CS 760: Machine Learning
SVMs and Kernels

University of Wisconsin-Madison

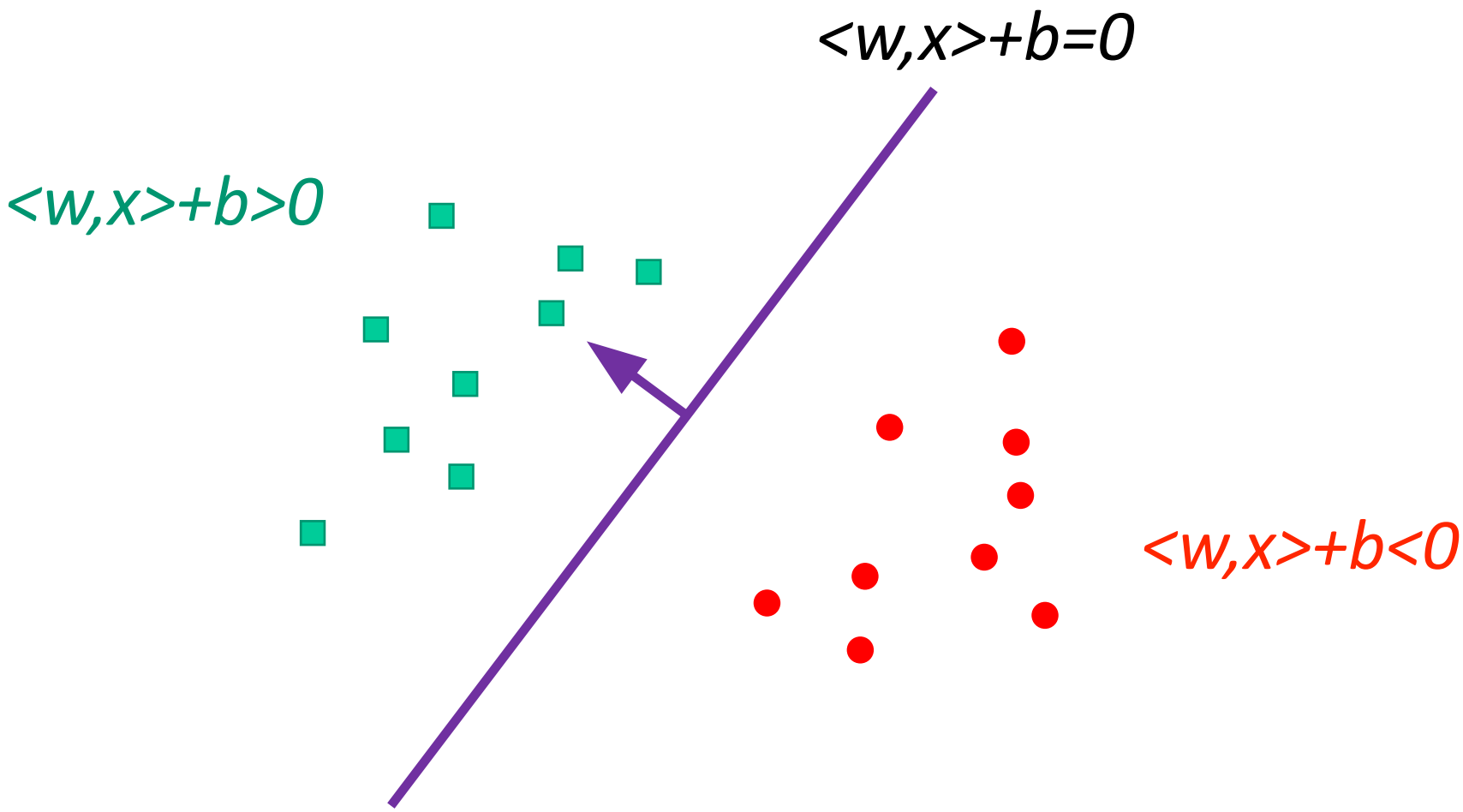
Outline

- Support Vector Machines (SVMs)
  - margins, training objectives
- Dual Formulation
  - Lagrangian, primal and dual problems
- Kernels
  - Feature maps, kernel trick, conditions

Outline

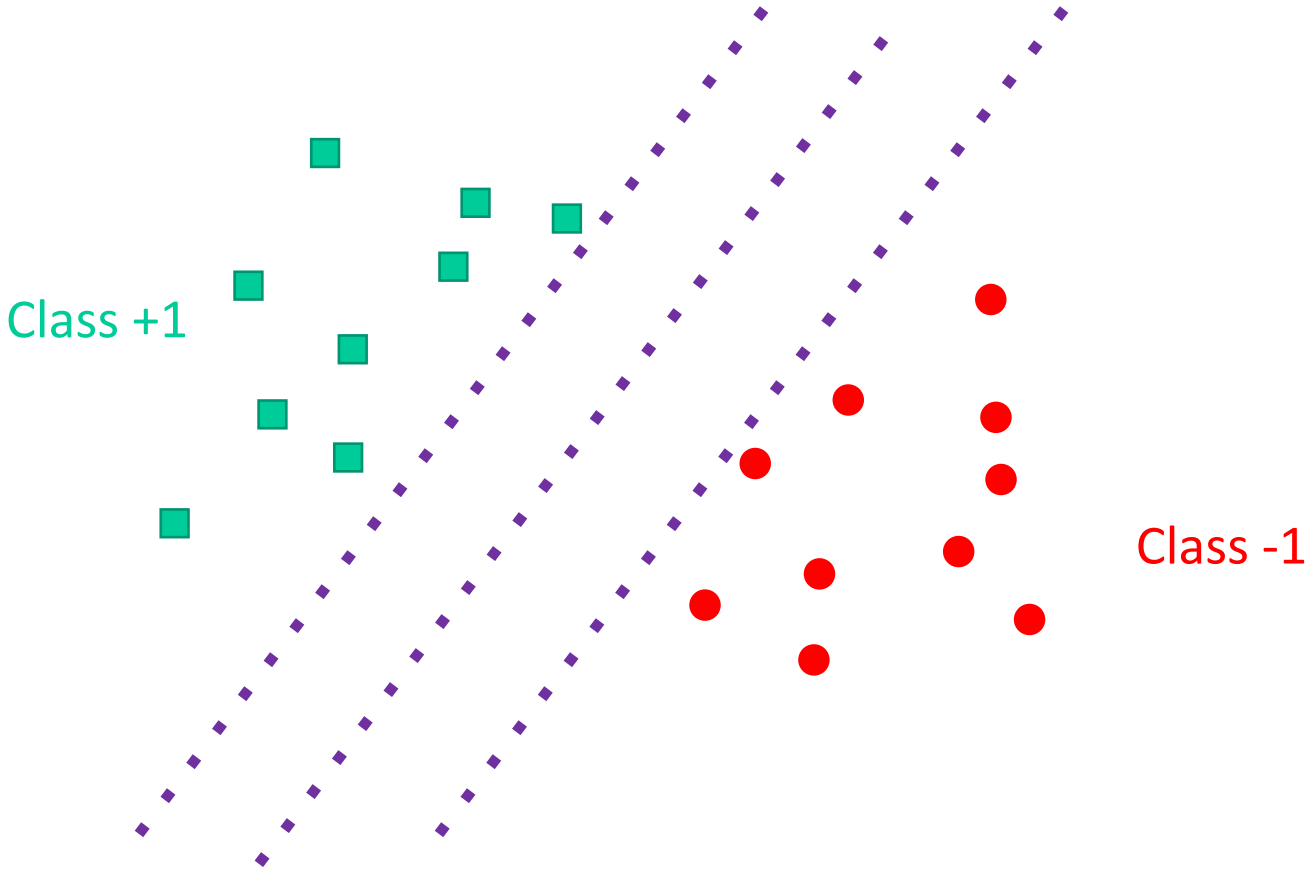
- **Support Vector Machines (SVMs)**
  - margins, training objectives
- **Dual Formulation**
  - Lagrangian, primal and dual problems
- **Kernels**
  - Feature maps, kernel trick, conditions

Linear classification revisited



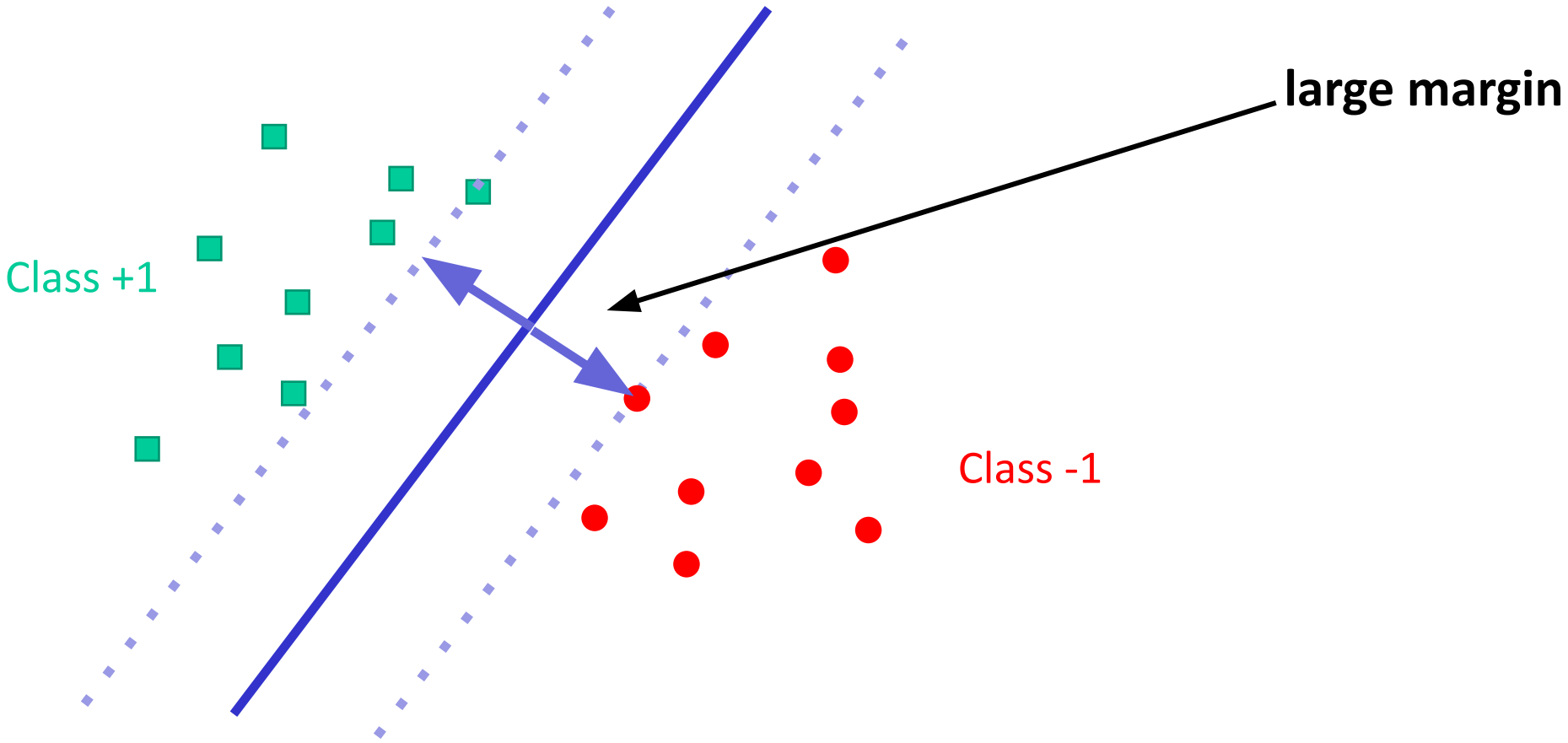
Linear classification revisited

- Which classifier is better for generalization?



Linear classification revisited

- Intuitively, expect a **large margin** to generalize better



- In fact, this intuition can be made formal!

Large-margin generalization

Informal theorem: if all input points $x \in X$ have norm ≤ 1 then w.p. $\geq 1 - \delta$ all linear models $h(x) = w^\top x$ with $\|w\| \leq 1$ have

$$generalization\ error \leq \frac{2}{\rho\sqrt{m}} + \sqrt{\frac{1}{2m}\log\frac{1}{\delta}}$$

Compare to the VC bound for d-dimensional linear classifiers:

$$generalization\ error \leq \sqrt{\frac{2(d+1)}{m}\log\frac{em}{d+1}} + \sqrt{\frac{1}{m}\log\frac{1}{\delta}}$$

If the margin $\rho = \Omega(1/\sqrt{d})$ the first is a much better guarantee!

Perhaps we should train classifiers to have a large margin?

Recall: Distance to a hyperplane

x has distance $\frac{|f_{w,b}(x)|}{\|w\|}$ to the hyperplane $f_{w,b}(z) = w^\top z + b = 0$

Support Vector Machines

The SVM idea: maximize the "minimum margin" over all training points:

$$\gamma(w,b) = \min_i \frac{|f_{w,b}(x_i)|}{\|w\|}$$

Equivalently:

$$\gamma(w,b) = \min_i \frac{y_i f_{w,b}(x_i)}{\|w\|}, \qquad y_i \in \{\pm 1\}$$

If $f_{w,b}$ incorrect on some x_i, the margin is **negative**

Support Vector Machines: Candidate Goal

Assume data is linearly separable (for now)

Objective idea 1: maximize margin over all training data points:

$$\max_{w,b} \gamma(w,b) = \max_{w,b} \min_{i} \frac{y_i f_{w,b}(x_i)}{\|w\|} = \max_{w,b} \min_{i} \frac{y_i (w^\top x_i + b)}{\|w\|}$$

Minimax Optimization may be difficult to solve!
(recall optimization difficulties with GANs)

SVM: Simplified Goal

Observation: when (w, b) scaled by a factor $c > 0$, the margin is unchanged

$$\frac{y_i(cw^T x_i + cb)}{\|cw\|} = \frac{y_i(w^T x_i + b)}{\|w\|}$$

Let us consider a fixed scale such that

$$y_{i^*}(w^T x_{i^*} + b) = 1$$

where x_{i^*} is the point closest to the hyperplane

SVM: Simplified Goal

Let us consider a fixed scale such that

$$y_{i^*}(w^T x_{i^*} + b) = 1$$

where x_{i^*} is the point closest to the hyperplane

Then for all points i we have $y_i(w^T x_i + b) \geq 1$, and the inequality is tight for at least one i

Then the margin over all training points is $\frac{|w^\top x_{i^*} + b|}{\|w\|} = \frac{1}{\|w\|}$

Writing the SVM as an optimization problem

Objective idea 2:

$$\max_{w,b} \quad \frac{1}{\|w\|} \quad \text{subject to} \quad y_i(w^\top x_i + b) \geq 1 \; \forall \, i$$

Rewrite as

$$\min_{w,b} \quad \tfrac{1}{2}\|w\|^2 \quad \text{subject to} \quad y_i(w^\top x_i + b) \geq 1 \; \forall \, i$$

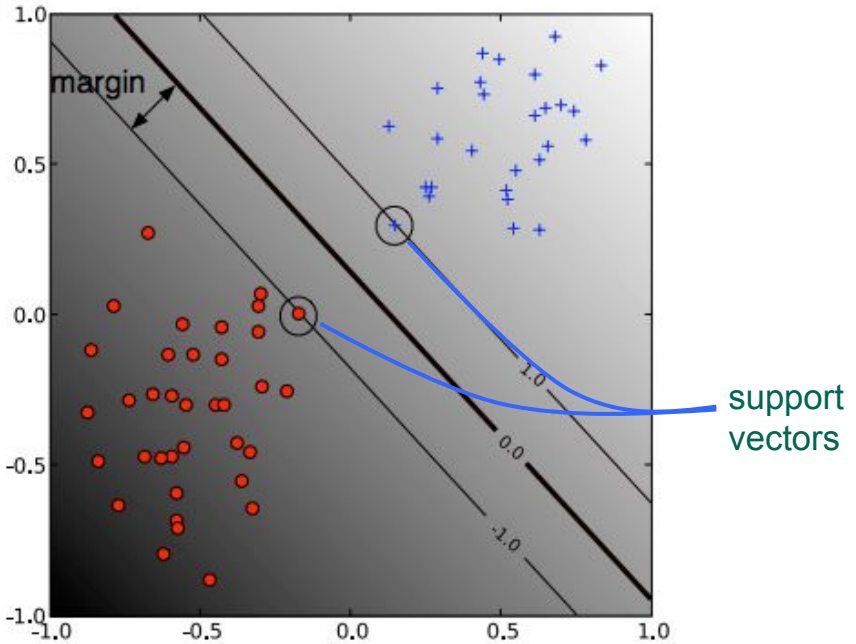
Why?

- It's a convex quadratic program, for which there are many efficient solvers.
- Can apply the kernel trick for **nonlinear** classification (coming up)

So why are they called support vector machines?

Instances where inequality is tight are the ***support vectors***

- Lie on the margin boundary
- Solution does not change if we delete other instances!



Ben-Hur & Weston, *Methods in Molecular Biology* 2010

SVM: Soft Margin

What if our data isn't linearly separable?

- Adjust approach by adding *slack variables* (denoted by ζ_i) to tolerate errors

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

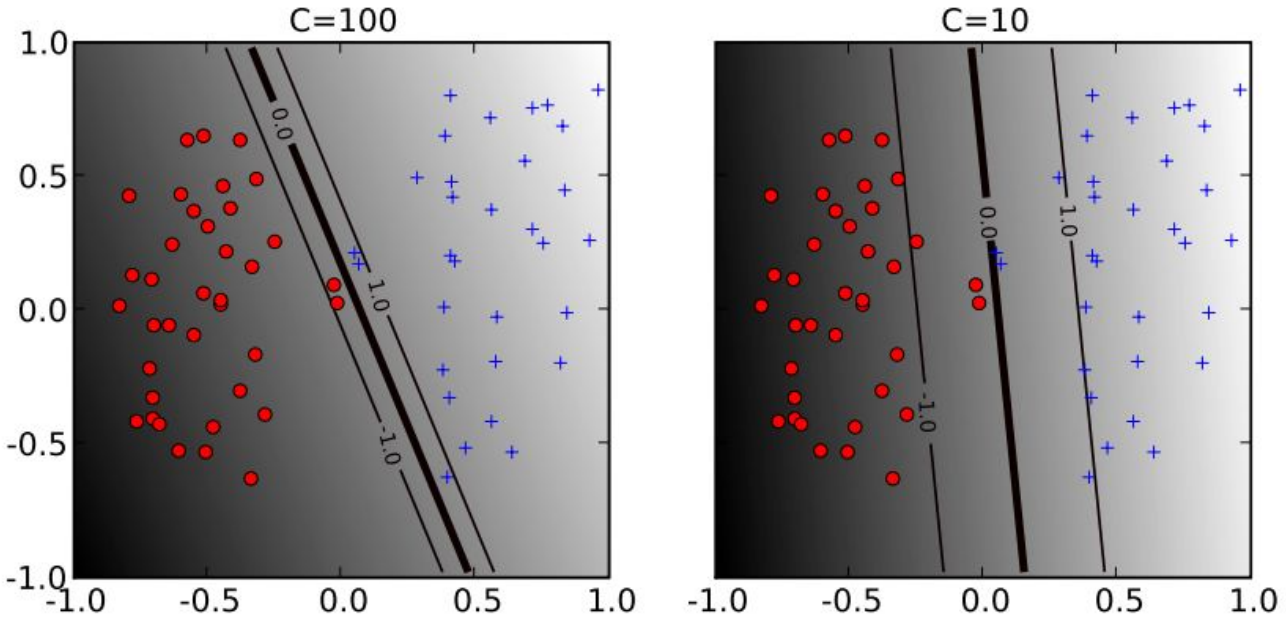
$$y_i(w^T x_i + b) \geq 1 - \zeta_i, \zeta_i \geq 0, \forall\ i$$

- adds a hyperparameter $C \geq 0$
  - trades-off maximizing margin vs. minimizing slack
  - roughly an inverse regularization parameter

SVM: Soft Margin

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

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



Ben-Hur & Weston, *Methods in Molecular Biology* 2010

Outline

- **Support Vector Machines (SVMs)**
  - margins, training objectives
- **Dual Formulation**
  - Lagrangian, primal and dual problems
- **Kernels**
  - Feature maps, kernel trick, conditions

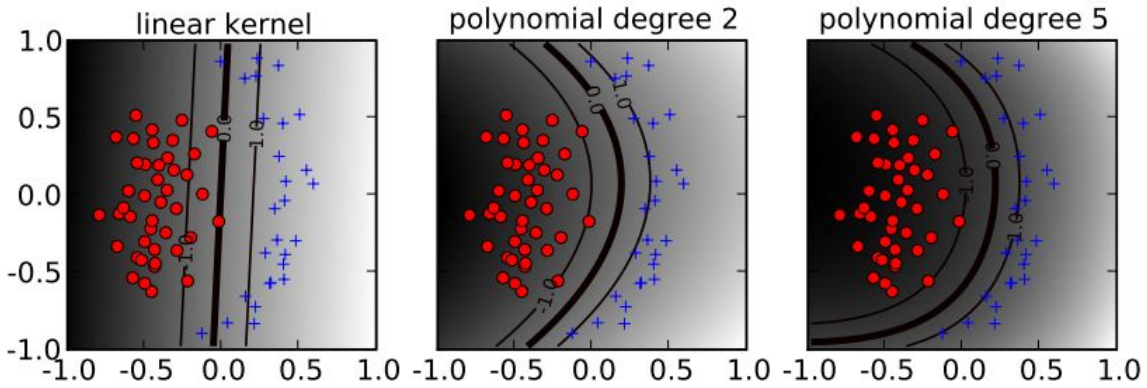
What if we have **nonlinearly** separated data?

Issue: sometimes the data is well-separated but not in a linear way

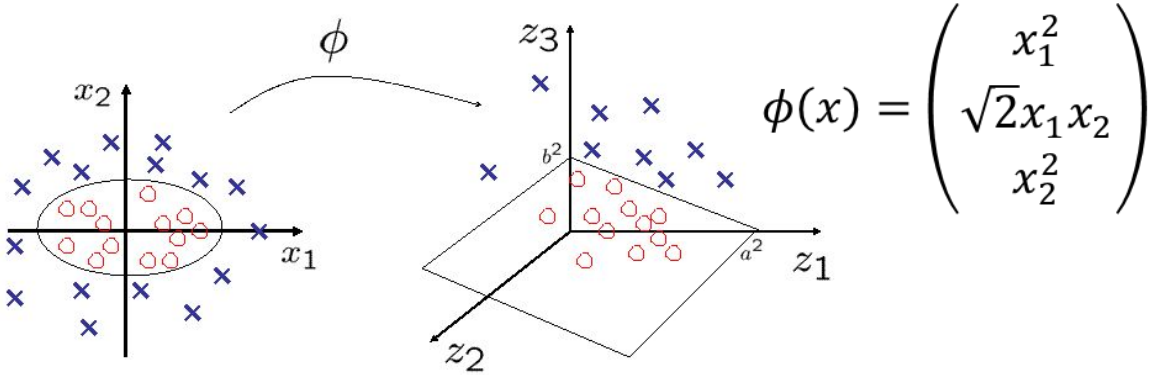
Solution: classify in a higher-dimensional space using a **feature map**

Issue: what if the dimension of the space is too high to represent efficiently?

Solution: reformulate the optimization problem to only depend on the **similarity between points**



Ben-Hur & Weston, *Methods in Molecular Biology* 2010



$$\phi(x) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$

$$K(x, x') = \phi(x)^\top \phi(x')$$

Brief introduction: Constrained optimization

- Consider the following problem:

$$\min_{w} f(w) \quad \leftarrow \text{Objective}$$

$$g_i(w) \leq 0, \forall 1 \leq i \leq k$$
$$h_j(w) = 0, \forall 1 \leq j \leq l \quad \} \text{ Constraints}$$

- It is associated with the **generalized Lagrangian**:

$$\mathcal{L}(w, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(w) + \sum_i \alpha_i g_i(w) + \sum_j \beta_j h_j(w)$$

where $\alpha_i \geq 0, \beta_j$'s are called **Lagrange multipliers**

Why do we care about the Lagrangian?

We can rewrite the original optimization problem as:

$$\min_{\substack{g_i(w)\leq 0 \\ h_j(w)=0}} f(w) = \min_{w} \max_{\alpha_i \geq 0, \beta_j} \mathcal{L}(w, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

Why?

$$\max_{\alpha_i \geq 0, \beta_j} \mathcal{L}(w, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \begin{cases} f(w) & \text{if } w \text{ satisfies all constraints} \\ +\infty & \text{otherwise} \end{cases}$$

Recall the constraints $g_i(w) \leq 0, h_j(w) = 0$ and the Lagrangian

$$\mathcal{L}(w, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(w) + \sum_i \alpha_i g_i(w) + \sum_j \beta_j h_j(w)$$

Why do we care about the Lagrangian?

We can rewrite the original optimization problem as:

$$\min_{\substack{g_i(w)\leq 0 \\ h_j(w)=0}} f(w) = \min_{w} \max_{\alpha_i \geq 0, \beta_j} \mathcal{L}(w, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

This **primal problem** is associated with a **dual problem**:

$$\max_{\alpha_i \geq 0, \beta_j} \min_{w} \mathcal{L}(w, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

Under certain assumptions (which hold for SVMs):

- the primal and dual problems have the same optimal value
- **we can solve one by solving the other**

Why do we care about the Lagrangian?

Under Slater's condition we have

$$\text{(primal)} \quad \min_{w} \max_{\alpha_i \geq 0, \beta_j} \mathcal{L}(w, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \max_{\alpha_i \geq 0, \beta_j} \min_{w} \mathcal{L}(w, \boldsymbol{\alpha}, \boldsymbol{\beta}) \quad \text{(dual)}$$

Why is this **duality** powerful?

- dual objective $f_{\text{dual}}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \min_{w} \mathcal{L}(w, \boldsymbol{\alpha}, \boldsymbol{\beta})$ often has a closed form
- maximizing f_{dual} over the dual variables $\boldsymbol{\alpha}, \boldsymbol{\beta}$ is often easier than solving the primal problem
- can recover the optimal primal values w from the optimal duals
- reformulation can have other side benefits (as we'll see in SVMs)

How do we use duality to reformulate SVMs?

Recall our SVM optimization problem:

$$\min_{w,b} \quad \tfrac{1}{2}\|w\|^2 \quad \text{subject to} \quad y_i(w^\top x_i + b) \geq 1 \; \forall \; i$$

To find its dual problem, we need to

- write out the Lagrangian: $\mathcal{L}(w, b, \boldsymbol{\alpha}) = \frac{1}{2}\|w\|^2 - \sum_i \alpha_i [y_i(w^T x_i + b) - 1]$
- minimize w.r.t. w, b: $f_{\text{dual}}(\boldsymbol{\alpha}) = \min_{w,b} \mathcal{L}(w, b, \boldsymbol{\alpha})$
- the dual problem is then a maximization over the **dual variables** $\boldsymbol{\alpha} \geq 0$

SVM: Reformulation

To minimize $\mathcal{L}(w, b, \boldsymbol{\alpha}) = \frac{1}{2}\|w\|^2 - \sum_i \alpha_i [y_i(w^T x_i + b) - 1]$ w.r.t w, b, take FOCs:

$$\nabla_w \mathcal{L}(w, b, \boldsymbol{\alpha}) = 0 \quad \rightarrow \quad w = \sum_i \alpha_i y_i x_i$$

$$\partial_b \mathcal{L}(w, b, \boldsymbol{\alpha}) = 0 \quad \rightarrow \quad 0 = \sum_i \alpha_i y_i$$

Plug back into $\mathcal{L}$:

$$f_{\text{dual}}(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^\top x_j - \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^\top x_j - b \sum_i \alpha_i y_i + \sum_i \alpha_i$$

Yielding the **dual SVM problem**

$$\max_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^\top x_j \quad \text{subject to} \quad \sum_i \alpha_i y_i = 0, \ \alpha_i \geq 0$$

SVM: Training with dual version

Simply take the training data (x_i, y_i) and find the dual variables optimizing

$$\max_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^\top x_j \quad \text{subject to} \quad \sum_i \alpha_i y_i = 0, \ \alpha_i \geq 0$$

- this is another convex quadratic program
- training only involves the input data via inner products $x_i^\top x_j$, **not** the vectors x_i themselves

SVM: Testing with dual version

Suppose we've found the dual variables $\boldsymbol{\alpha}^*$ optimizing

$$\max_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^\top x_j \quad \text{subject to} \quad \sum_i \alpha_i y_i = 0,\ \alpha_i \geq 0$$

How do we make predictions on a new input point $x \in X$?

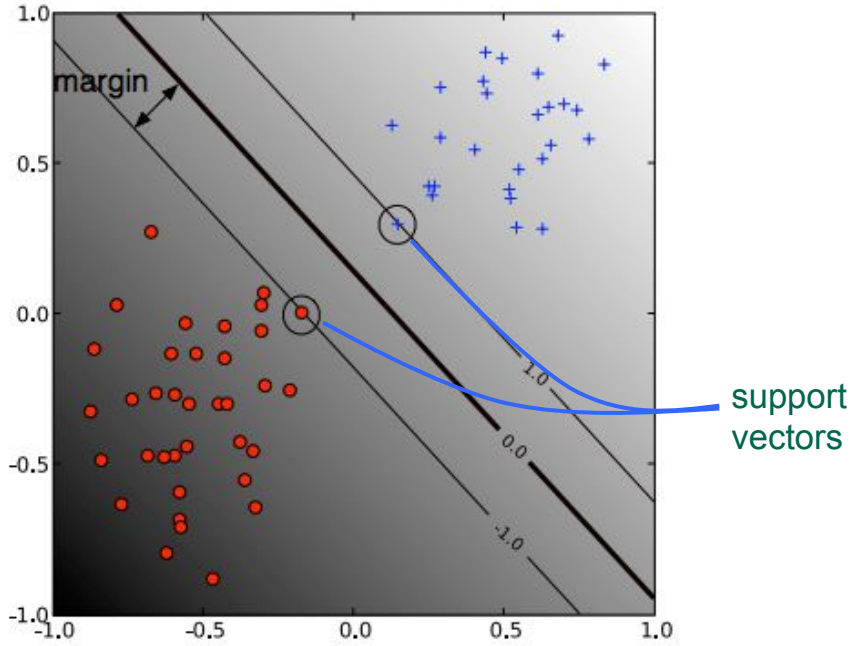
1. compute the optimal primal variables:
   - $w^* = \sum_i \alpha_i^* y_i x_i$ (from the first-order conditions)
   - b^* is more involved but can be computed
2. predict 1 if ${w^*}^\top x + b^* = \sum_i \alpha_i^* y_i x_i^\top x + b^* \geq 0$ and -1 otherwise

Prediction also depends on x, x_i **only through inner products!**

SVM: Support vectors in the dual case

data points x_i with $\alpha_i^* > 0$ lie on the margin boundary and are called ***support vectors***

- the solution w^* is a linear combination of support vectors!
- the solution does not change if we delete points with $\alpha_i = 0$



Break & Quiz

Quiz

Which of the following statements are true?

A. the solution of an SVM will always change if we remove some instances from the training set.

B. if we know that our data is linearly separable, then it does not make sense to use slack variables.

C. if you only had access to the labels $\{y_i\}_i$ and the inner products $\{x_i^\top x_j\}_{i,j}$, we can still find the SVM solution.

A: False, B: False, C: True

Outline

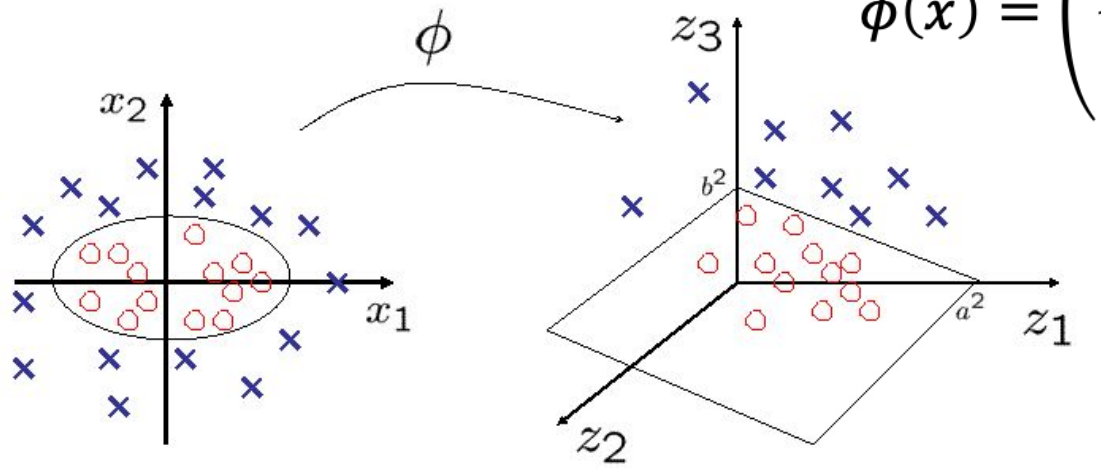
- **Support Vector Machines (SVMs)**
  - margins, training objectives
- **Dual Formulation**
  - Lagrangian, primal and dual problems
- **Kernels**
  - Feature maps, kernel trick

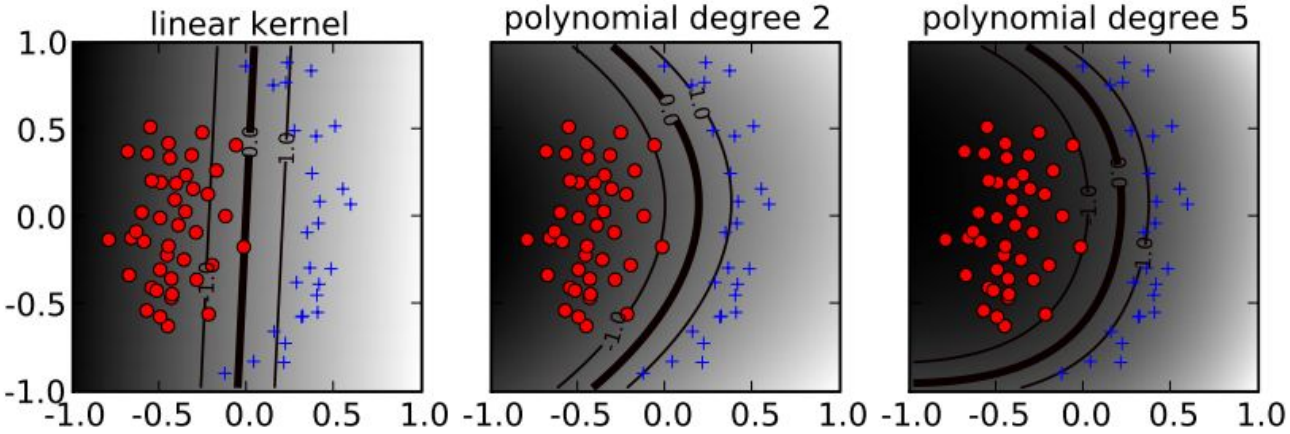
Feature Maps

We can convert a linear classifier to a nonlinear classifier using a **feature maps** ϕ

- transforms points to higher dimensions and use a linear classifier there
- useful if the classes are separated nonlinearly

$$\phi(x) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$





Ben-Hur & Weston, *Methods in Molecular Biology* 2010

Feature Maps and SVMs

Goal: use feature space $\{\phi(x_i)\}$ in a linear classifier

- issue: dimension might be high (possibly infinite)
- specifically, we do not want to write down $\phi(x_i) = [0.2, 0.3, \ldots]$

Recall our SVM dual form:

$$\max_{\boldsymbol{\alpha}} \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^\top x_j \quad \text{subject to} \quad \sum_i \alpha_i y_i = 0, \ \alpha_i \geq 0$$

Training / testing only rely on inner products $x_i^\top x_j$

Thus to run SVM on the feature space $\{\phi(x_i)\}$ we only need $\phi(x_i)^\top \phi(x_j)$

Kernel Trick

If we only need $\phi(x_i)^\top \phi(x_j)$, we don't need to initialize $\phi(\cdot)$ at all! All we need is a function k that quantifies the similarity:

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

Kernel

Feature Maps

Can learn over **any** space you can construct a (valid) kernel over

- "valid" means the $n \times n$ kernel matrix has to be positive definite
- lots of scope for custom similarity measures in specific domains

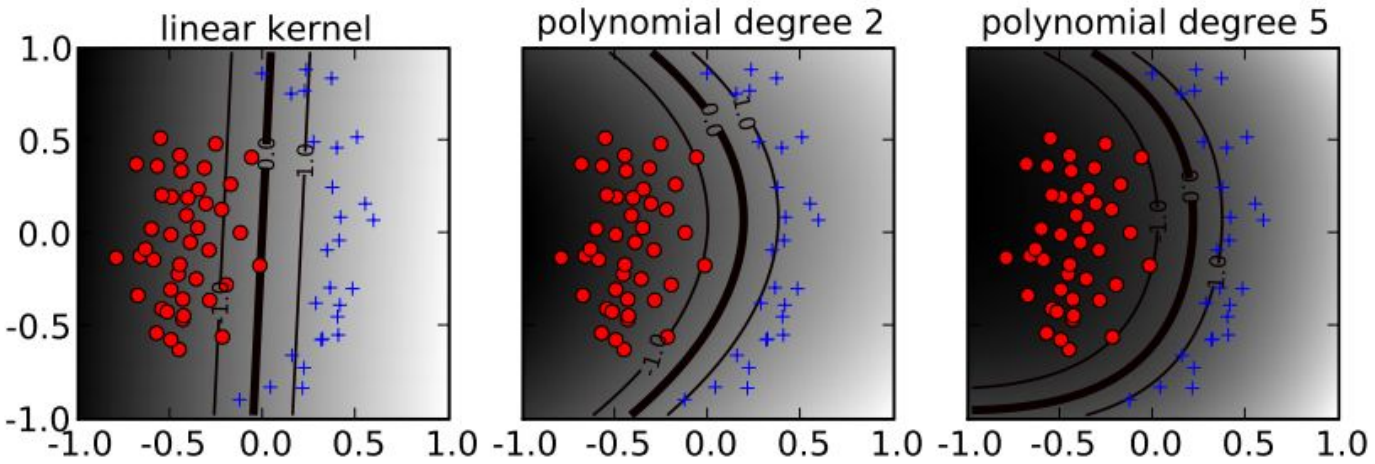
Kernel Types: Polynomial

Fix degree d and constant c:

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

What is $\phi(x)$? Expand the above expression:

$$k(x, x') = (x_1 x_1' + x_2 x_2' + c)^2 = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 x_2 \\ \sqrt{2c}x_1 \\ \sqrt{2c}x_2 \\ c \end{pmatrix} \cdot \begin{pmatrix} x_1'^2 \\ x_2'^2 \\ \sqrt{2}x_1' x_2' \\ \sqrt{2c}x_1' \\ \sqrt{2c}x_2' \\ c \end{pmatrix}$$



Ben-Hur & Weston, *Methods in Molecular Biology* 2010

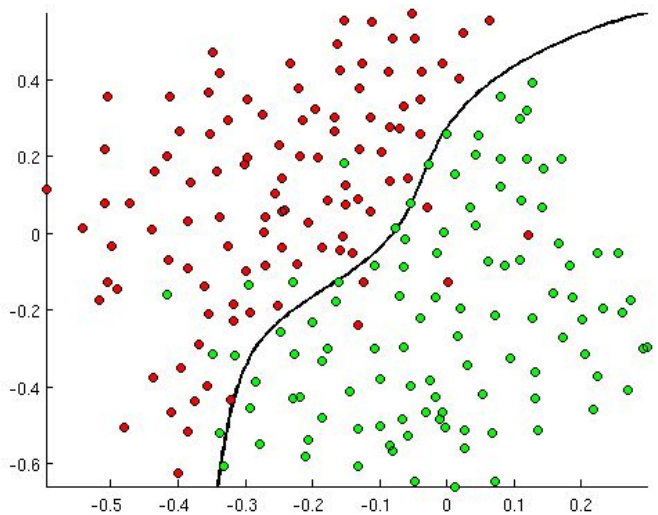
Kernel Types: Gaussian/RBF

- Fix γ:

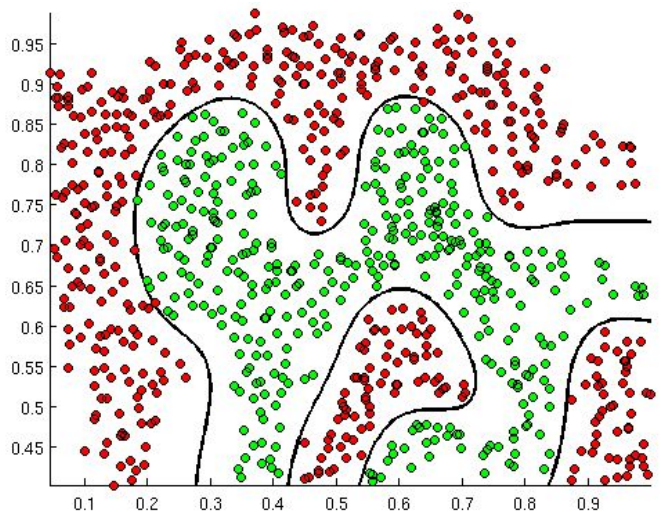
$$k(x, x') = \exp(-\gamma \parallel x - x' \parallel^2)$$

- With RBF kernels, we are projecting to an infinite dimensional space

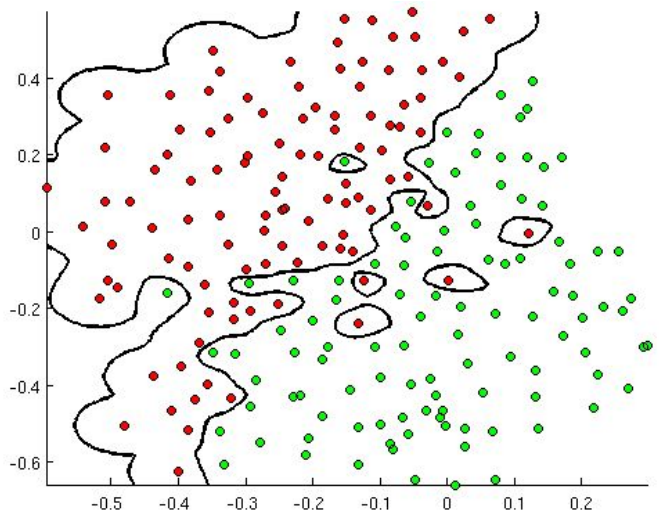
$\gamma = 10$



$\gamma = 100$



$\gamma = 1000$



Break & Quiz

Quiz

Which of the following statements are true?

A. SVMs with nonlinear kernels implicitly transform the low dimensional features to a high dimensional space and then performing linear classification in that space.

B. The “Kernel trick” refers to computing this transformation and then applying the dot product between the transformed points.

A: True, B: False

Quiz

Consider the kernel $k(x, x') = (xx' + 1)^3$ for $x \in \mathbb{R}$. Give an explicit expression for a feature map ϕ such that $\phi(x)^\top \phi(x') = k(x, x')$.

1. $\phi(x)^\top = [x^3, x^2, x, 1]$
2. $\phi(x)^\top = [x^3, \sqrt{3}x^2, \sqrt{3}x, 1]$
3. $\phi(x)^\top = [x^3, \sqrt{3}x^2, x, \sqrt{3}]$
4. $\phi(x)^\top = [x^3, \sqrt{3}x^2, \sqrt{3}x]$

Ans: 2

$$
\begin{aligned}
k(x, x') &= (xx' + 1)^3 \\
&= (xx')^3 + 3(xx')^2 + 3xx' + 1 \\
&= \begin{bmatrix} x^3 & \sqrt{3}x^2 & \sqrt{3}x & 1 \end{bmatrix} \begin{bmatrix} (x')^3 \\ \sqrt{3}(x')^2 \\ \sqrt{3}x' \\ 1 \end{bmatrix}
\end{aligned}
$$

Quiz

Why might we prefer an SVM over a neural network?

A. With an SVM we can map inputs to an infinite dimensional space. With neural networks, we cannot.

B. SVMs are easier to train: An SVM would not get stuck in a local optima, whereas a neural network might.

C. Tuning hyper-parameters in an SVM may be easier than in neural networks.

Ans: all of the above

Extensions of kernel SVM

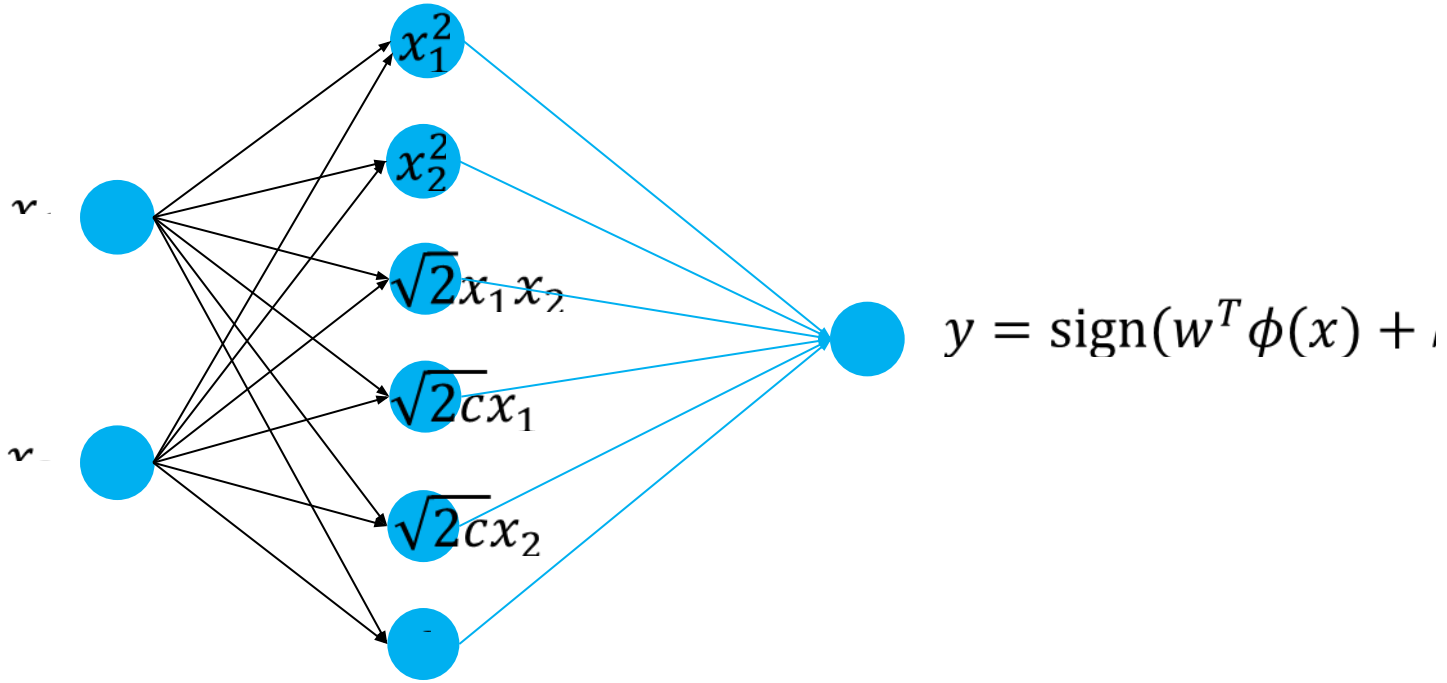
- soft-margin kernel SVM also works with slack variables

- multi-class classification usually done via K one-vs-rest binary classification problems

- regression
  - support vector regression
  - kernel ridge regression (e.g. Gaussian process regression)

Kernel Methods vs. Neural Networks

Can think of kernel SVM approach as fixing a layer of a neural network, but using kernel feature representations instead:



Kernel Methods vs. Neural Networks

Kernel methods were popular in 90's and 2000's

- SVM is one of the biggest successes of learning theory
- still powerful in small / moderate data regimes

Challenges with kernel methods (when we have a lot of data):

- Computational:
  - Computing all pairs of kernel values requires $O(n^2)$ memory
  - Overall compute cost is typically $O(n^3)$
    - solving an LP with n constraints or inverting an $n \times n$ matrix
    - can be accelerated using random Fourier features
- Representation: using a fixed representations is limiting

Thanks Everyone!

Some of the slides in these lectures have been adapted/borrowed from materials developed by Misha Khodak, Mark Craven, David Page, Jude Shavlik, Tom Mitchell, Nina Balcan, Elad Hazan, Tom Dietterich, Pedro Domingos, Jerry Zhu, Yingyu Liang, Volodymyr Kuleshov, Fei-Fei Li, Justin Johnson, Serena Yeung, Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Josiah Hanna