CS 760: Machine Learning
SVMs and Kernels

Fred Sala

University of Wisconsin-Madison

Nov. 2, 2021

Announcements

- **Logistics**:
  - HW 5 coming out shortly
  - Probability tutorial?

- Class roadmap:

Tuesday, Nov. 2	Kernels + SVMs
Thursday, Nov. 4	Graphical Models I
Tuesday, Nov. 9	Graphical Models II
Thursday, Nov. 11	Less-than-full Supervision

Outline

- **Review & Generative Adversarial Networks**
  - Applications, histograms, autoregressive models
- **Support Vector Machines (SVMs)**
  - Lagrangian duality, margins, training objectives
- **Kernels**
  - Feature maps, kernel trick, conditions

Outline

- **Review & Generative Adversarial Networks**
  - Applications, histograms, autoregressive models
- **Support Vector Machines (SVMs)**
  - Lagrangian duality, margins, training objectives
- **Kernels**
  - Feature maps, kernel trick, conditions

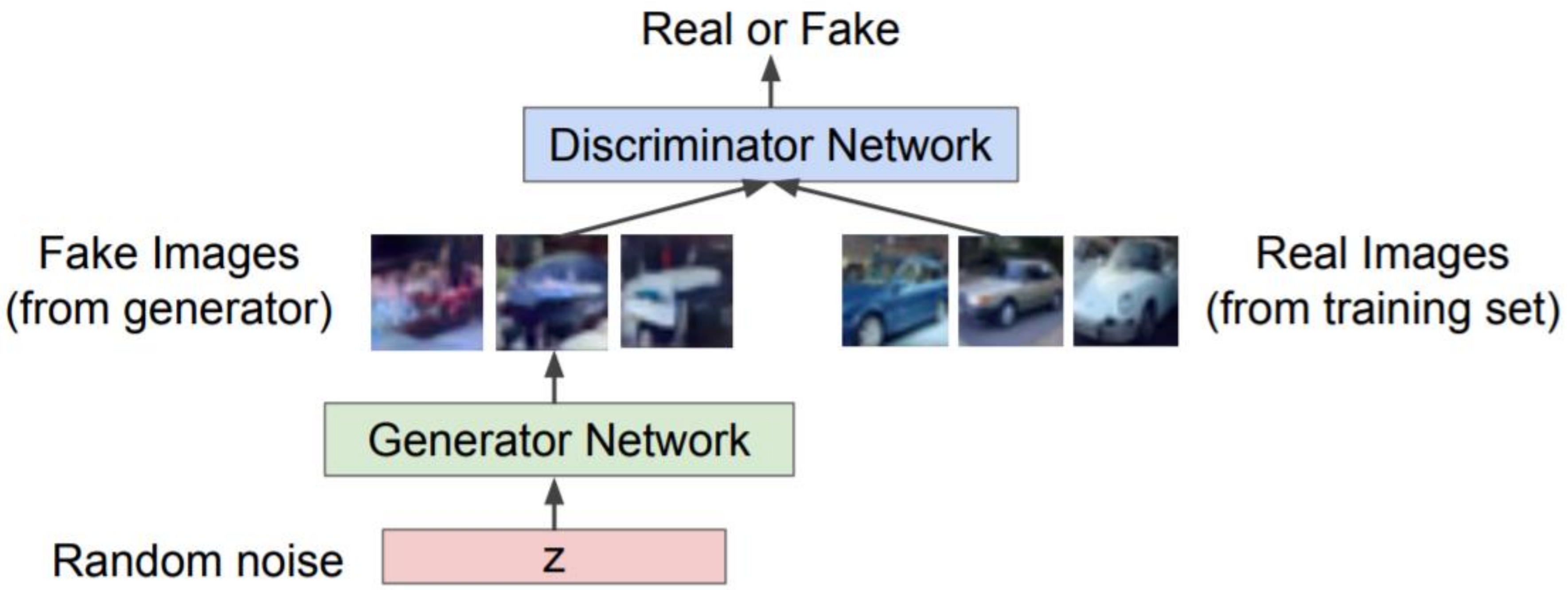
GANs: Generative Adversarial Networks

- So far, we've been modeling the density…
  - What if we just want to get high-quality samples?

- GANs do this. Based on a clever idea:
  - Art forgery: very common through history
  - Left: original
  - Right: forged version
  - Two-player game. **Forger** wants to pass off the forgery as an original; **investigator** wants to distinguish forgery from original

GANs: Basic Setup

- Let's set up networks that implement this idea:
  - Discriminator network: like the **investigator**
  - Generator network: like the **forger**



Stanford CS231n / Emily Denton

GAN Training: Discriminator

- How to train these networks? Two sets of parameters to learn: θ_d (discriminator) and θ_g (generator)

- Let's fix the generator. What should the discriminator do?
  - Distinguish fake and real data: binary classification.
  - Use the cross entropy loss, we get

$$\max_{\theta_d} \mathbb{E}_{x\sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

Real data, want to classify 1

Fake data, want to classify 0

GAN Training: Generator & Discriminator

- How to train these networks? Two sets of parameters to learn: θ_d (discriminator) and θ_g (generator)

- This makes the discriminator better, but also want to make the generator more capable of fooling it:
  - Minimax game! Train jointly.

$$\min_{\theta_g} \max_{\theta_d} \mathbb{E}_{x\sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

Real data, want to classify 1

Fake data, want to classify 0

GAN Training: Alternating Training

- So we have an optimization goal:

$$\min_{\theta_g} \max_{\theta_d} \mathbb{E}_{x\sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

- Alternate training:
  - **Gradient ascent**: fix generator, make the discriminator better:

$$\max_{\theta_d} \mathbb{E}_{x\sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

  - **Gradient descent**: fix discriminator, make the generator better

$$\min_{\theta_g} \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

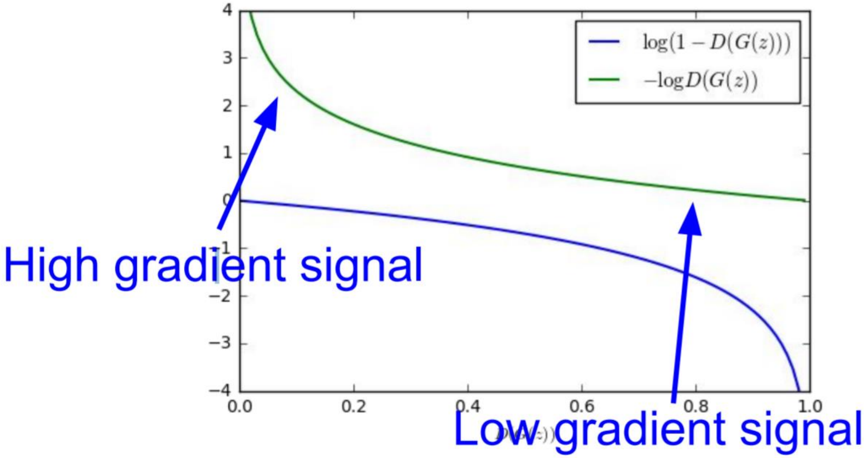
GAN Training: Issues

- Training often not stable
- Many tricks to help with this:
  - Replace the generator training with

$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

  - Better gradient shape
  - Choose number of alt. steps carefully

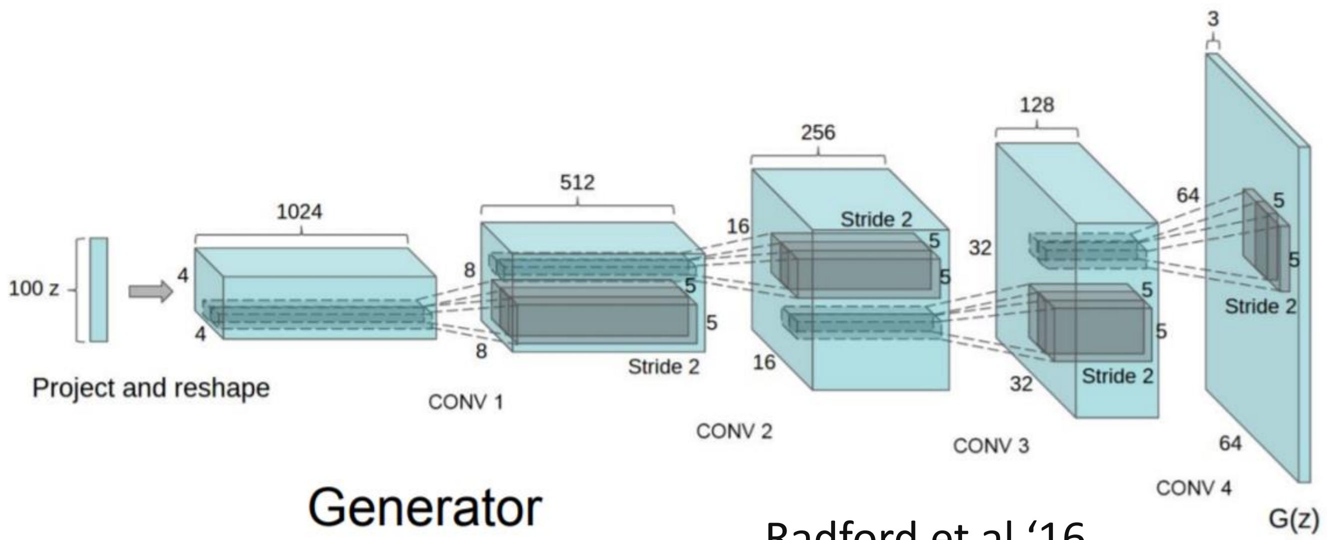
- Can still be challenging.



Stanford CS231n

GAN Architectures

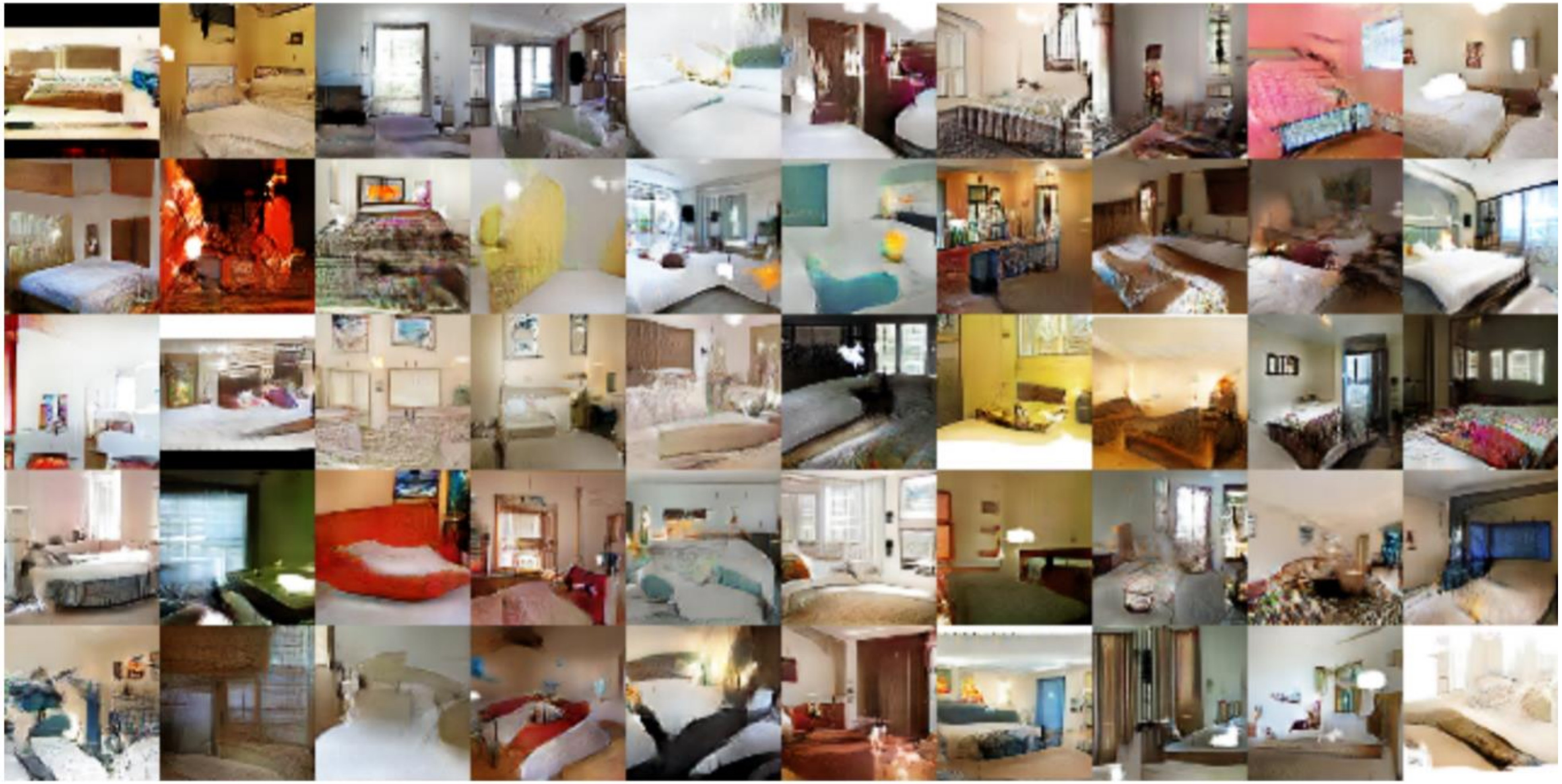
- So far we haven't commented on what the networks are
- **Discriminator**: image classification, use a **CNN**
- What should **generator** look like
  - Input: noise vector z. Output: an image (ie, volume 3 x width x height)
  - Can just reverse our CNN pattern…



Radford et al '16

GANs: Example

- From Radford’s paper, with 5 epochs of training:



Break & Quiz

Outline

- **Review & Generative Adversarial Networks**
  - Applications, histograms, autoregressive models
- **Support Vector Machines (SVMs)**
  - Lagrangian duality, margins, training objectives
- **Kernels**
  - Feature maps, kernel trick, conditions

Mini-Tutorial: Constrained Optimization

- Take optimization problem:

$$\min_{w} \; f(w)$$

$$g_i(w) \leq 0, \forall 1 \leq i \leq k$$

$$h_j(w) = 0, \forall 1 \leq j \leq l$$

Objective

Constraints

- 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 α_i, β_j's are called **Lagrange multipliers**

Mini-Tutorial: Lagrangian

- Form the quantity:

$$\theta_P(w) := \max_{\boldsymbol{\alpha},\boldsymbol{\beta}:\alpha_i \geq 0} \mathcal{L}(w, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

$$:= \max_{\boldsymbol{\alpha},\boldsymbol{\beta}:\alpha_i \geq 0} f(w) + \sum_i \alpha_i g_i(w) + \sum_j \beta_j h_j(w)$$

$$g_i(w) \leq 0, \forall 1 \leq i \leq k$$

$$h_j(w) = 0, \forall 1 \leq j \leq l$$

- Note:

$$\theta_P(w) = \begin{cases} f(w), & \text{if } w \text{ satisfies all the constraints} \\ +\infty, & \text{if } w \text{ does not satisfy the constraints} \end{cases}$$

Mini-Tutorial: Lagrangian

- Form the quantity:

$$\theta_P(w) := \max_{\boldsymbol{\alpha},\boldsymbol{\beta}:\alpha_i \geq 0} \mathcal{L}(w, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

- Note:

$$\theta_P(w) = \begin{cases} f(w), & \text{if } w \text{ satisfies all the constraints} \\ +\infty, & \text{if } w \text{ does not satisfy the constraints} \end{cases}$$

- Minimizing $f(w)$ with constraints is the same as minimizing $\theta_P(w)$

$$\min_w f(w) = \min_w \theta_P(w) = \min_w \max_{\boldsymbol{\alpha},\boldsymbol{\beta}:\alpha_i \geq 0} \mathcal{L}(w, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

Mini-Tutorial: Duality

- The primal problem

$$p^* := \min_w f(w) = \min_w \max_{\boldsymbol{\alpha},\boldsymbol{\beta}:\alpha_i \geq 0} \mathcal{L}(w, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

- The dual problem

$$d^* := \max_{\boldsymbol{\alpha},\boldsymbol{\beta}:\alpha_i \geq 0} \min_w \mathcal{L}(w, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

- Always true:

$$d^* \leq p^*$$

Mini-Tutorial: Duality

- Always true:

$$d^* \le p^*$$

Let's see why:

Definition

$$
\begin{aligned}
d^* &:= \max_{\boldsymbol{\alpha},\boldsymbol{\beta}:\alpha_i \ge 0} \min_{w} \mathcal{L}(w, \boldsymbol{\alpha}, \boldsymbol{\beta}) \\
&= \max_{\boldsymbol{\alpha},\boldsymbol{\beta}:\alpha_i \ge 0} \min_{w} f(w) + \textstyle\sum_i \alpha_i g_i(w) + \sum_j \beta_j h_j(w) \\
&\le \max_{\boldsymbol{\alpha},\boldsymbol{\beta}:\alpha_i \ge 0} f(w^*) + \underbrace{\textstyle\sum_i \alpha_i g_i(w^*) + \sum_j \beta_j h_j(w^*)}_{\textbf{Non-positive}} \\
&= p^*
\end{aligned}
$$

Mini-Tutorial: Duality Gap

- Always true:

$$d^* \leq p^*$$

If **actual equality**, could solve dual instead of primal… when?

- Under conditions (ex: Slater's), there exists $(w^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ such that

$$d^* = \mathcal{L}(w^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*) = p^*$$

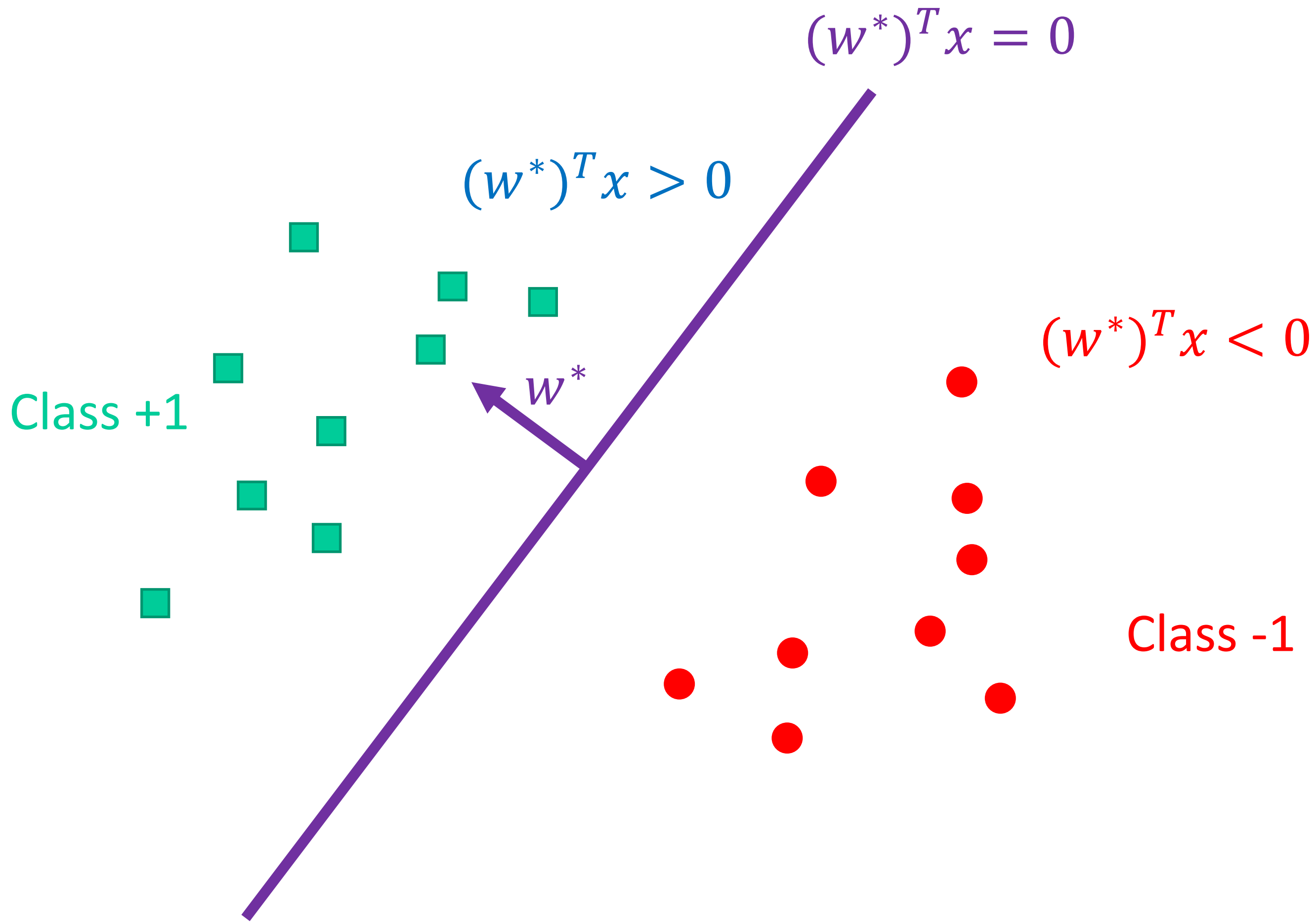
- $(w^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ satisfy Karush-Kuhn-Tucker (KKT) conditions:

$$\frac{\partial \mathcal{L}}{\partial w_i} = 0, \qquad \alpha_i g_i(w) = 0$$

$$g_i(w) \leq 0, \ \ h_j(w) = 0, \qquad \alpha_i \geq 0$$

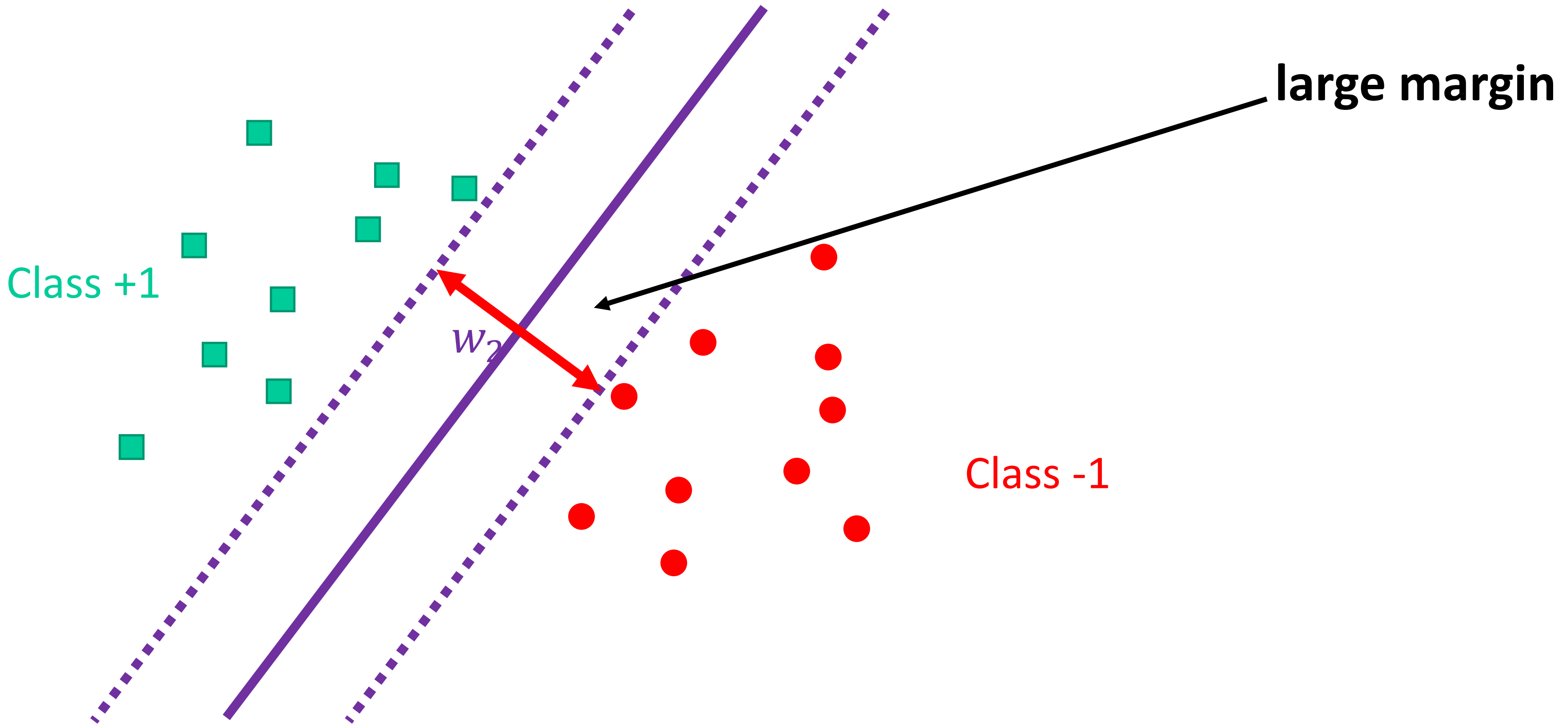
Review: Linear Classification

- Assuming linear separability,



Review: Training & Margins

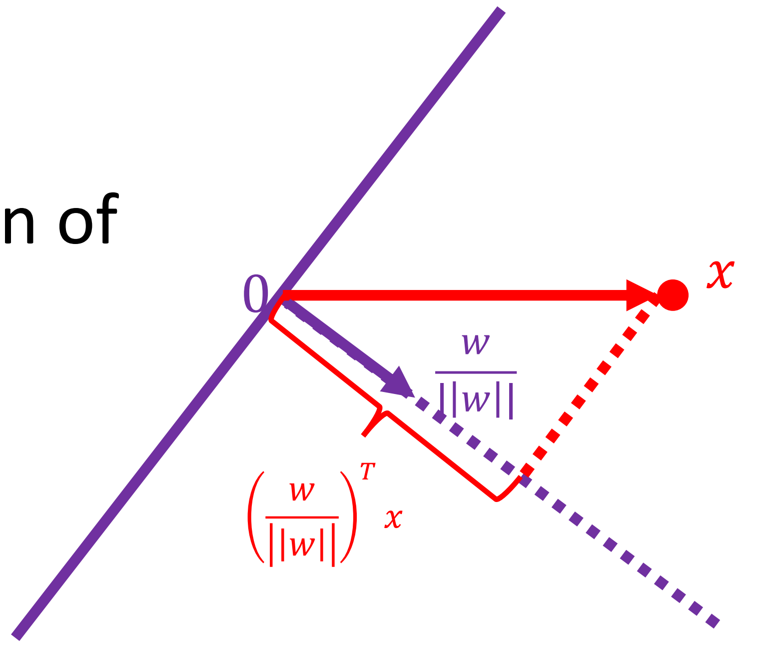
- Want: a **large margin**



Mini-Tutorial: Linear Algebra & Margin

- What's the expression for the margin?
  - We write $y = \text{sign}(f_w(x)) = \text{sign}(w^T x)$

- x has distance $\frac{|f_w(x)|}{||w||}$ to the hyperplane $w^T z = 0$
  - Let's show it. w is orthogonal to the hyperplane
  - The unit direction is $\frac{w}{||w||}$
  - For any unit vector v, the length of the projection of x on v is $|v^T x|$
  - The projection of x is $\left(\frac{w}{||w||}\right)^T x = \frac{f_w(x)}{||w||}$



Mini-Tutorial: Linear Algebra & Margin

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

w is orthogonal to $w^T z + b = 0$

Proof:

- Let $x = x_\perp + r \frac{w}{||w||}$, then $|r|$ is the distance

- Multiply both sides by w^T and add b
- Left hand side: $w^T x + b = f_{w,b}(x)$
- Right hand side: $w^T x_\perp + r \frac{w^T w}{||w||} + b = 0 + r||w||$

Support Vector Machines: Candidate Goal

- The absolute margin over all training data points:

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

Using our result

- We want correct $f_{w,b}$, (recall $y_i \in \{+1, -1\}$). Define the margin to be

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

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

Support Vector Machines: Candidate Goal

- One way: maximize margin over all training data points:

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

- A bit complicated …
  - How do we use our optimization approaches?

SVM: Simplified Goal

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

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

- Let's 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’s consider a fixed scale such that

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

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

- Now we have for all data

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

and at least for one i the equality holds

- Then the margin over all training points is $\frac{1}{||w||}$

SVM: Loss Function

- Optimization simplified to

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

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

- How to find the optimum $\hat{w}^*$?
- Let's use our **Lagrange multiplier method**

$$\mathcal{L}(w, b, \boldsymbol{\alpha}) = \frac{1}{2}||w||^2 - \sum_i \alpha_i [y_i(w^T x_i + b) - 1]$$

SVM: Optimization

- To meet the KKT conditions:

$$\frac{\partial \mathcal{L}}{\partial w} = 0, \rightarrow w = \sum_i \alpha_i y_i x_i \quad (1)$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0, \rightarrow 0 = \sum_i \alpha_i y_i \quad (2)$$

$$\mathcal{L}(w, b, \boldsymbol{\alpha}) = \frac{1}{2} ||w||^2 - \sum_i \alpha_i [y_i (w^T x_i + b) - 1]$$

- **Two rules.** Plug into $\mathcal{L}$:

$$\mathcal{L}(w, b, \boldsymbol{\alpha}) = \sum_i \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j x_i^T x_j \quad (3)$$

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

SVM: Dual Version

- Reduces to dual problem:

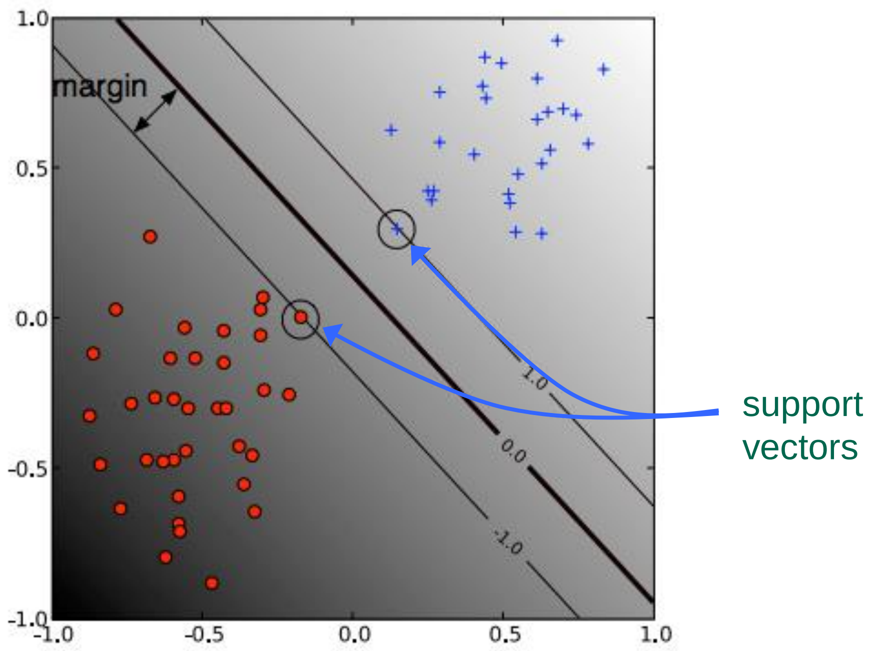
$$\max_{\boldsymbol{\alpha}} \mathcal{L}(w, b, \boldsymbol{\alpha}) = \max_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

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

- Since $w = \sum_i \alpha_i y_i x_i$, we have $w^T x + b = \sum_i \alpha_i y_i x_i^T x + b$
- Note: only deals with data via **inner products** $x_i^T x_j$

SVM: Support Vectors

- Solution is a sparse linear combination of training instances
- Those instances with $\alpha_i > 0$ are called ***support vectors***
  - Lie on the margin boundary
- Solution does not change if we delete instances with $\alpha_i = 0$



SVM: Soft Margin

What if our data isn't linearly separable?

- Can adjust our approach by using *slack variables* (denoted by ζ_i) to tolerate errors

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

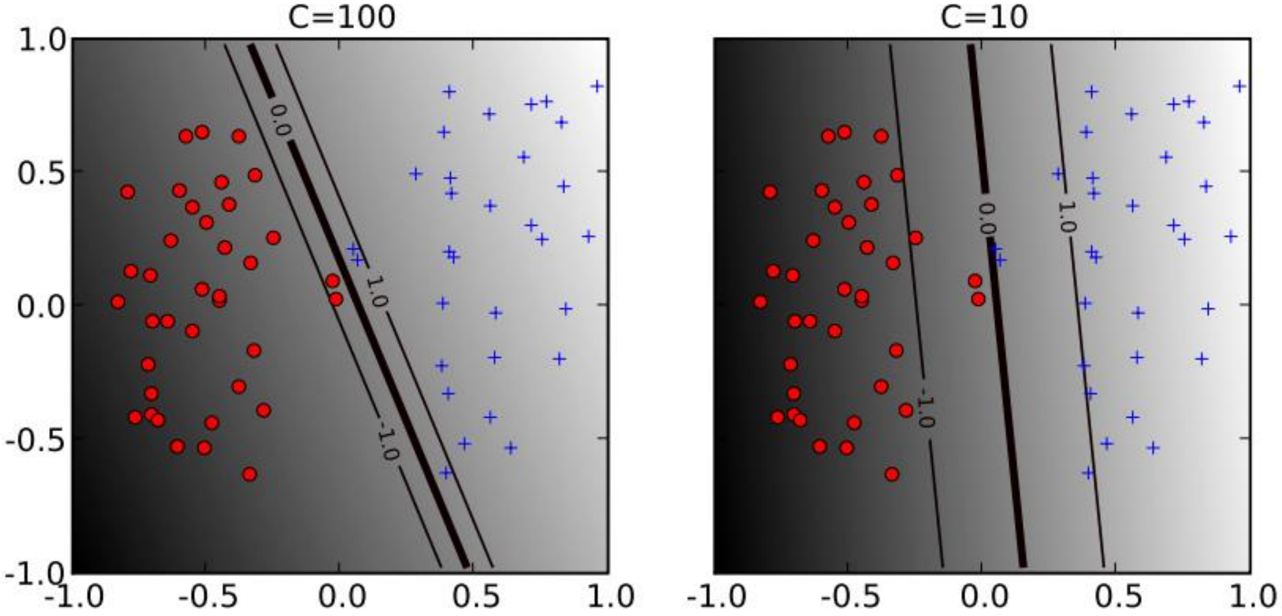
$$y_i(w^T x_i + b) \geq 1 - \zeta_i, \zeta_i \geq 0, \forall i$$

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

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

Break & Quiz

Outline

- **Review & Generative Adversarial Networks**
  - Applications, histograms, autoregressive models
- **Support Vector Machines (SVMs)**
  - Lagrangian duality, margins, training objectives
- **Kernels**
  - Feature maps, kernel trick, conditions

Feature Maps

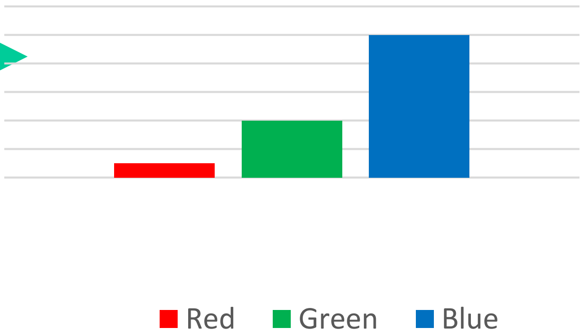
- Can take a set of features and map them into another
  - Can also construct non-linear features
  - Use these inside a linear classifier?

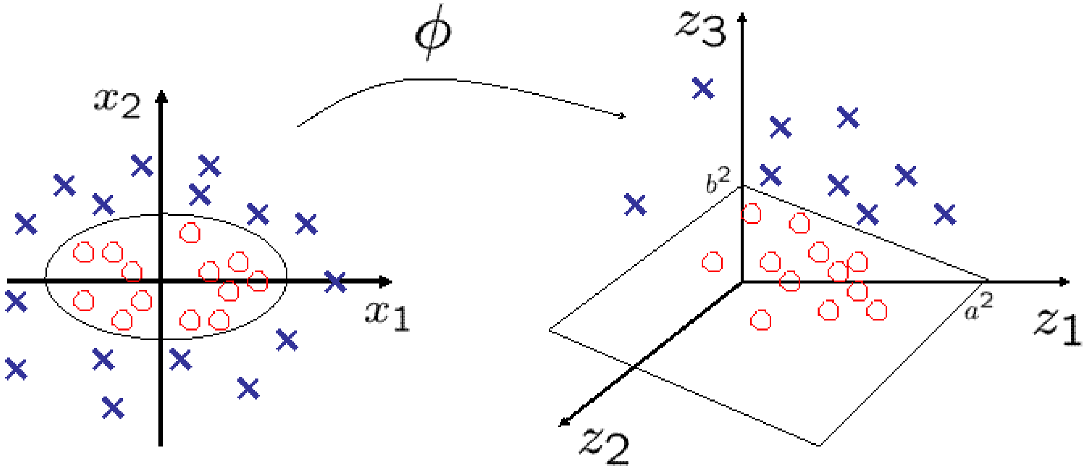
x

Extract features

$\phi(x)$

Color Histogram





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

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

Feature Maps and SVMs

Want to use feature space $\{\phi(x_i)\}$ in linear classifier…

- Downside: dimension might be high (even infinite!)
- So we don't want to write down $\phi(x_i) = [0.2, 0.3, \ldots]$

Recall our SVM dual form:

- Only relies on inner products $x_i^T x_j$

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

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

Kernel Trick

- 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

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

Kernel Matrix   Feature Maps

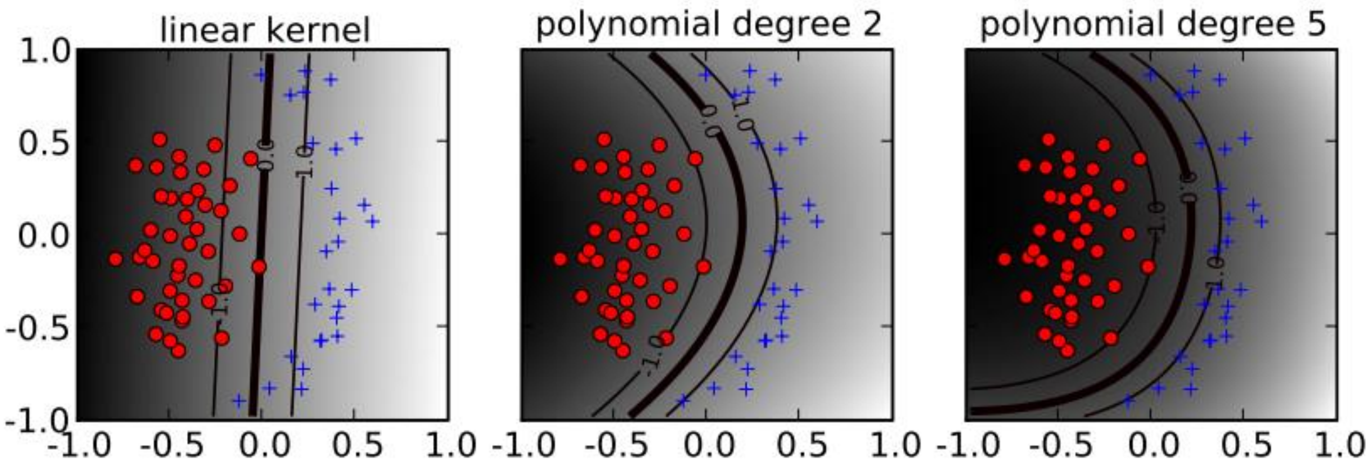
Kernel Types: Polynomial

- Fix degree d and constant c:

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

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

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



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

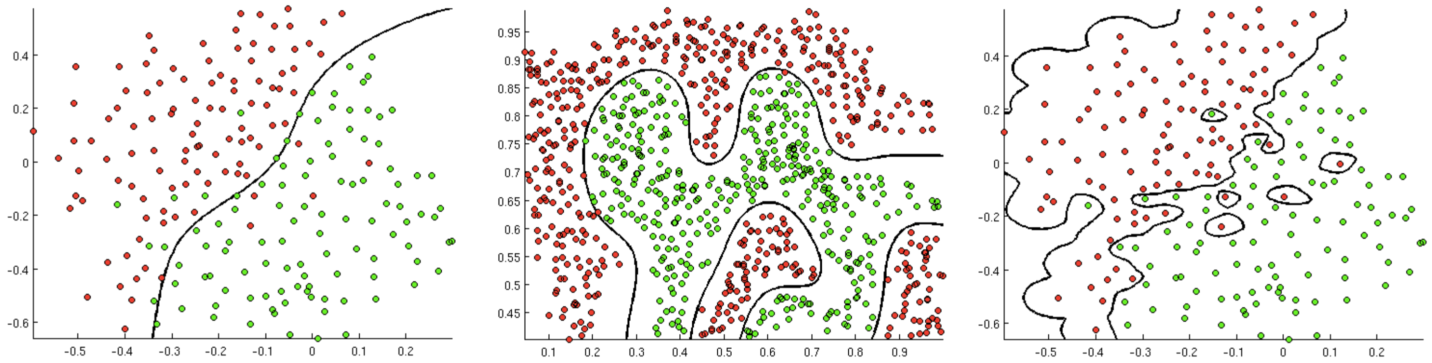
Kernel Types: Gaussian/RBF

- Fix bandwidth σ:

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

  - Also called radial basis function (RBF) kernels

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



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

Andrew Ng

Theory of Kernels

- Part of a deep mathematical theory
- With some conditions, any kernel yields a feature map:
  - Theorem: $k(x, x')$ has expansion

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

Feature Maps

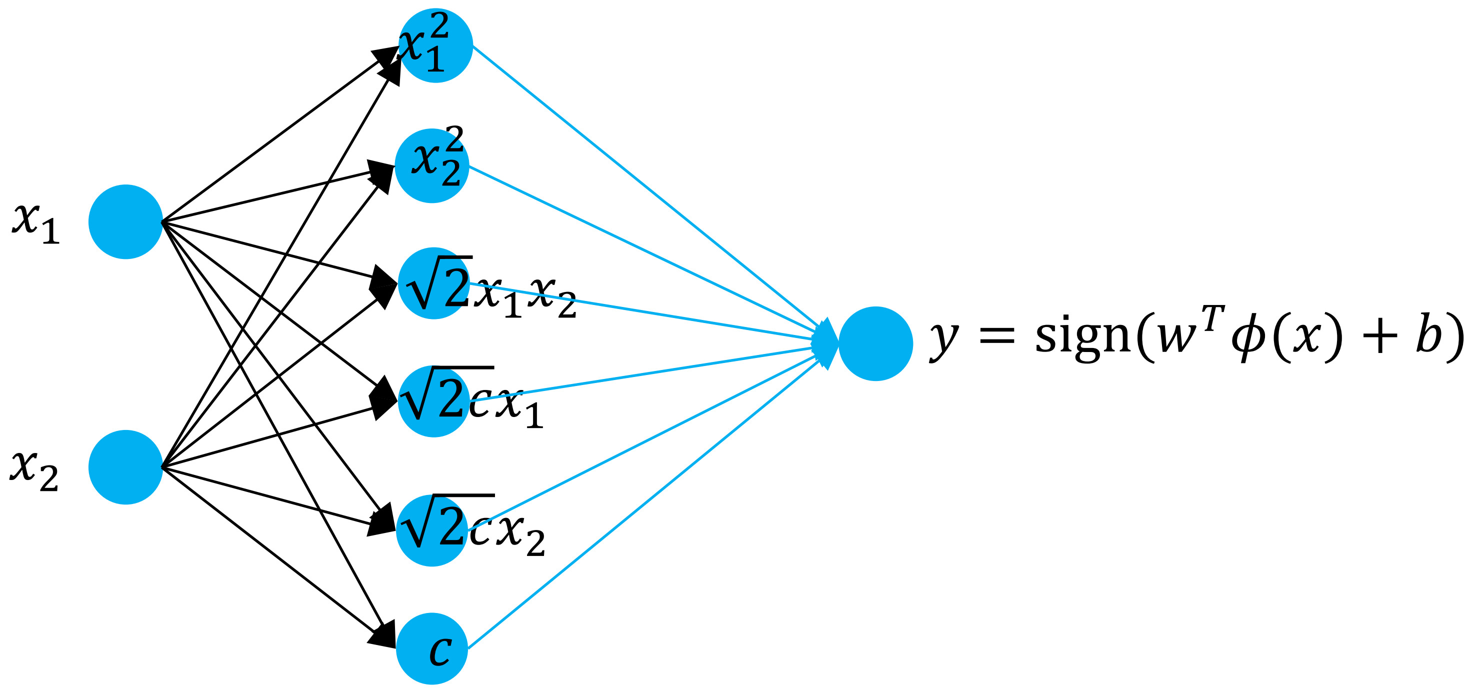
for nonnegative a_i's, if and only if for any function $c(x)$,

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

- Given certain requirements/conditions, can construct a bunch of new kernels from existing ones

Kernel Methods VS Neural Networks

- Can think of our kernel SVM approach as fixing a layer of a neural network



SVM Review

- Can find globally optimal solutions: convex optimization
  - No local minima (unlike training general NNs)
- Can train primal or dual
  - Dual: relies on **support vectors**; enables use of **kernels**
- Variety of pre-existing optimization techniques
- Kernels: allow non-linear decision boundaries
  - And to represent all sorts of new data (strings, trees)
  - High-dimensional representations, but can use kernel trick to avoid explicitly computing feature maps
  - Good performance! Sometimes close to DNNs

Thanks Everyone!

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, 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