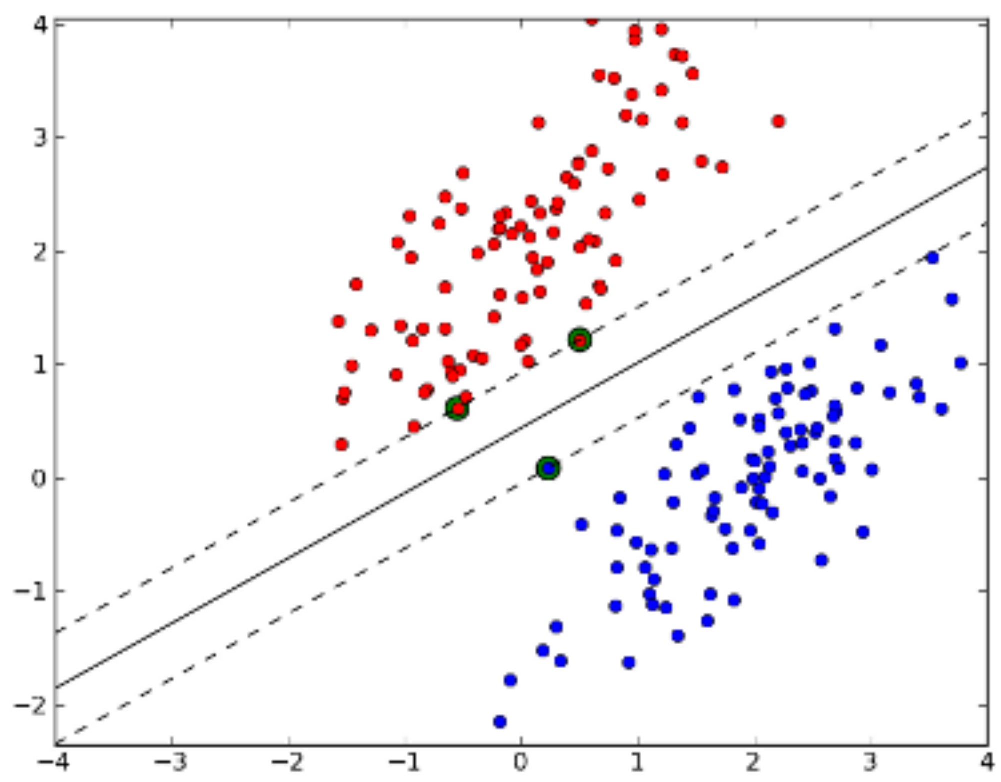
Kernel Method

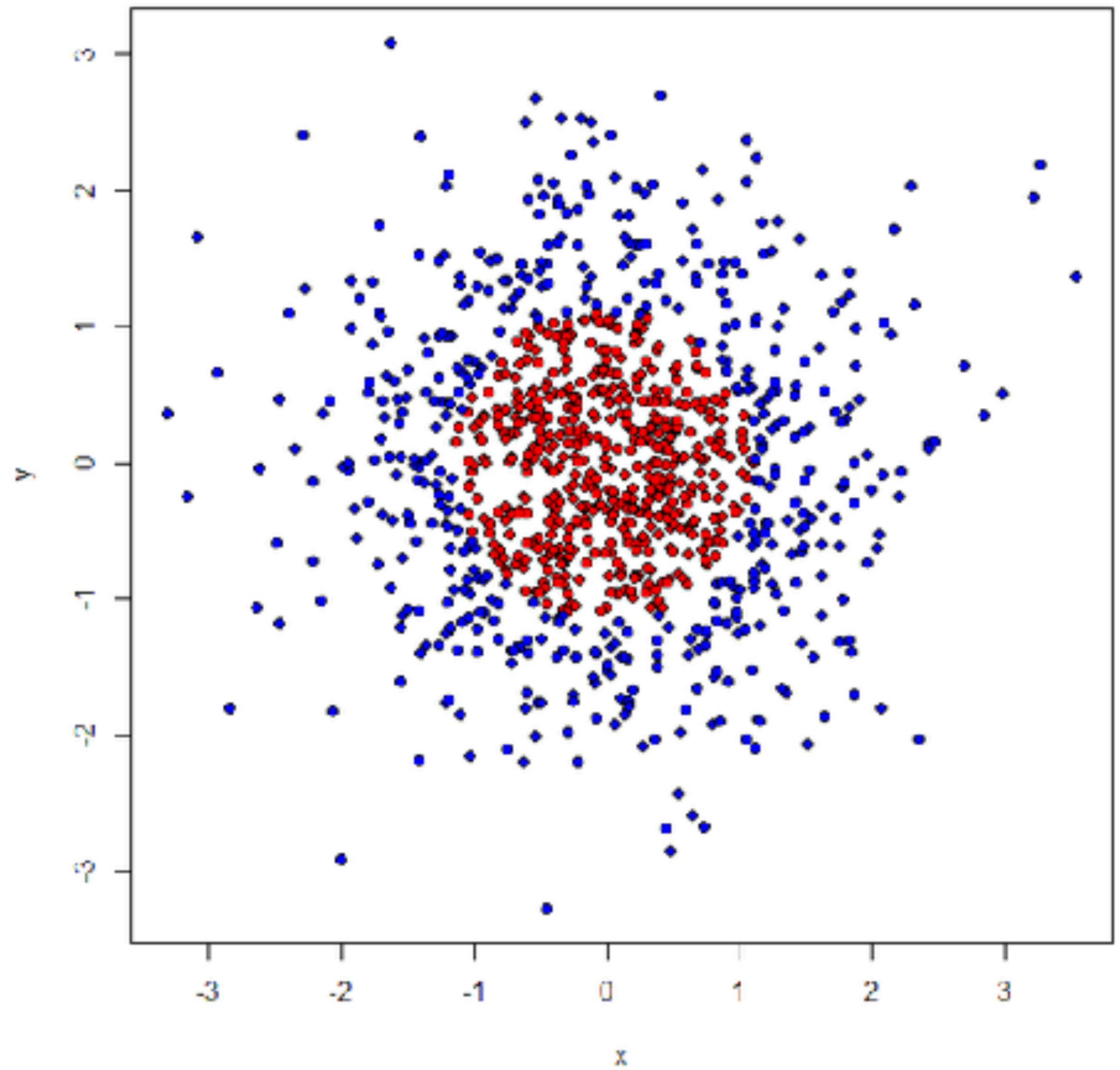
Lingqiao Liu

Motivation

- Limitation of Linear Classifier



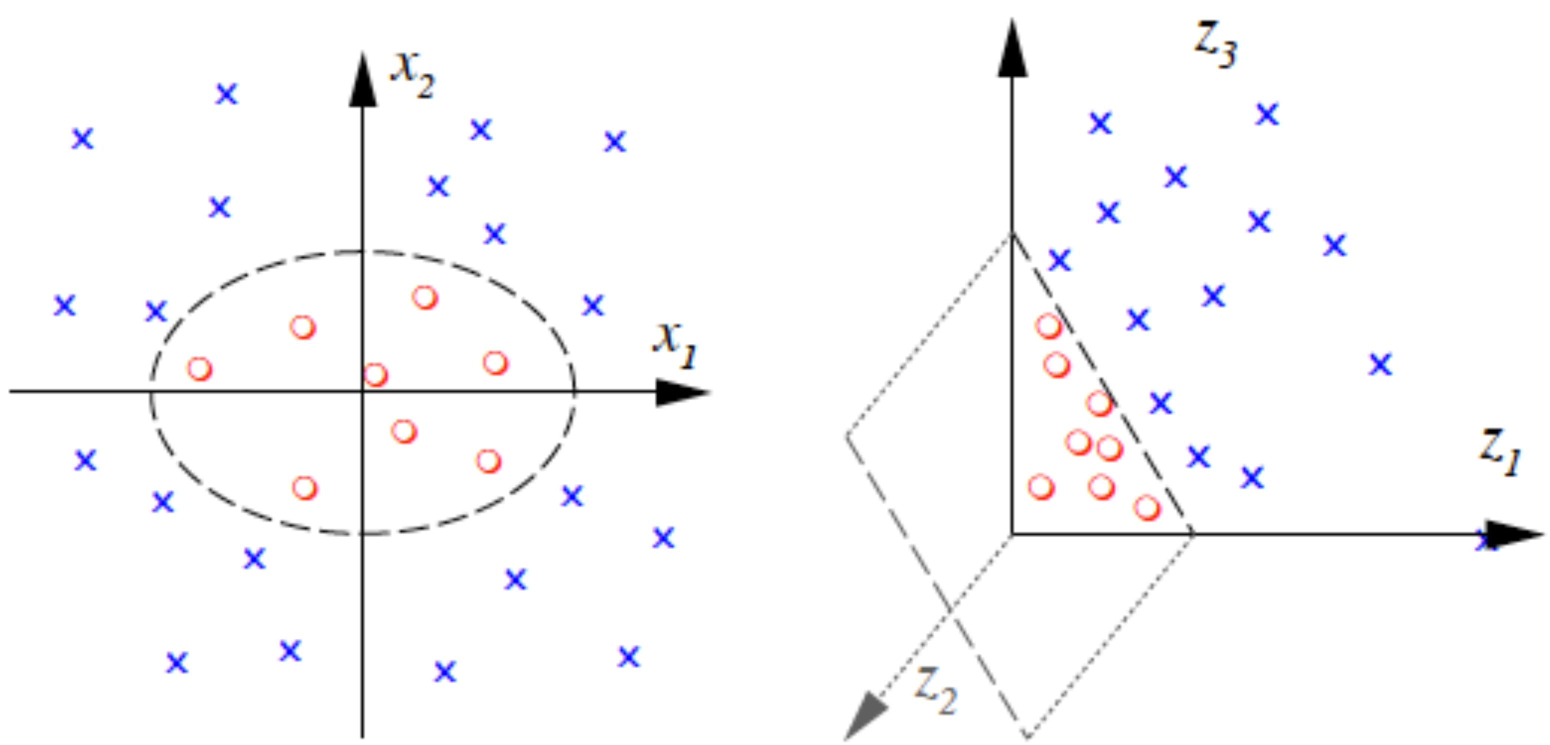
Limitation of linear model



Feature transform

- Idea: transform data to another feature space

$$\Phi : R^2 \to R^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt(2)x_1 x_2, x_2^2)$$



Demo

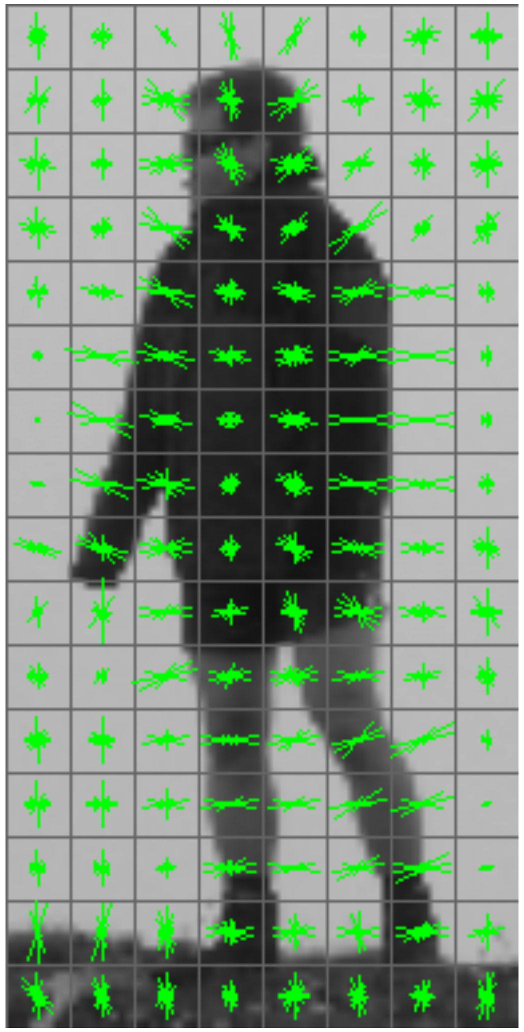
- https://www.youtube.com/watch?v=3liCbRZPrZA

Feature transform

- Importance of the feature transform
  - High-dimensional feature space usually works
- The transform function
  - Can be a simple arithmetic operation
  - Can be anything!

Feature transform

e.g. Instead of using raw pixel, using histograms of oriented gradient



THE UNIVERSITY of ADELAIDE

Issue

- The dimensionality of the mapped feature
  - can be high or even infinite
  - computational cost or infeasible
- More convenient to define it implicitly

Kernel

- In many cases, we are only interested in the inner product of the mapped features

$$K(\mathbf{x}, \mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle$$

Kernel tricks

- Kernel function
  - can be more concise than the feature map

Polynomial kernel

$$K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + c)^d$$

Can be expand as (when d =2)

$$\begin{aligned} K(\mathbf{x}, \mathbf{x}') &= (\mathbf{x}^T \mathbf{x}' + c)^2 \\ &= \sum_{i=1}^{n} (x_i^2)({x'}_i^2) + \sum_{i=2}^{n} \sum_{j=1}^{i-1} (\sqrt{2} x_i x_j)(\sqrt{2} {x'}_i, {x'}_j) \\ &\quad + \sum_{i=1}^{n} (\sqrt{2c} x_i)(\sqrt{2c} {x'}_i) + c^2 \end{aligned}$$

Polynomial kernel

Equivalent to the following feature transform

$$\varphi(\mathbf{x}) = \left\langle x_n^2, \cdots, x_1^2, \sqrt{2}x_n x_{n-1}, \sqrt{2}x_{n-1}x_{n-2}, \cdots, \sqrt{2}x_{n-1}x_1, \cdots, \sqrt{2}x_2 x_1, \sqrt{2c}x_n, \cdots, \sqrt{2c}x_1, c \right\rangle$$

$$K(\mathbf{x}, \mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle$$

Calculating inner product

1. using kernel function
2. using feature transform

Which one is more efficient?

THE UNIVERSITY of ADELAIDE

Gaussian Kernel

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{\sigma^2}\right)$$

Can be expand as

$$= \sum_{j=0}^{\infty} \sum_{\sum n_i = j} \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x}\|^2\right) \frac{x_1^{n_1} \cdots x_k^{n_k}}{\sqrt{n_1! \cdots n_k!}} \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x}'\|^2\right) \frac{x'^{n_1}_1 \cdots x'^{n_k}_k}{\sqrt{n_1! \cdots n_k!}}$$

Understand Kernels

- Intuitively, modelling the similarity between two feature points
  - Guideline for designing kernels
  - Not all similarity measurement can be a kernel function

Criterion for a valid kernel

- Check if $K(\mathbf{x}, \mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle$
- Given any m samples

$$K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_m) \\ \vdots & \ddots & \vdots \\ k(x_m, x_1) & \cdots & k(x_m, x_m) \end{bmatrix}$$

can be decomposed into $\mathbf{K} = \phi(\mathbf{X})^T \phi(\mathbf{X})$

Criterion for a valid kernel

- Implies Semi-positive-definite of K
- Definition

$$\mathbf{a}^T \mathbf{K} \mathbf{a} \geq 0$$
$$= \sum_i \sum_j K(\mathbf{x}_i, \mathbf{x}_j) a_i a_j \geq 0$$

- We can extend it to Hilbert space

Criterion for a valid kernel

- Mercer Condition

A real-valued function $K(x, y)$ is said to fulfill Mercer's condition if for all square integrable function $g(x)$ one has

$$\int\int g(x)K(x,y)g(y)dxdy \geq 0$$

$$\int_{-\infty}^{\infty} |g(x)|^2 dx < \infty$$

Kernel SVM

- Linear SVM (Primal form)

$$
\begin{aligned}
&\min_{\mathbf{w},b,\{\xi_i\}} \|\mathbf{w}\|_2^2 + \lambda \sum_i \xi_i \\
s.t. \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \\
& \xi_i \geq 0 \quad \forall i
\end{aligned}
$$

Kernel SVM

- Dual form

$$\max_{\{\alpha_i\}} \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j y_i \alpha_i y_j \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

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

$$0 \le \alpha_i \le \lambda \quad \forall i$$

Kernel SVM

SVM dual

$$\max_{\{\alpha_i\}} \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j y_i \alpha_i y_j \alpha_j \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$$

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

$$0 \leq \alpha_i \leq \lambda \quad \forall i$$

Kernel SVM

SVM dual

Inner product

$$\max_{\{\alpha_i\}} \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j y_i \alpha_i y_j \alpha_j \boxed{\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle}$$

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

$$0 \le \alpha_i \le \lambda \quad \forall i$$

Kernel SVM

SVM dual

$$\max_{\{\alpha_i\}} \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j y_i \alpha_i y_j \alpha_j k(\mathbf{x}_i, \mathbf{x}_j)$$

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

$$0 \leq \alpha_i \leq \lambda \quad \forall i$$

Kernel SVM

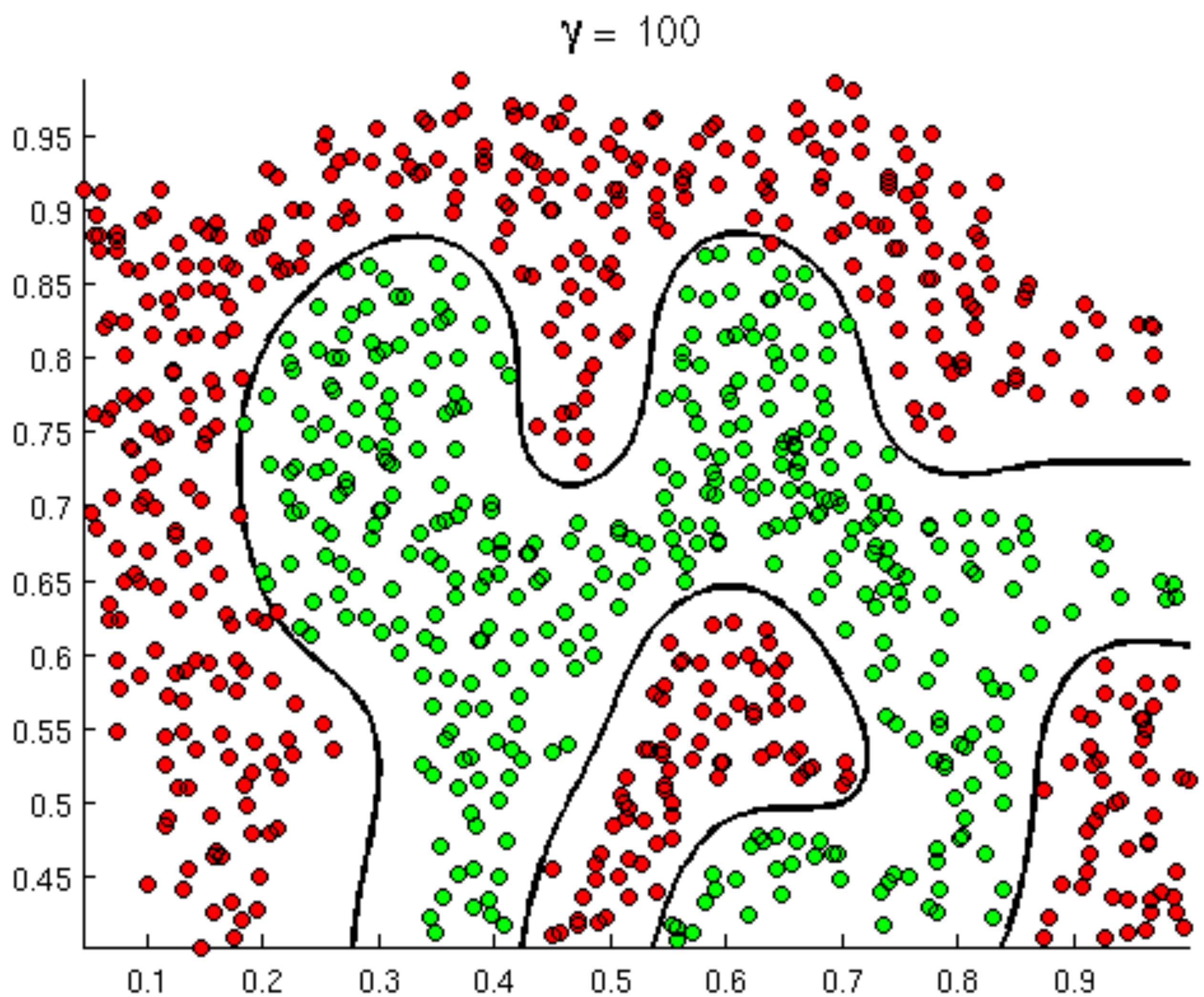
Decision Function for Kernel SVM

$$f(\mathbf{x}_t) = \sum_{i}^{N} \alpha_i y_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_t) \rangle$$

$$f(\mathbf{x}_t) = \sum_{i}^{N} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}_t)$$

Kernel SVM

- Decision boundary of kernel SVM



Lesson Learned 1

- Kernel SVM produce more flexible decision boundary and thus can lead to better classification performance

- Choosing a good kernel is the key to success

Commonly used Kernels

- Linear kernel

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$$

- Polynomial Kernel

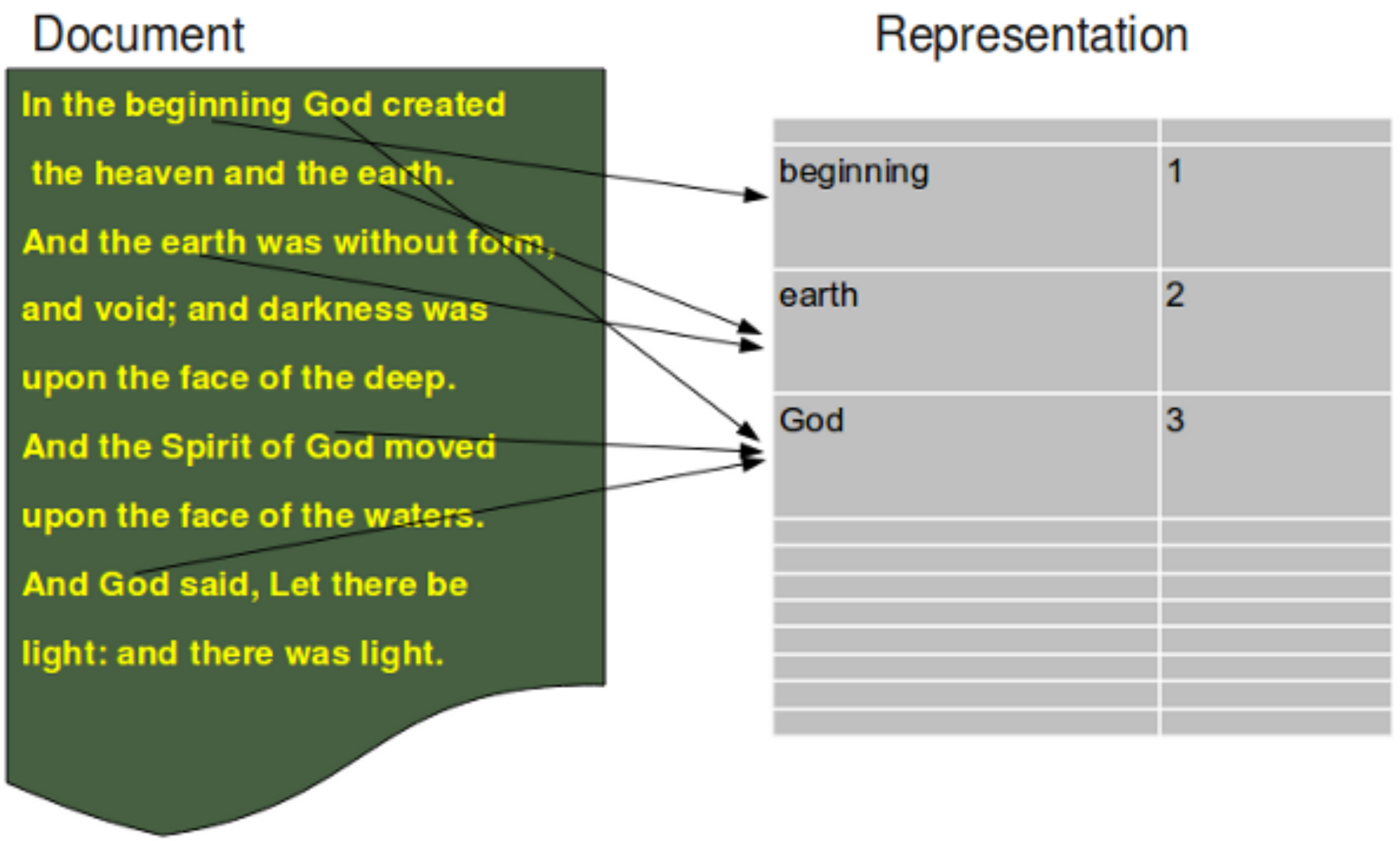
$$K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + c)^d$$

- Gaussian RBF Kernel

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{\sigma^2}\right)$$

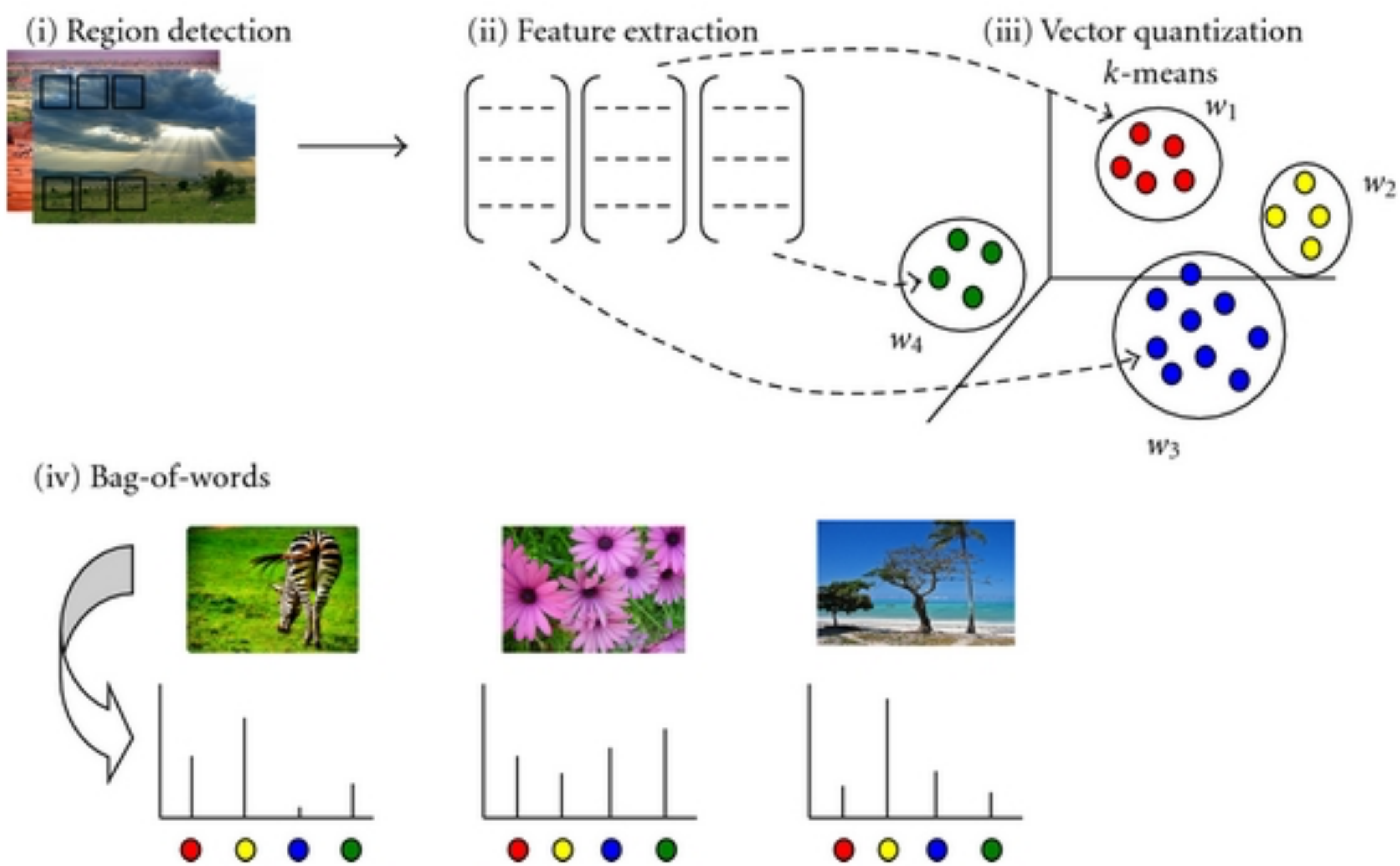
Commonly used Kernels

- Histogram like feature



Commonly used Kernels

- Histogram like feature



Commonly used Kernels

- Histogram feature
  - linear kernel or RBF kernel can be inappropriate
  - difference of frequency is affected by base-frequency

Commonly used Kernels

- Difference of frequency is affected by base-frequency
    - e.g. RBF kernel
        - “SVM” 6 times/4times vs. 2 times/0times
    - e.g. linear kernel
        - “SVM” 10 times/4times vs. 6 times/4 times

Commonly used Kernels

- Histogram feature
  - linear kernel or RBF kernel can be inappropriate
  - Improved kernel: Hellinger kernel, Histogram intersection kernel, χ^2 RBF kernel

Commonly used Kernels

- Hellinger kernel

$$K(\mathbf{x}, \mathbf{x}') = \sqrt{\mathbf{x}^T \mathbf{x}'}$$

- Histogram Intersection Kernel

$$K(\mathbf{x}, \mathbf{x}') = \sum_i \min\{x_i, x_i'\}$$

- χ^2 RBF Kernel

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma \sum_i \frac{(x_i - x_i')^2}{x_i + x_i'}\right)$$

Lesson Learned 1

- Kernel SVM produce more flexible decision boundary and thus can lead to better classification performance

- Choosing a good kernel is the key to success

Lesson Learned 2

- We need to reformulate the original form to make the problem only depend on $\langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle$
- How?
  - assume $\varphi(\mathbf{x})$ is known
  - make the term $\langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle$

Kernelize ML methods

- Revisiting the machine learning approaches
- Apply kernel tricks

Euclidean distance

$$\|\varphi(\mathbf{x}) - \varphi(\mathbf{x}')\|_2^2$$

Euclidean distance

$$\|\varphi(\mathbf{x}) - \varphi(\mathbf{x}')\|_2^2$$

$$\begin{aligned}
&= \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}) \rangle + \langle \varphi(\mathbf{x}'), \varphi(\mathbf{x}') \rangle - 2\langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle \\
&= K(\mathbf{x}, \mathbf{x}) + K(\mathbf{x}', \mathbf{x}') - 2K(\mathbf{x}, \mathbf{x}')
\end{aligned}$$

k-means

- Given an initial set of means $\{\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_k\}$ alternate 2 steps
- 1) Assign each $\mathbf{x}_i$ to the nearest centre
- 2) Based on new assignment C, recompute the centre
- Iterate until we have convergence of the means

Kernel k-means

Key: Assigning to the nearest centre: calculating distance between feature and centre

$$\mathrm{assign}(\mathbf{x}_i) = \underset{k}{argmin} \left\| \phi(\mathbf{x}_i) - \mathbf{m}_k \right\|^2$$

$$\mathbf{m}_k = \sum_i c_i^k \phi(\mathbf{x}_i)$$

$$c_i^k \in \{0, 1\}$$

Kernel PCA

- Original problem

$$\mathbf{X} \in R^{d \times N}$$

$$\mathbf{\Sigma} = \bar{\mathbf{X}}\bar{\mathbf{X}}^T$$

$$\bar{\mathbf{X}}\bar{\mathbf{X}}^T\mathbf{v} = \lambda\mathbf{v}$$

- Issue: we cannot calculate covariance matrix

Kernel PCA

- We can only calculate kernel matrix

$$\mathbf{K} = \bar{\mathbf{X}}^T \bar{\mathbf{X}}$$

$$\bar{\mathbf{X}}^T \bar{\mathbf{X}} \mathbf{v}' = \lambda \mathbf{v}'$$

$$\mathbf{K} \mathbf{v}' = \lambda \mathbf{v}'$$

- Using the following relationship

$$\bar{\mathbf{X}} \bar{\mathbf{X}}^T \bar{\mathbf{X}} \mathbf{v}' = \lambda \bar{\mathbf{X}} \mathbf{v}'$$

$$\mathbf{v} = \bar{\mathbf{X}} \mathbf{v}'$$

Kernel PCA

- When apply to new data

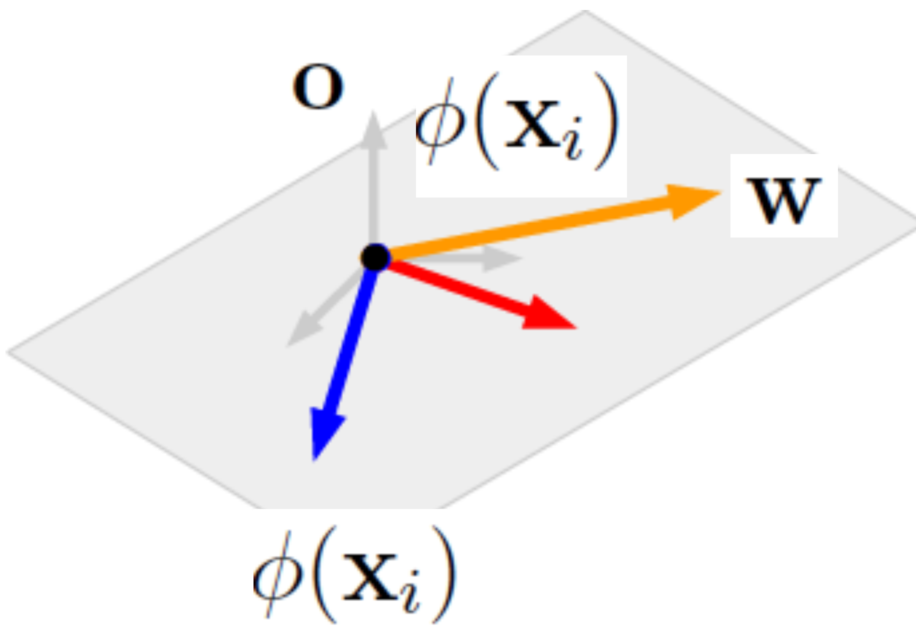
$$
\begin{aligned}
\mathbf{v}'^T\mathbf{x_t} &= \mathbf{v}'^T\bar{\mathbf{X}}'^T\mathbf{x_t} \\
&= \sum_i v'_i \langle \bar{\mathbf{x}}_\mathbf{i}, \mathbf{x_t} \rangle \\
&= \sum_i v'_i \langle \mathbf{x_i} - \frac{1}{N}\sum_j \mathbf{x_j}, \mathbf{x_t} \rangle \\
&= \sum_i v'_i \langle \mathbf{x_i}, \mathbf{x_t} \rangle - \left(\frac{1}{N}\sum_i v'_i\right)\sum_i \langle \mathbf{x_i}, \mathbf{x_t} \rangle \\
&= \sum_i v'_i k(\mathbf{x_i}, \mathbf{x_t}) - \left(\frac{1}{N}\sum_i v'_i\right)\sum_i k(\mathbf{x_i}, \mathbf{x_t})
\end{aligned}
$$

Kernel Regression

- Considering a simple regression model

$$\sum_i \|\mathbf{w}^T \phi(\mathbf{x}_i) - y_i\|_2^2$$

Kernel Regression



$$\mathbf{w} = \sum_i a_i \phi(\mathbf{x}_i) + \mathbf{o}$$

Kernel Regression

$$\sum_i \|\mathbf{w}^T\phi(\mathbf{x}_i) - y_i\|_2^2$$

$$\mathbf{w} = \sum_i a_i \phi(\mathbf{x}_i) + \mathbf{o}$$

$$\mathbf{w}^T\phi(\mathbf{x}_j) = \sum_i a_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle + \langle \mathbf{o}, \phi(\mathbf{x}_j) \rangle$$

$$= \sum_i a_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$$

Kernel Regression

$$\sum_i \|\mathbf{w}^T \phi(\mathbf{x}_i) - y_i\|_2^2$$

$$= \sum_i \| \sum_j a_j \langle \phi(\mathbf{x}_j), \phi(\mathbf{x}_i) \rangle - y_i\|_2^2$$

$$= \sum_i \|\mathbf{a}^T K(\mathbf{x}_i, \mathbf{X}) - y_i\|_2^2$$

Lesson learned

- Represent model parameters by linear combination of training features, learn the combination weight instead

- Representer theorem: https://en.wikipedia.org/wiki/Representer_theorem

Kernel LDA

- LDA: original form

$$J(\mathbf{W}) = \frac{\mathbf{w}^T \mathbf{S}_B^{\phi} \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W^{\phi} \mathbf{w}}$$

$where$

$$\mathbf{S}_B^{\phi} = (\mathbf{m}_2^{\phi} - \mathbf{m}_1^{\phi})(\mathbf{m}_2^{\phi} - \mathbf{m}_1^{\phi})^T$$

$$\mathbf{S}_W^{\phi} = \sum_{i=1,2} \sum_{n=1}^{l_i} (\phi(\mathbf{x}_n^i) - \mathbf{m}_i^{\phi})(\phi(\mathbf{x}_n^i) - \mathbf{m}_i^{\phi})^T$$

$$\mathbf{m}_i^{\phi} = \frac{1}{l_i} \sum_{n}^{l_i} \phi(\mathbf{x}_n^i)$$

Kernel LDA

- LDA: original form

$$\mathbf{w} = \sum_{i}^{l} \alpha_i \phi(\mathbf{x}_i)$$

$$J(\alpha) = \frac{\alpha^T \mathbf{M} \alpha}{\alpha^T \mathbf{N} \alpha}$$

$$where$$

$$(\mathbf{M}_i)_j = \frac{1}{l_i} \sum_{k=1}^{l_i} k(\mathbf{x}_j, \mathbf{x}_k^i)$$

$$\mathbf{N} = \sum_{j=1,2} \mathbf{K}_j (\mathbf{I} - \mathbf{1}_{l_j}) \mathbf{K}_j^T$$

Kernel LDA

- Let's try $\mathbf{w}^T \mathbf{S}_W^{\phi} \mathbf{w}$

Kernel Learning

- Problem
  - Exisiting kernel has some meta-parameters

$$\exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{\boxed{\sigma^2}}\right)$$

  - Multiple-kernel Learning

$$K(\mathbf{x},\mathbf{x}') = \sum_i \gamma_i K_i(\mathbf{x},\mathbf{x}')$$

Kernel Learning

- Choosing multiple kernel parameters
  - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.119.524&rep=rep1&type=pdf
- Multiple kernel learning (SimpleMKL)
  - http://www.jmlr.org/papers/volume9/rakotomamonjy08a/rakotomamonjy08a.pdf