Kernel Trick

Map feature vector to new space

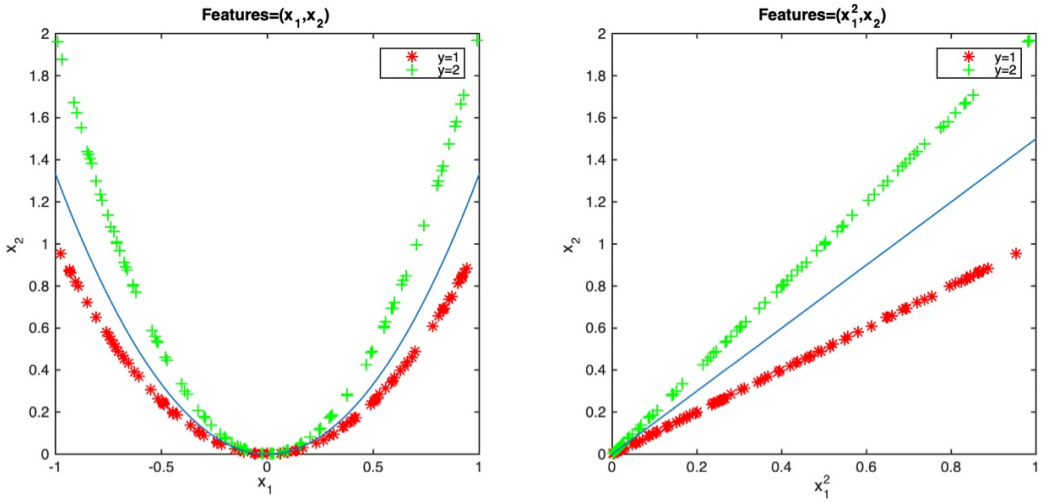
$\underline{x}$ → $\underline{\phi}(\cdot)$ → $\underline{\phi}(\underline{x})$

$\underline{x}$: d-dim

$\underline{\phi}(\cdot)$: nonlinear

$\underline{\phi}(\underline{x})$: typically of dimension > d

why? Easier to separate training data using linear boundaries in new space



Dot Products and Kernels

- Building linear classifiers often involves only dot (inner) products between vectors
  - Classifier boundary : $\underline{w}^T \underline{x} = \beta$
  - Euclidean distance : $\Delta(\underline{x}_1, \underline{x}_2) = \sqrt{(\underline{x}_1 - \underline{x}_2)^T (\underline{x}_1 - \underline{x}_2)}$
- If we map $\underline{x}$ to $\underline{\phi}(\underline{x})$, then dot products in new space formed using

$$\underline{\phi}(\underline{x})^T \underline{\phi}(\underline{y}) \triangleq \kappa(\underline{x}, \underline{y})$$ ← kernel

Mercer Kernel If K satisfies the following:

(1) $K(\cdot,\cdot)$ is continuous,

(2) $K(\cdot,\cdot)$ is symmetric, i.e., $K(\underline{x},\underline{y}) = K(\underline{y},\underline{x})$

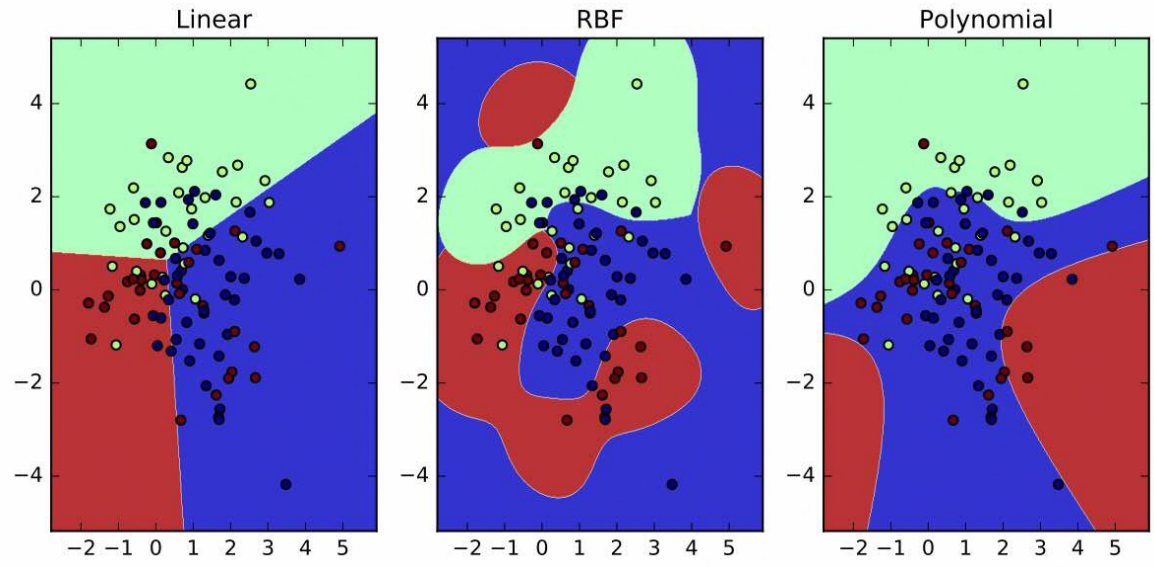
(3) $K(\cdot,\cdot)$ is positive semi-definite.

Then K is called a Mercer kernel, and for any Mercer kernel it can be shown that there exists $\underline{\phi}$ s.t. $K(\underline{x},\underline{y}) = \underline{\phi}(\underline{x})^T \underline{\phi}(\underline{y})$

Examples of Kernels

1. Linear: $K(\underline{x},\underline{y}) = c\,\underline{x}^T\underline{y}$
2. Radial Basis Function (RBF): $K(\underline{x},\underline{y}) = h(\|\underline{x}-\underline{y}\|)$ ← monotonic

   e.g. Gaussian RBF: $K(\underline{x},\underline{y}) = e^{-c\|\underline{x}-\underline{y}\|^2}$
3. Polynomial kernel: $K(\underline{x},\underline{y}) = (1+\underline{x}^T\underline{y})^l$, $l = 1, 2, \ldots$
4. Sigmoid: $K(\underline{x},\underline{y}) = \tanh(c\,\underline{x}^T\underline{y} + b)$ not Mercer



SVM

How to Handle Labelled Data

- Need to be careful not to use labelled data in way that leads to circular conclusions
- Classifier that works perfectly on training data, i.e., $Err_{train} = 0$, may perform poorly on actual inputs — poor generalization

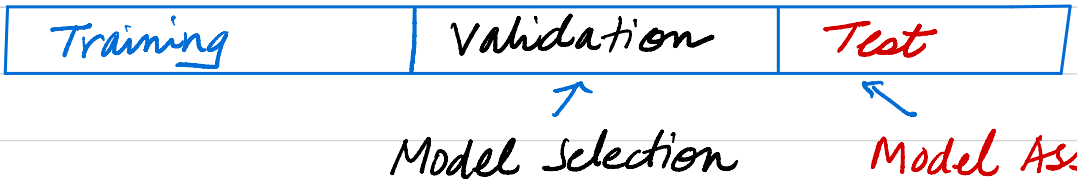
  Recall: $f(\underline{\tilde{x}}) = \begin{cases} \tilde{y} & \text{if } (\underline{\tilde{x}}, \tilde{y}) \in \mathcal{T} \\ -1 & \text{else} \end{cases}$

- Two fundamental steps in designing good $f(\cdot)$:

1. Model Selection — Given several models, estimate their performance to choose "best" one

2. Model Assessment — Having decided on best model, estimate how well it generalizes. $\hat{Err}_{pred}$

Split labelled data into 3 parts:



CAUTION Don't touch test data until Test stage

Model Selection:

- Use training data on several models; e.g. K-NN, LDA, ..
- Use Validation data to compute Err_{val} for models
- Select best — smallest Err_{val}

Cross-Validation

- In many applications, limited labelled data
- Cannot afford to split into 3 parts and use exclusively for training, validation, and testing

Step 0 Set aside test data

Step 1 split remaining labelled data into K approximately equal parts $\tau_1, \tau_2, \ldots, \tau_K$

τ_ℓ – called the ℓ-th fold

Step 2 For $\ell = 1, 2, \ldots, K$,

- Use all data except that in τ_ℓ to train models
- Use τ_ℓ to validate models

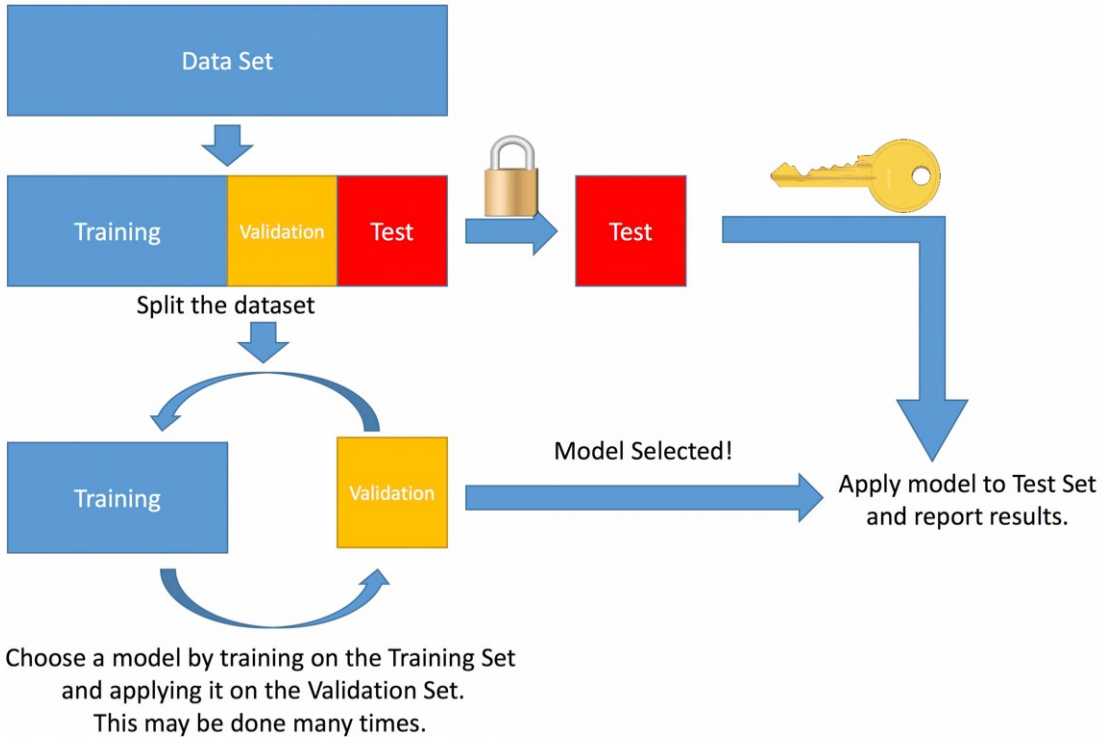
L_ℓ = validation error for a given model

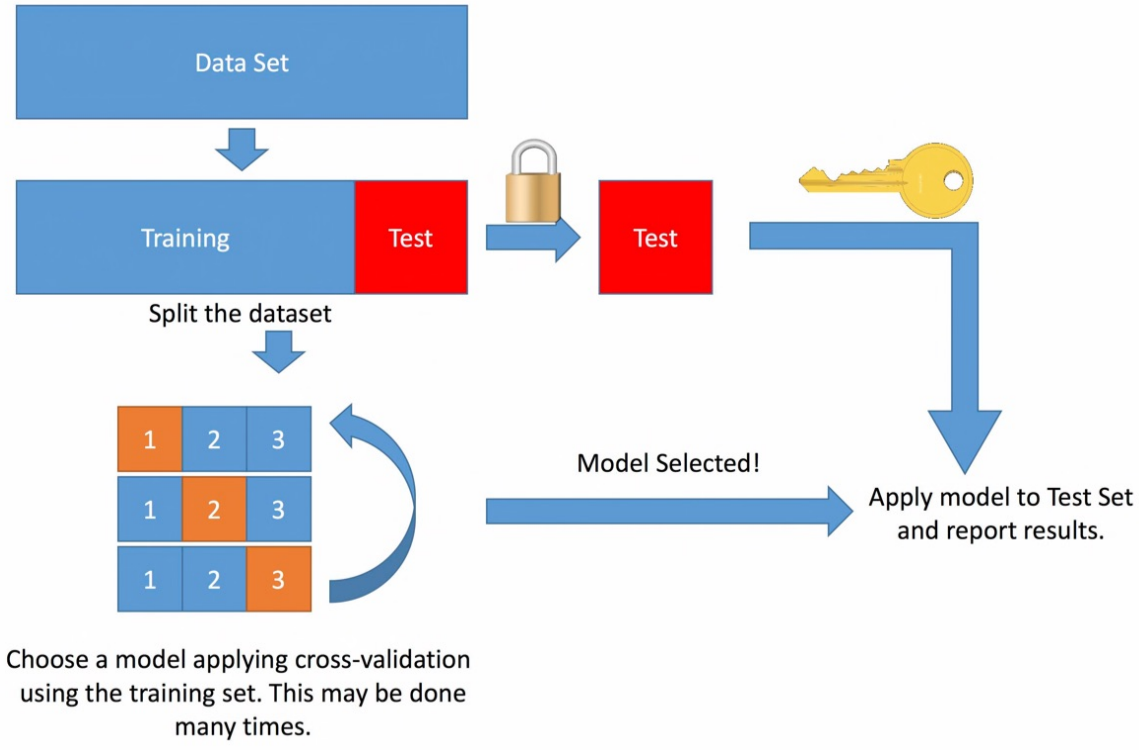
Step 3 Compute cross-validation error:

$$Err_{CV} = \frac{1}{K} \sum_{\ell=1}^{K} L_\ell .$$

Step 4 Use Err_{CV} of different models to select best

Step 5 Evaluate error of selected model on test data. This gives an accurate estimate of Err_{pred} for selected model


Data Set
Training
Validation
Test
Test
Split the dataset
Training
Validation
Model Selected!
Apply model to Test Set
and report results.
Choose a model by training on the Training Set
and applying it on the Validation Set.
This may be done many times.


Data Set
Training
Test
Test
Split the dataset
1
2
3
1
2
3
1
2
3
Model Selected!
Apply model to Test Set
and report results.
Choose a model applying cross-validation
using the training set. This may be done
many times.