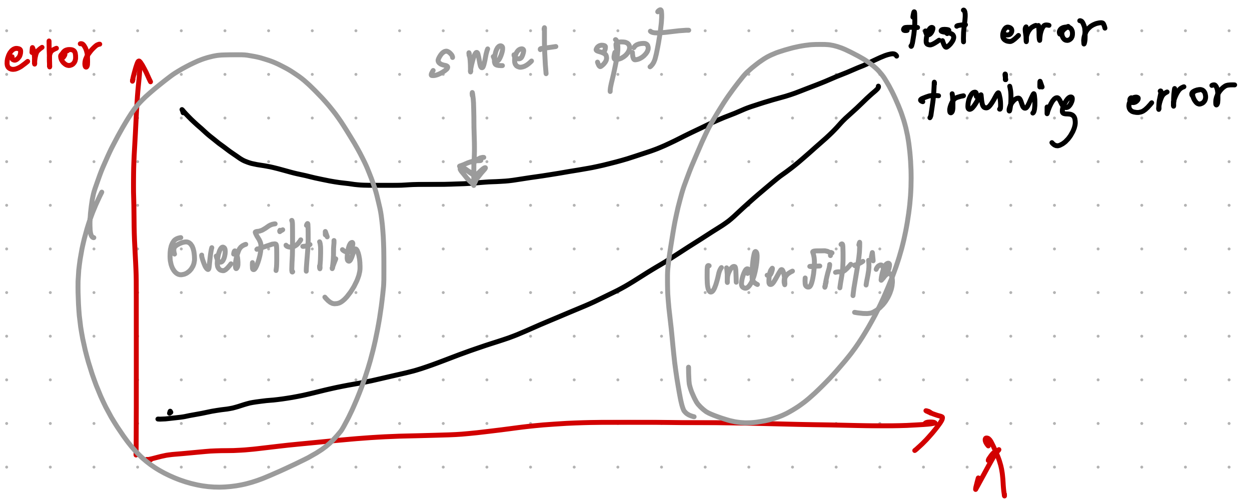
Model Selection:

- Recall ERM:

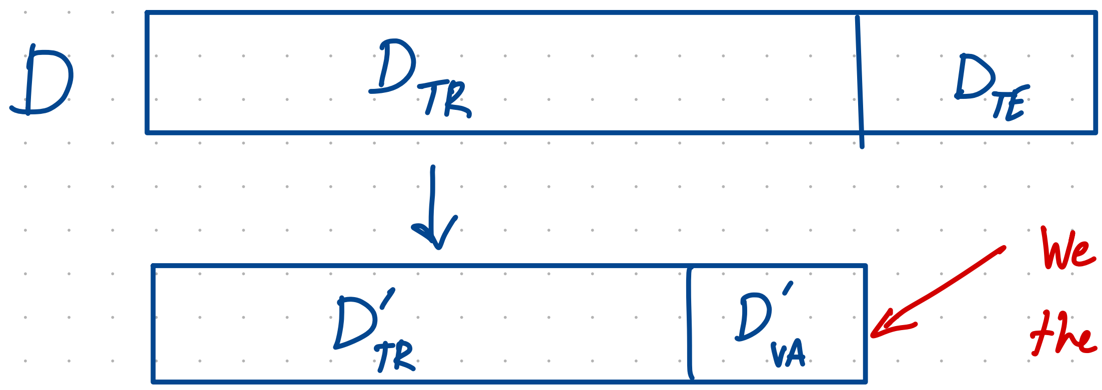
$$\boxed{\min_{\vec{w}} \frac{1}{n} \sum_{i=1}^{n} \underbrace{\ell(h_{\vec{w}}(x_i), y_i)}_{\text{loss}} + \underbrace{\lambda r(w)}_{\text{Regularizer}}}$$

- Underfitting: The solution is too simple. The trianing and test error will be high.

- Overfitting: The solution is too complex. The training error decrease over time, the test error will begin to increase.



- Identifying sweet spot: Divide data into training and validation portions. Train on the "training" split and evaluate it on the "validation" split, for various value of λ $(10^{-5}, 10^{-4}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2})$



We may overfit the validation set.

K-fold cross validation:

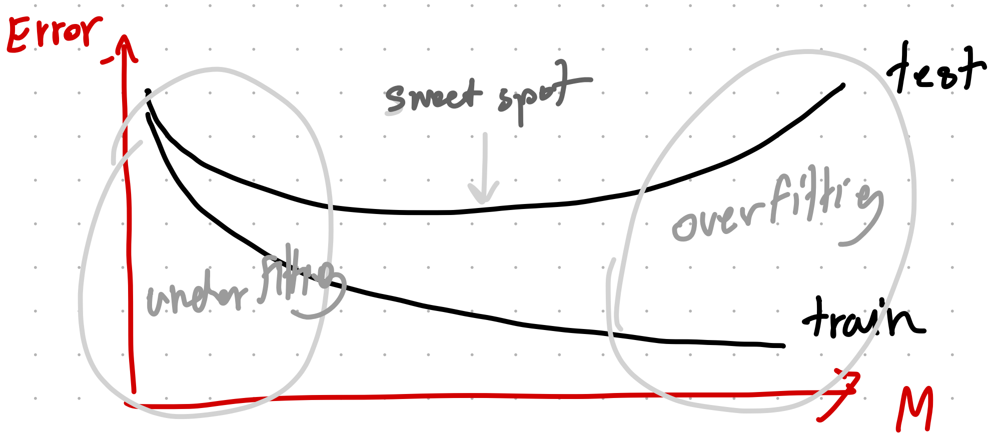
- Divide your training data into k parts.
- Train on k-1 parts and leave one out as validation set. Do this k times and averge the validation error across all runs (ideally, $K=n$).

Telescopic search for λ:

- Do two search:
  - 1st step: find the best order of magnitude for λ
    - For example, first we try 0.01, 0.1, 1, 10, 100.

      the best
  - 2nd step: do a more fine-grained search around the best λ so far
    - Then, we try 5, 10, 15, 20, 25, ..., 95 to test values around 10.

Early Stopping: Stop our optimization after $M \geq 0$ number of gradient steps, even if optimization has not converged yet.



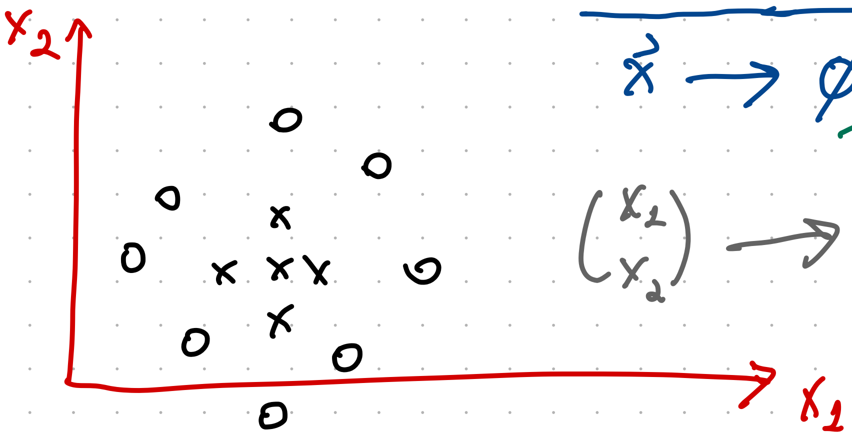
Kernels:

See visualization at: https://www.youtube.com/watch?v=3liCbRZPrZA

- A way to incorporate non-linearities into most linear classifiers.

Feature Transformation:

$$\vec{x} \to \phi(\vec{x})$$



$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \to \begin{pmatrix} x_1 \\ x_2 \\ x_1^2 + x_2^2 \end{pmatrix}$$

Extreme case:

$$\begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix} \to \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_d \\ x_1 x_2 \\ \vdots \\ x_{d-1} x_d \\ \vdots \\ x_1 x_2 \cdots x_d \end{pmatrix} \in 2^d$$