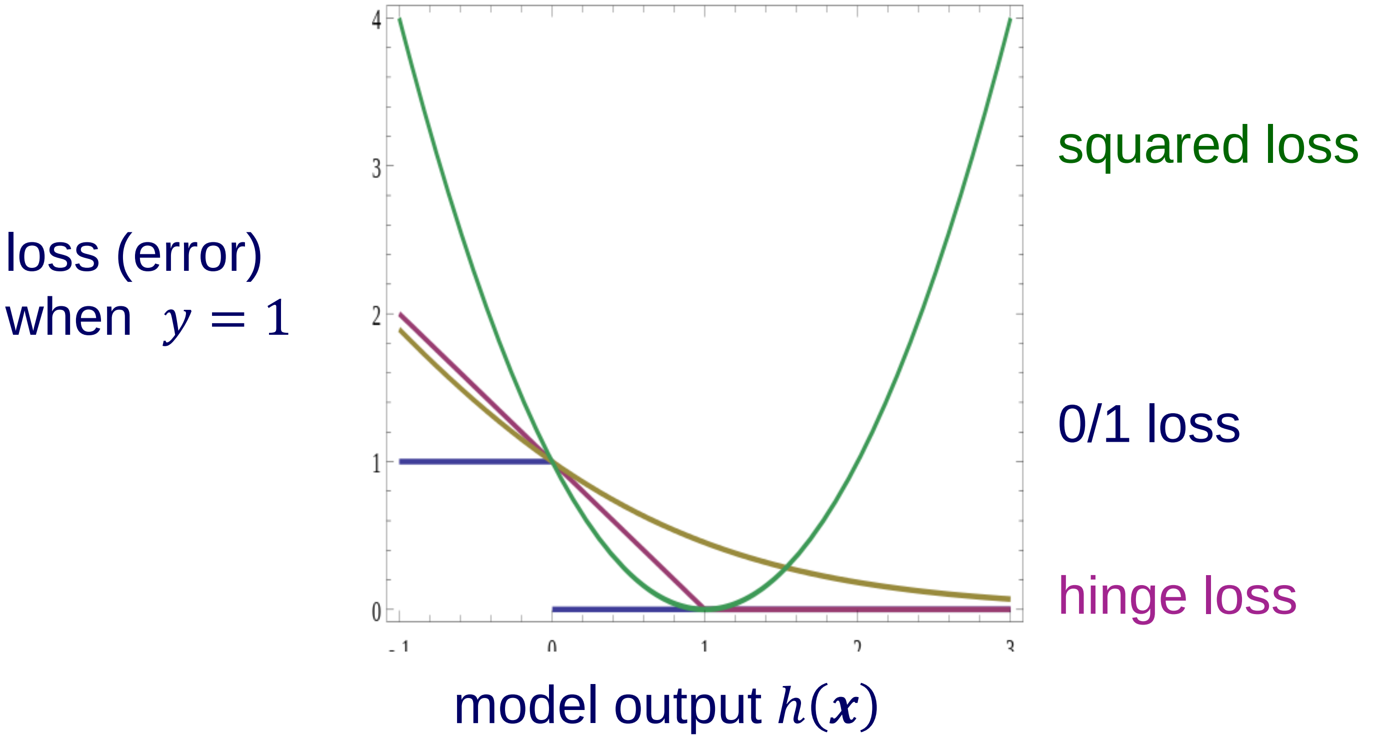


Figure from Ben-Hur & Weston,
Methods in Molecular Biology 2010

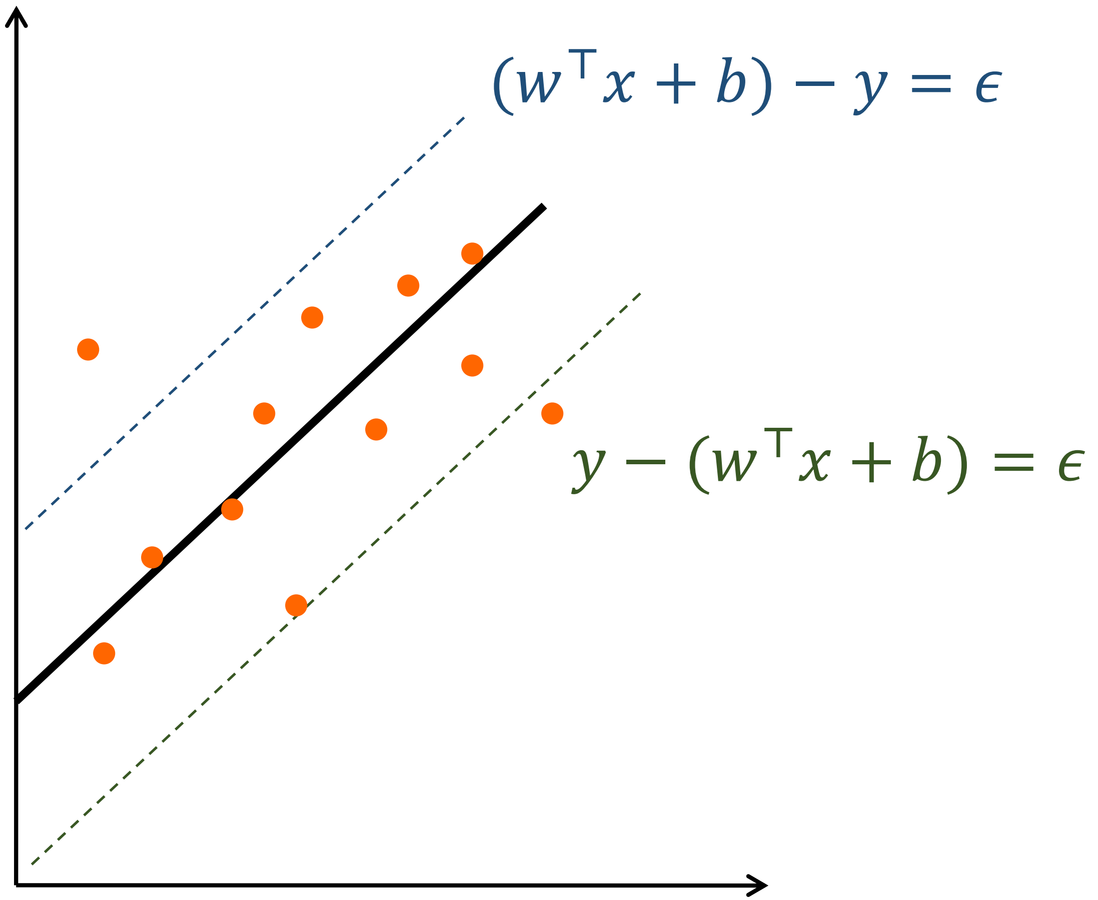
Hinge loss

- when we covered neural nets, we talked about minimizing squared loss and cross-entropy loss
- SVMs minimize *hinge loss*



Support Vector Regression

- the SVM idea can also be applied in regression tasks
- an ϵ-insensitive error function specifies that a training instance is well explained if the model's prediction is within ϵ of y_i



Support Vector Regression

- Regression using *slack variables* (denoted by ζ_i, ξ_i) to tolerate errors

$$\min_{w,b,\zeta_i,\xi_i} \frac{1}{2}||w||^2 + C\sum_i \zeta_i + \xi_i$$

$$(w^T x_i + b) - y_i \leq \epsilon + \zeta_i,$$
$$y_i - (w^T x_i + b) \leq \epsilon + \xi_i,$$
$$\zeta_i, \xi_i \geq 0.$$

slack variables allow predictions for some training instances to be off by more than ϵ

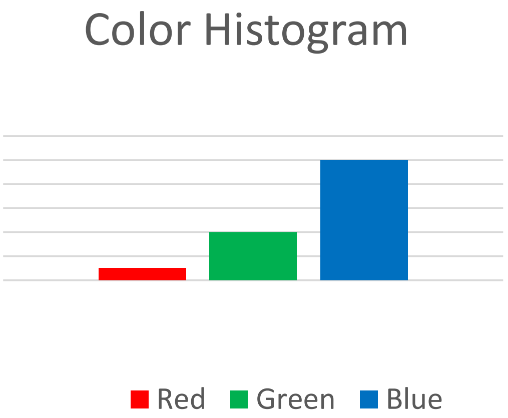
Kernel methods

Features

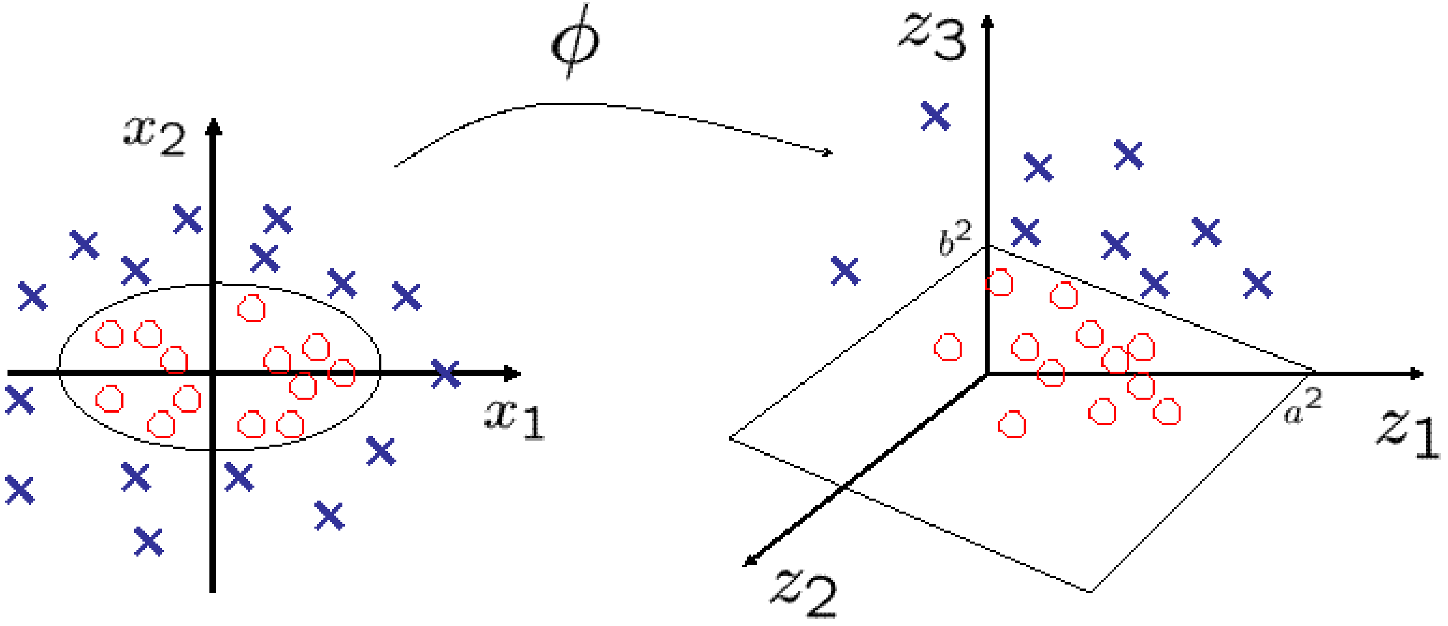
x

Extract features

$\phi(x)$



Features



$$\phi : (x_1, x_2) \longrightarrow (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

$$\left(\frac{x_1}{a}\right)^2 + \left(\frac{x_2}{b}\right)^2 = 1 \longrightarrow \frac{z_1}{a^2} + \frac{z_3}{b^2} = 1$$

Proper feature mapping can make non-linear to linear!

Recall: SVM dual form

Only depend on inner products

- Reduces to dual problem:

$$\mathcal{L}(w, b, \boldsymbol{\alpha}) = \sum_i \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$\sum_i \alpha_i y_i = 0, \alpha_i \geq 0$$

- Since $w = \sum_i \alpha_i y_i x_i$, we have $w^T x + b = \sum_i \alpha_i y_i x_i^T x + b$

Features

- Using SVM on the feature space $\{\phi(x_i)\}$: only need $\phi(x_i)^T\phi(x_j)$

- Conclusion: no need to design $\phi(\cdot)$, only need to design

$$k(x_i, x_j) = \phi(x_i)^T\phi(x_j)$$

Polynomial kernels

- Fix degree d and constant c:

$$k(x, x') = (x^T x' + c)^d$$

- What are $\phi(x)$?
- Expand the expression to get $\phi(x)$

Polynomial kernels

$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^2, \quad K(\mathbf{x}, \mathbf{x}') = (x_1 x_1' + x_2 x_2' + c)^2 = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}\, x_1 x_2 \\ \sqrt{2c}\, x_1 \\ \sqrt{2c}\, x_2 \\ c \end{bmatrix} \cdot \begin{bmatrix} {x'}_1^2 \\ {x'}_2^2 \\ \sqrt{2}\, x_1' x_2' \\ \sqrt{2c}\, x_1' \\ \sqrt{2c}\, x_2' \\ c \end{bmatrix}$$

Figure from Foundations of Machine Learning, by M. Mohri, A. Rostamizadeh, and A. Talwalkar

SVMs with polynomial kernels

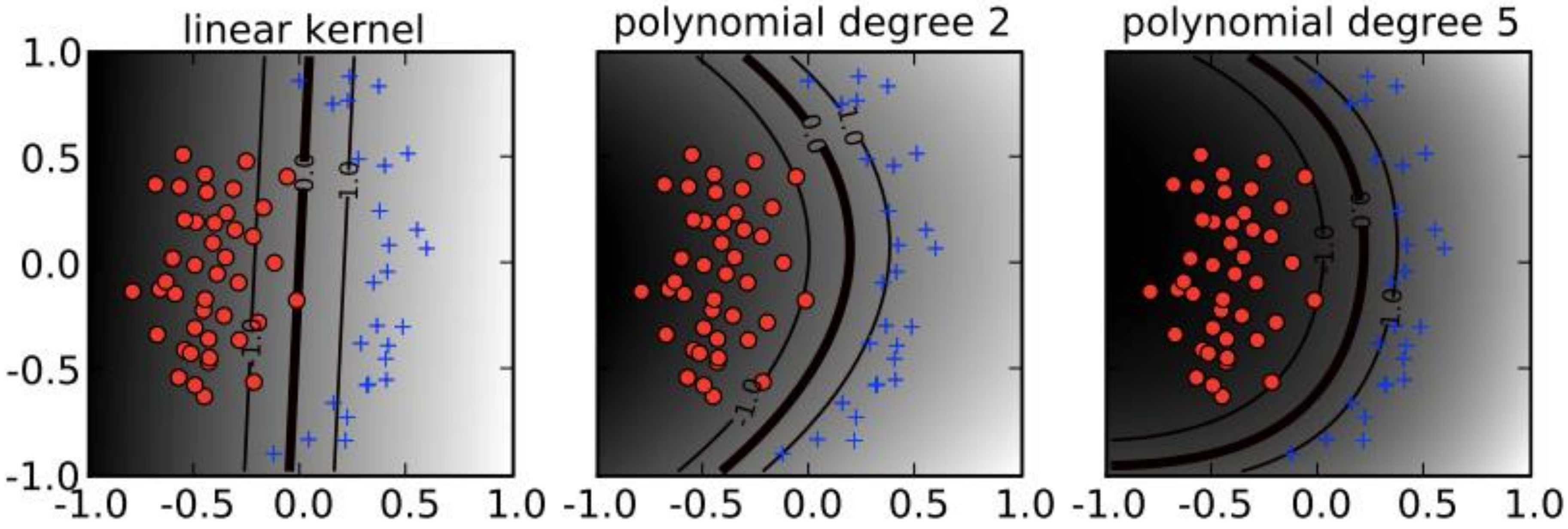


Figure from Ben-Hur & Weston,
Methods in Molecular Biology 2010

Gaussian/RBF kernels

- Fix bandwidth σ:

$$k(x, x') = \exp(-||x - x'||^2 / 2\sigma^2)$$

- Also called radial basis function (RBF) kernels

- What are $\phi(x)$? Consider the un-normalized version

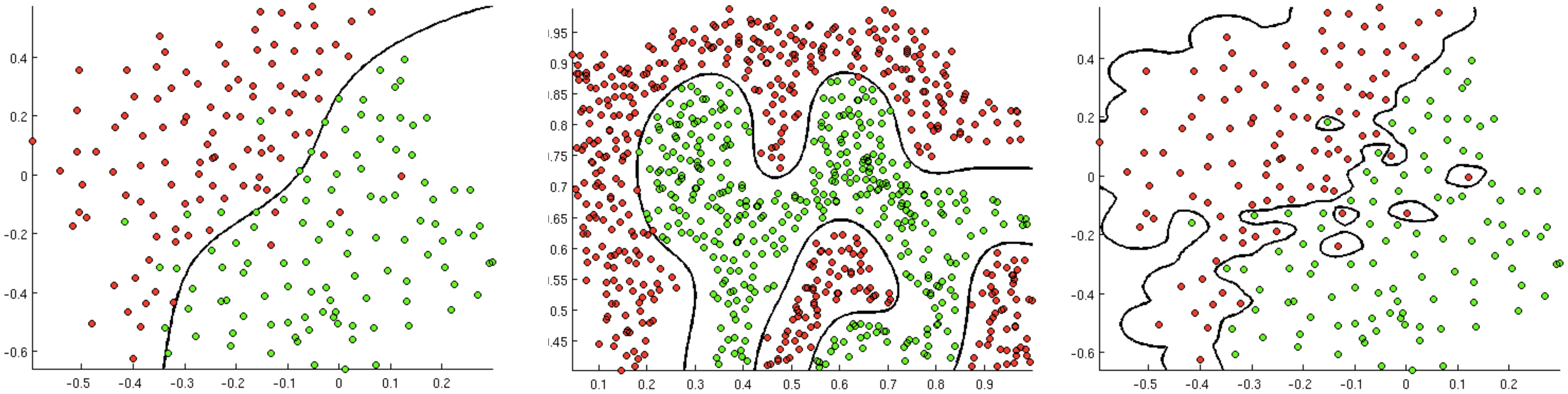
$$k'(x, x') = \exp(x^T x' / \sigma^2)$$

- Power series expansion:

$$k'(x, x') = \sum_{i}^{+\infty} \frac{(x^T x')^i}{\sigma^i i!}$$

The RBF kernel illustrated

$\gamma = -10$ $\gamma = -100$ $\gamma = -1000$



Figures from openclassroom.stanford.edu (Andrew Ng)

Mercer's condition for kenerls

- Theorem: $k(x, x')$ has expansion

$$k(x, x') = \sum_{i}^{+\infty} a_i \phi_i(x) \phi_i(x')$$

if and only if for any function $c(x)$,

$$\int \int c(x)c(x')k(x, x')dxdx' \geq 0$$

(Omit some math conditions for k and c)

Constructing new kernels

- Kernels are closed under positive scaling, sum, product, pointwise limit, and composition with a power series $\sum_i^{+\infty} a_i k^i(x, x')$

- Example: $k_1(x, x'), k_2(x, x')$ are kernels, then also is

$$k(x, x') = 2k_1(x, x') + 3k_2(x, x')$$

- Example: $k_1(x, x')$ is kernel, then also is

$$k(x, x') = \exp(k_1(x, x'))$$

Kernel algebra

- given a valid kernel, we can make new valid kernels using a variety of operators

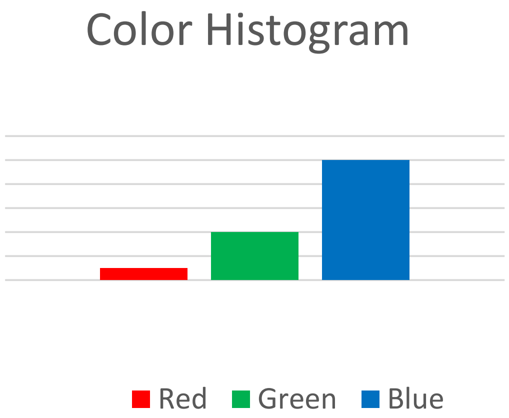
kernel composition	mapping composition
$k(\boldsymbol{x},\boldsymbol{v}) = k_a(\boldsymbol{x},\boldsymbol{v}) + k_b(\boldsymbol{x},\boldsymbol{v})$	$f(\boldsymbol{x}) = \left(f_a(\boldsymbol{x}),\ f_b(\boldsymbol{x})\right)$
$k(\boldsymbol{x},\boldsymbol{v}) = g\, k_a(\boldsymbol{x},\boldsymbol{v}),\ g > 0$	$f(\boldsymbol{x}) = \sqrt{g}\ f_a(\boldsymbol{x})$
$k(\boldsymbol{x},\boldsymbol{v}) = k_a(\boldsymbol{x},\boldsymbol{v}) k_b(\boldsymbol{x},\boldsymbol{v})$	$f_l(\boldsymbol{x}) = f_{ai}(\boldsymbol{x}) f_{bj}(\boldsymbol{x})$
$k(\boldsymbol{x},\boldsymbol{v}) = \boldsymbol{x}^{\top} A \boldsymbol{v},\quad A \text{ is p.s.d.}$	$\phi(\boldsymbol{x}) = L^{\top}\boldsymbol{x},\ \text{where } A = LL^{\top}$
$k(\boldsymbol{x},\boldsymbol{v}) = f(\boldsymbol{x}) f(\boldsymbol{v}) k_a(\boldsymbol{x},\boldsymbol{v})$	$f(\boldsymbol{x}) = f(\boldsymbol{x}) f_a(\boldsymbol{x})$

Kernels v.s. Neural networks

Features

x

Extract features

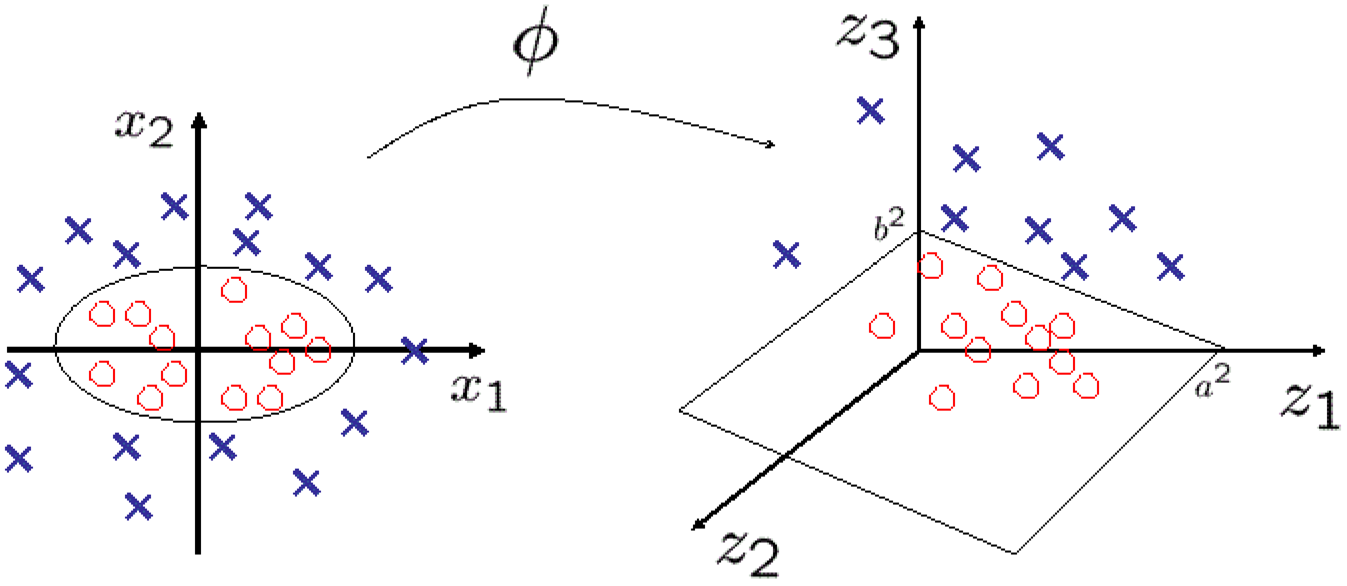


build hypothesis

$$y = w^T \phi(x)$$

Features: part of the model

Nonlinear model



build hypothesis

$$y = w^T \phi(x)$$

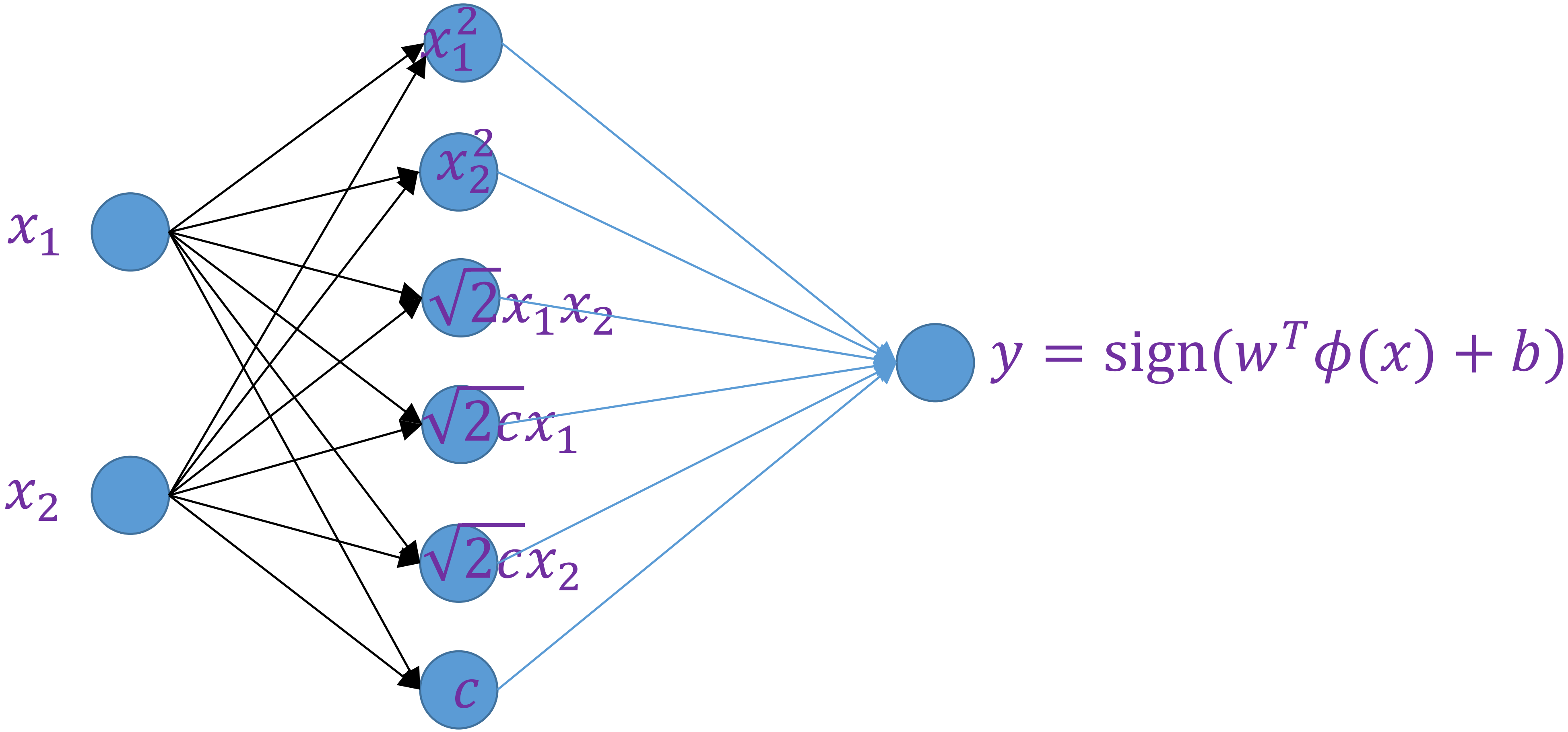
Linear model

Polynomial kernels

$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^2, \quad K(\mathbf{x}, \mathbf{x}') = (x_1 x_1' + x_2 x_2' + c)^2 = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}\, x_1 x_2 \\ \sqrt{2c}\, x_1 \\ \sqrt{2c}\, x_2 \\ c \end{bmatrix} \cdot \begin{bmatrix} {x'}_1^2 \\ {x'}_2^2 \\ \sqrt{2}\, x_1' x_2' \\ \sqrt{2c}\, x_1' \\ \sqrt{2c}\, x_2' \\ c \end{bmatrix}$$

Figure from Foundations of Machine Learning, by M. Mohri, A. Rostamizadeh, and A. Talwalkar

Polynomial kernel SVM as two layer neural network



First layer is fixed. If also learn first layer, it becomes two layer neural network

Comments on SVMs

- we can find solutions that are globally optimal (maximize the margin)
  - because the learning task is framed as a convex optimization problem
  - no local minima, in contrast to multi-layer neural nets

- there are two formulations of the optimization: *primal* and *dual*
  - dual represents classifier decision in terms of support vectors
  - dual enables the use of kernel functions

- we can use a wide range of optimization methods to learn SVM
  - standard quadratic programming solvers
  - SMO [Platt, 1999]
  - linear programming solvers for some formulations
  - etc.

Comments on SVMs

- kernels provide a powerful way to
  - allow nonlinear decision boundaries
  - represent/compare complex objects such as strings and trees
  - incorporate domain knowledge into the learning task
- using the kernel trick, we can implicitly use high-dimensional mappings without explicitly computing them
- one SVM can represent only a binary classification task; multi-class problems handled using multiple SVMs and some encoding
- empirically, SVMs have shown (close to) state-of-the art accuracy for many tasks
- the kernel idea can be extended to other tasks (anomaly detection, regression, etc.)