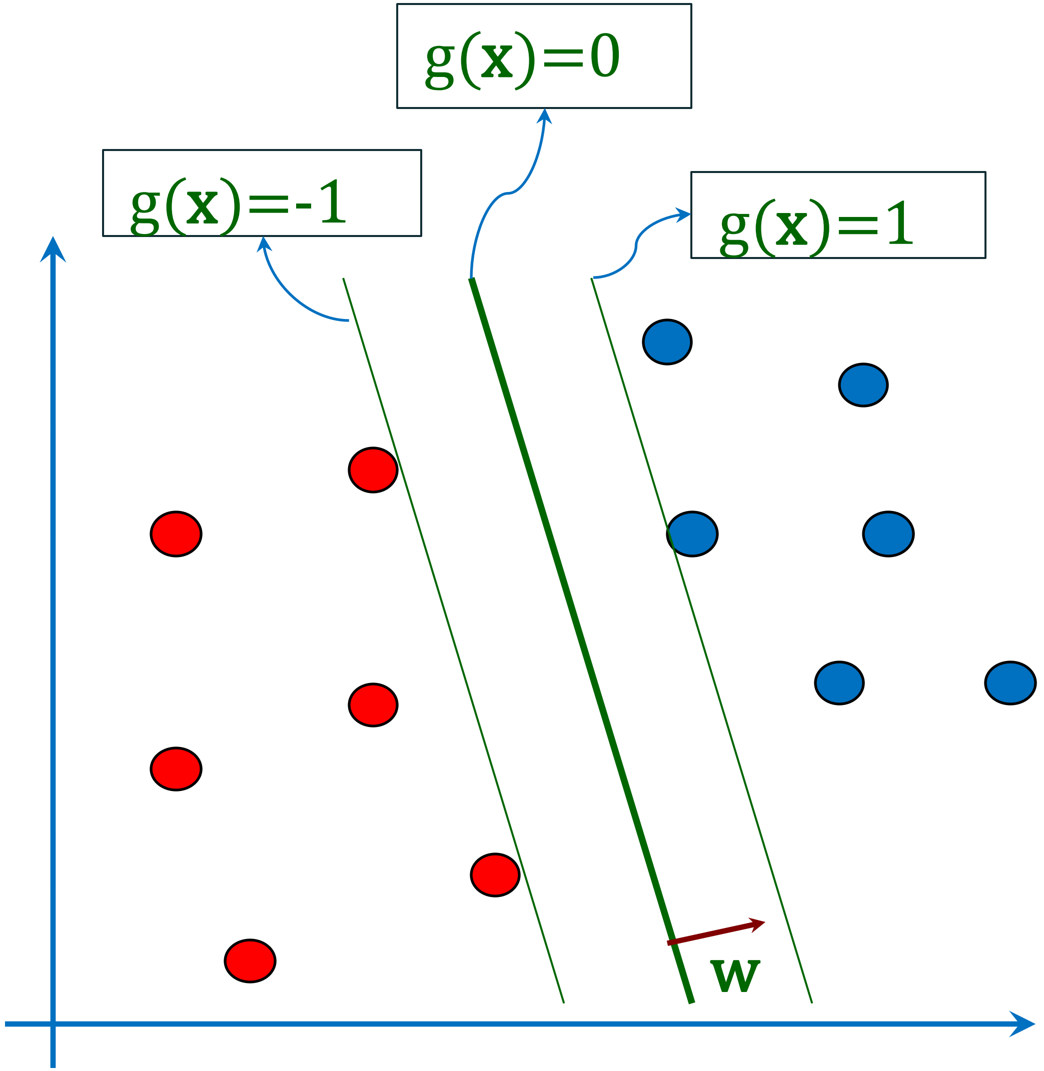
Focus for this lecture

Acquire Raw Data → Prepare / Clean Data, Visualization → Feature Engineering → Pick Model and Hyper-params for Task → Model Training / Optimization → Evaluate Model Performance → Deploy Model

- SVM Formulation
- Non-Linear Feature Mapping
    - The Kernel Trick

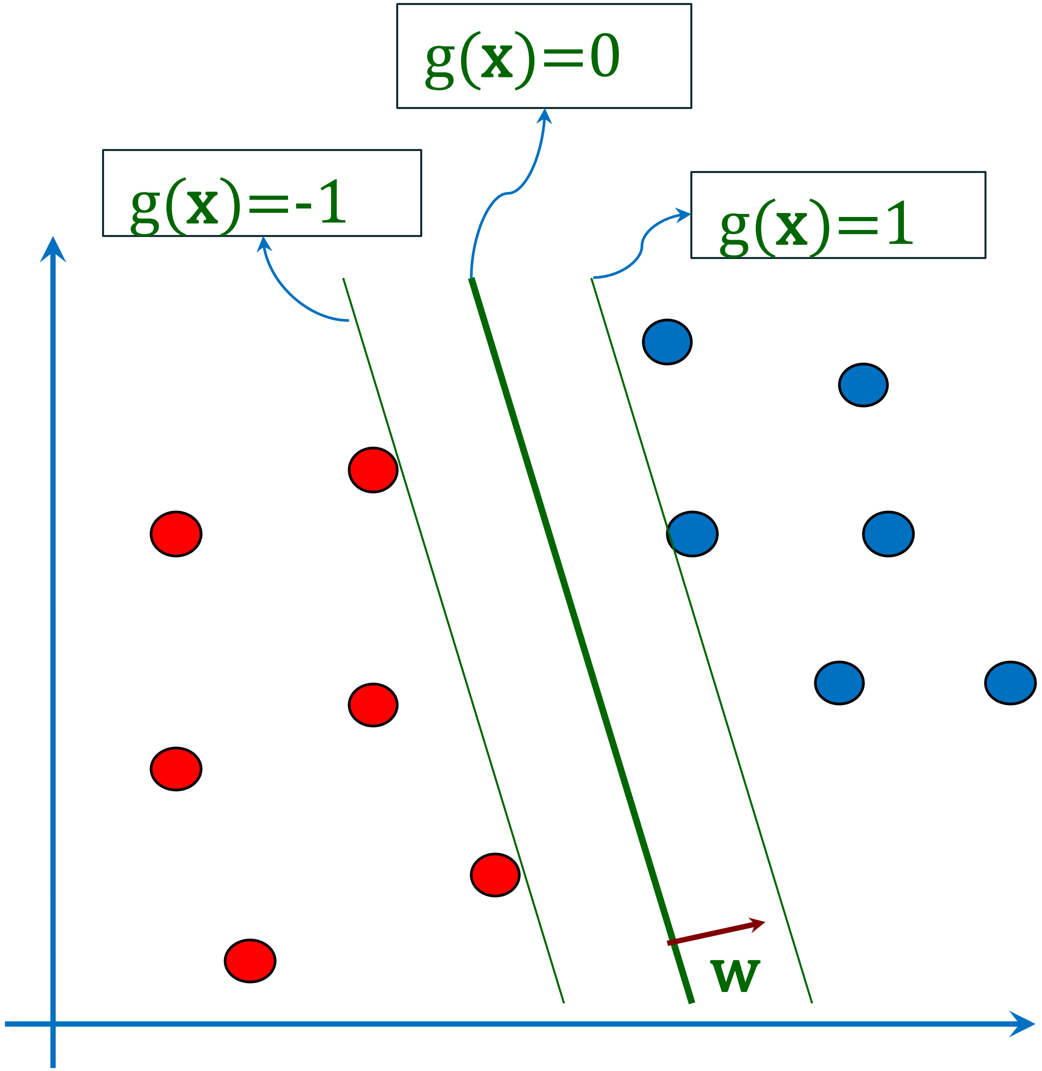
SVM: Formulation

- Let $g(X) = W^T X + b$
- We want to maximize margin:
  - $W^T X_i + b \leq -1$ for $y_i = -1$
  - $W^T X_i + b \geq 1$ for $y_i = 1$
  - Or $y_i(W^T X_i + b) \geq 1$ for i.



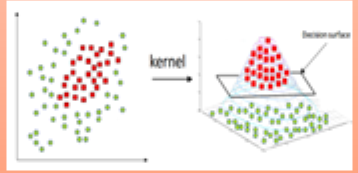
SVM: Formulation

- Mathematically, Minimize
  $\frac{1}{2}W^TW$
- Subject to:
  - $y_i(W^TX_i + b) \geq 1$ for all i.
- This is convex optimization.
- Exact solutions exist



Acquire Raw Data → Prepare / Clean Data, Visualization → Feature Engineering → Pick Model and Hyper-params for Task → Model Training / Optimization → Evaluate Model Performance → Deploy Model

- ✓ SVM Formulation
- Non-Linear Feature Mapping
  - The Kernel Trick

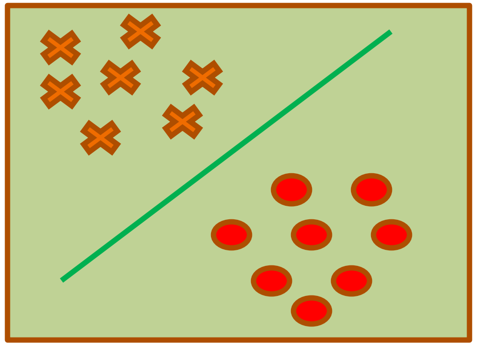


SVMs and Kernels

Kernel as Similarity Function

"Linear" Learning techniques

- Linear classifier



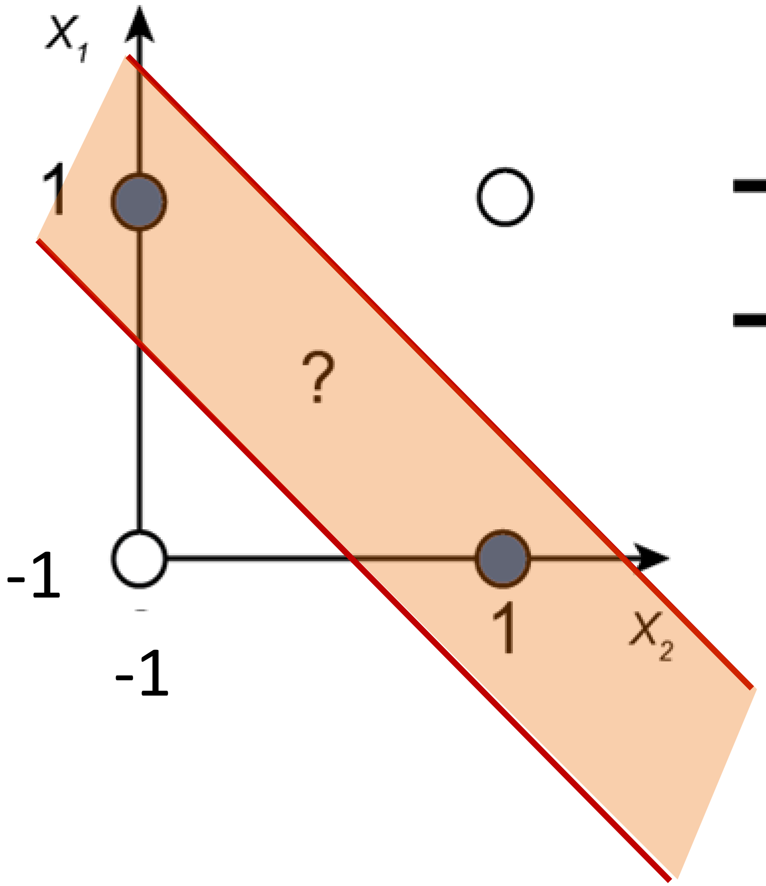
$$g(x_n) = sign(w^T x_n)$$

where *w* is an *d*-dim vector (learned)

- Techniques:
  - Perceptron
  - Logistic regression
  - Support vector machine (SVM)
  - Etc.

XOR: Limitation of Linear Methods

x_1	x_2	$x_1 \; XOR \; x_2$
-1	-1	-1 (-)
-1	1	1 (+)
1	-1	1 (+)
1	1	-1 (-)



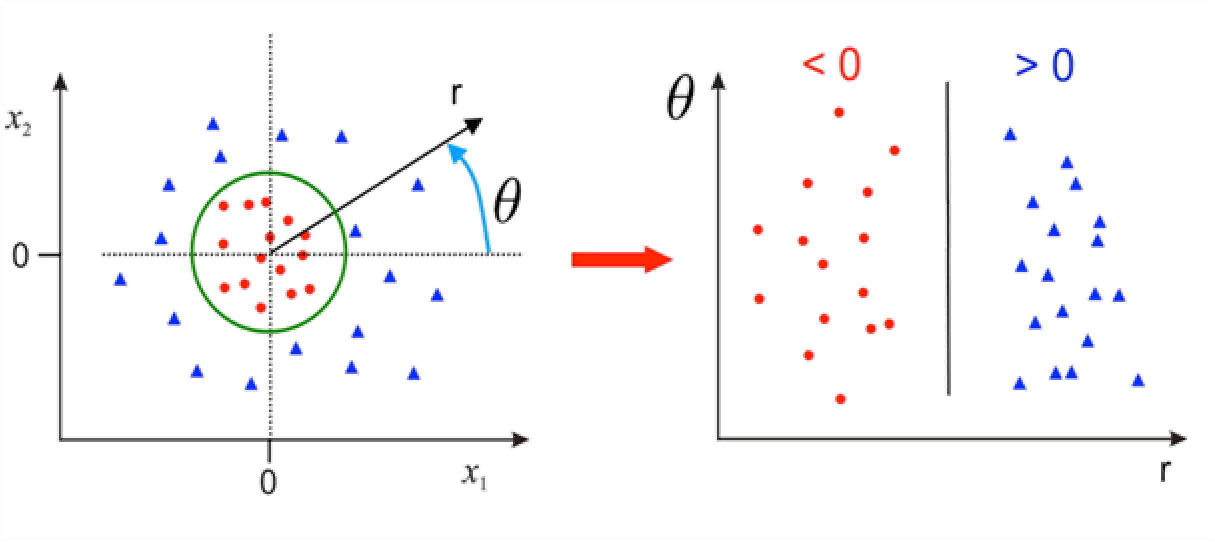
Consider a new feature x3 = x1x2

- With a “new feature”, a problem that is linearly non-separable has become separable!!

- A difficult problem became easy!!

X1	X2	X3	XOR
-1	-1	1	-
-1	1	-1	+
1	-1	-1	+
-1	-1	1	-

Nonlinearity with Feature Maps

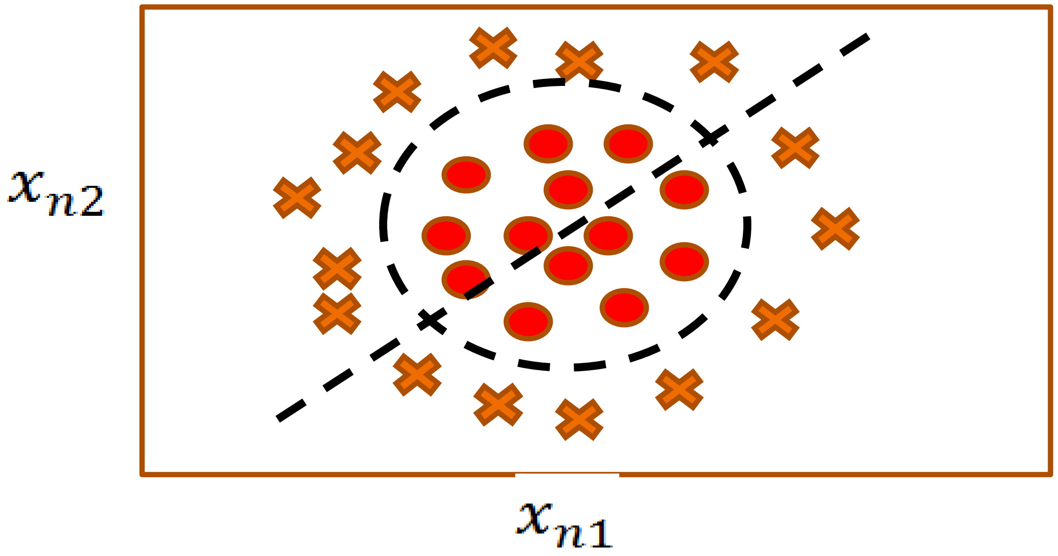


$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} r \\ \theta \end{pmatrix}$$

- With a “smart” feature map, a linearly non-separable problem can be converted to a separable problem.!!
- The feature mapping is often denoted by: $\phi(X)$

More General

- Non-linear case

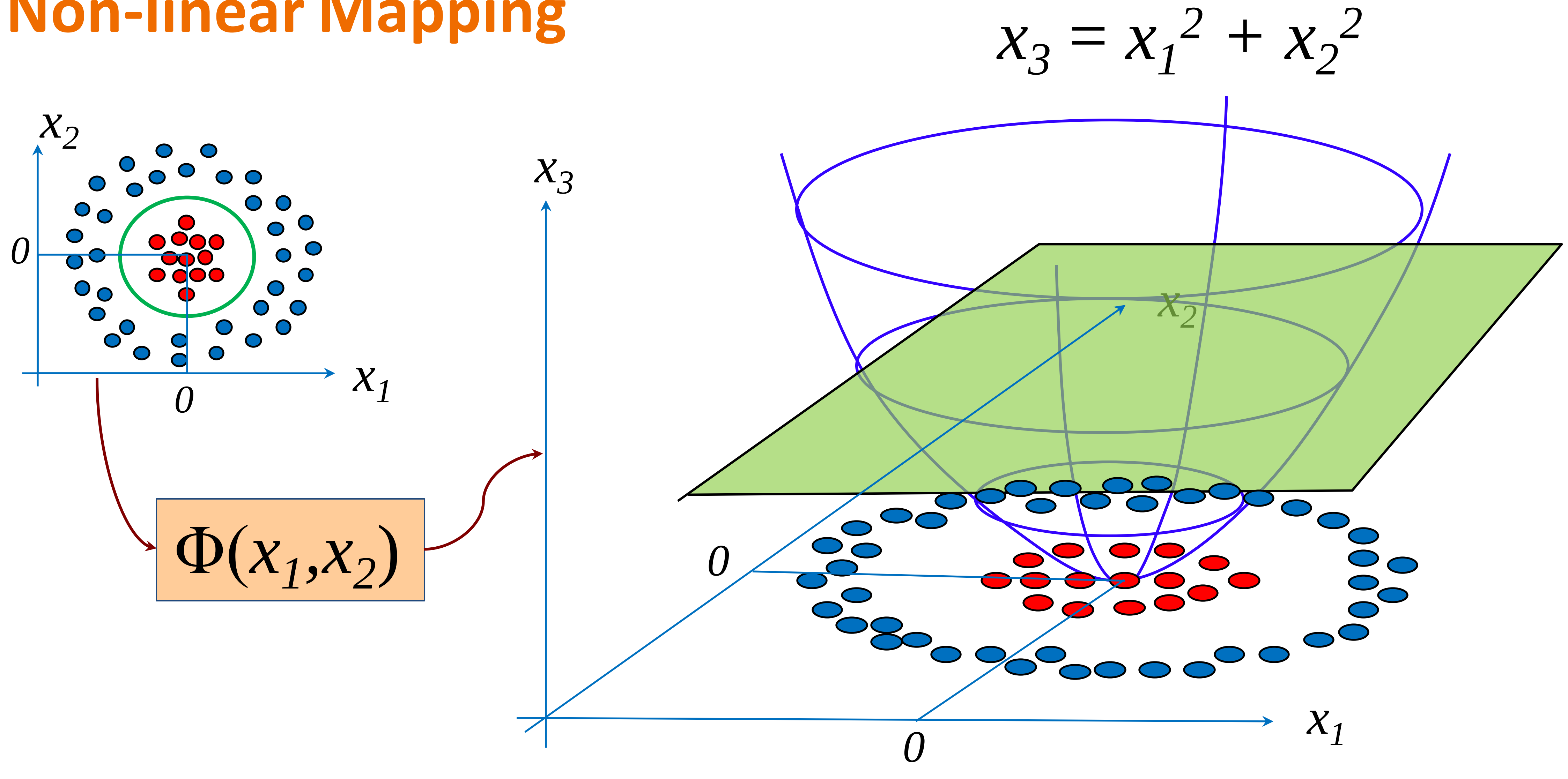


$$x_n = [x_{n1}, x_{n2}]$$

$$x_n = [x_{n1}, x_{n2}, x_{n1} * x_{n2}, {x_{n1}}^2, {x_{n2}}^2]$$

$$g(x_n) = sign(w^T x_n)$$

Non-linear Mapping



Φ is a non-linear mapping into a possibly high-dimensional space

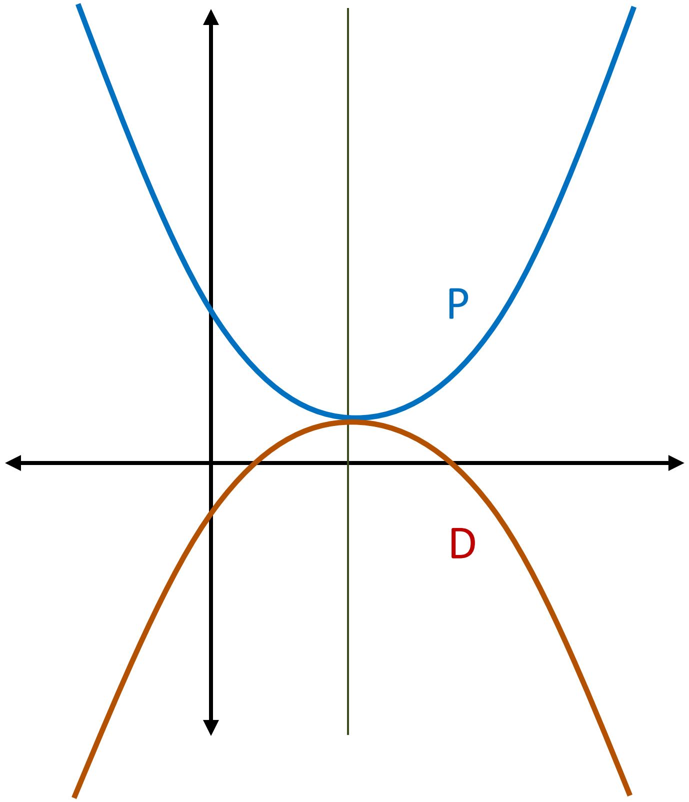
SVM: Primal and Dual

$$\text{Minimize: } \emptyset(w) = \frac{1}{2} w^T w$$

$$\text{Subject to: } y_i(w^T x_i + b) - 1 \geq 0 \ \forall i$$

$$Maximize: \ Q(\alpha) = \sum_{i=1}^{N} \propto_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \propto_i \propto_j \ y_i y_j {x_i}^T x_j$$

$$Subject\ to: \ \alpha_i \geq 0 \ \forall_i \ and \sum_{i=1}^{N} \propto_i \ y_i = 0$$



$$w_0 = \sum_{i=1}^{N} \propto_i \ y_i \ x_i$$

Kernel Strategy

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

What we need is only $w^T x = \sum_{i=1}^{N} \alpha_i y_i x_i^T x$

We can do the same in a new feature space:

$$w^T x = \sum_{i=1}^{N} \alpha_i y_i \phi(x_i)^T \phi(x) \qquad w^T x = \sum_{i=1}^{N} \alpha_i y_i K(x_i, x)$$

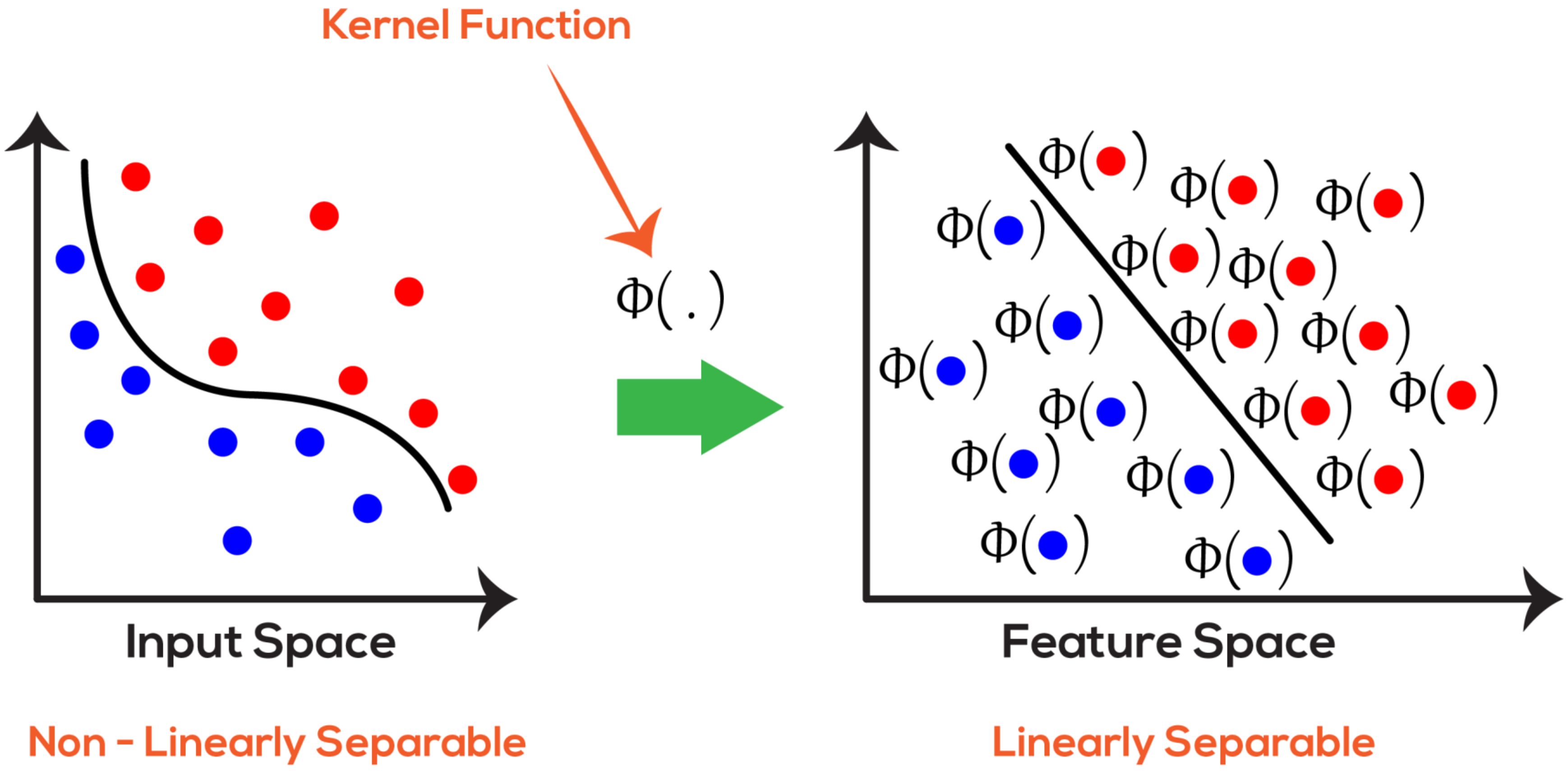
Kernels

- Interestingly, it is possible to do this without explicitly doing the non-linear mapping to high dimensions $K(x_i, x_j)$

- We need only a kernel function $K(s_i, x_i) = \emptyset(s_i).\emptyset(x_i)$

Non Linear SVM

- Transform the data to a high dimension space



Popular Kernels

- Polynomial:

$$K_p(X_1, X_2) = (1 + X_1 \cdot X_2)^p$$

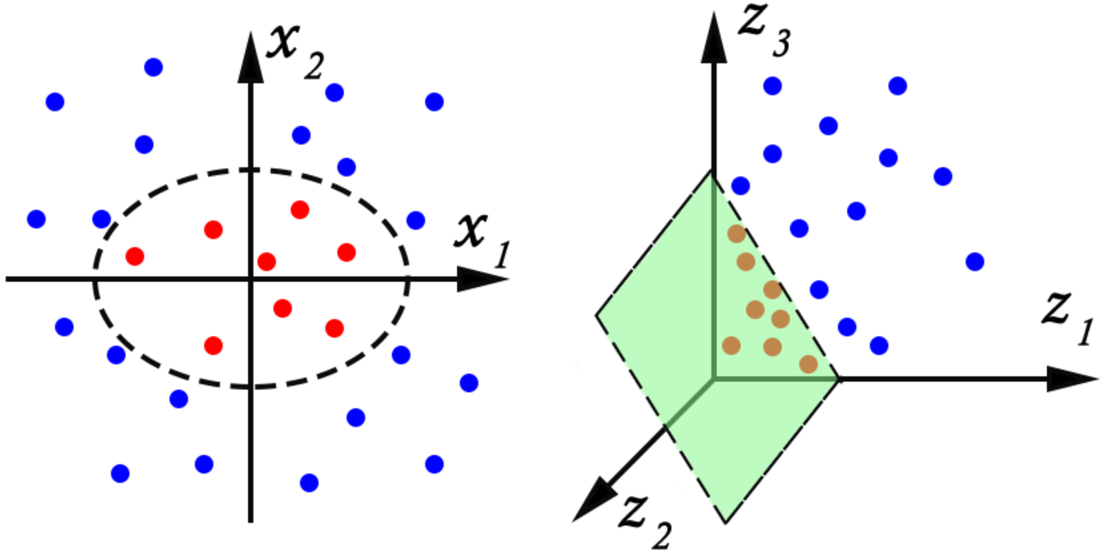
- Radial Basis Function (RBF) or Gaussian:

$$K_r(X_1, X_2) = e^{-\frac{(\|X_1 - X_2\|)^2}{2\sigma^2}}$$

- Hyperbolic Tangent:

$$K_s(X_1, X_2) = \tanh(\beta_0 X_1 \cdot X_2 + \beta_1)$$

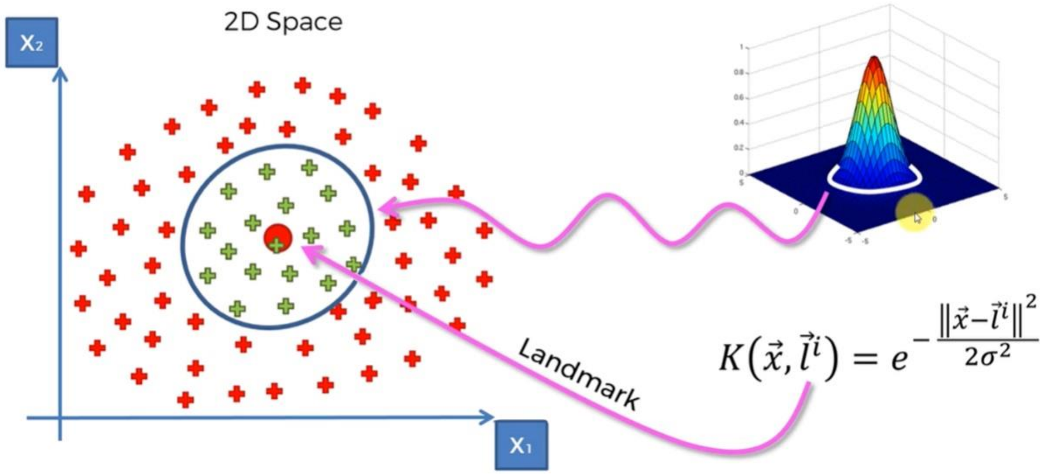
Polynomial Mapping



- $\phi : R^2 \rightarrow R^3$
- $(X_1, X_2) \rightarrow (Z_1, Z_2, Z_3) \;= (X_1^2, \sqrt{(2)}X_1X_2, X_2^2)$
- $K_p(X_1, X_2) = (1 + X_1 \cdot X_2)^p$

Radial Basis Function (RBF)

- $K_r\left(\vec{x}, \vec{l^i}\right) = e^{-\frac{\left(\left\|\vec{x}-\vec{l^i}\right\|\right)^2}{2\sigma^2}}$
- Where,
  - $\vec{x}$: X vector (some point in our dataset)
  - $\vec{l^i}$: l stands for landmarks (i represents the number of landmarks)
  - $\left(\left\|\vec{x} - \vec{l^i}\right\|\right)^2$: Distance between a point and the landmark squared



http://diamondclover.com/kernal-svm.html

Summary

- Linear SVMs generalize well, but cannot separate non-linear data
- If features can be transformed appropriately, simple linear algorithms (classification, regression) are enough.
- How do we find the feature transformation?
  – Make a reasonable guess?
  – Ans: Use some popular complex functions.
- Kernels (nonlinear) SVMs are also good at generalization and can deal with non-linear data.
- Need not be as efficient/compact.

Reference Links

- https://www.youtube.com/watch?v=9_DJ4KvyYoo

Summary

Acquire Raw Data → Prepare / Clean Data, Visualization → Feature Engineering → Pick Model and Hyper-params for Task → Model Training / Optimization → Evaluate Model Performance → Deploy Model

- ✓ SVM Formulation
- ✓ Non-Linear Feature Mapping
  - ✓ The Kernel Trick

Thanks!!

Questions?