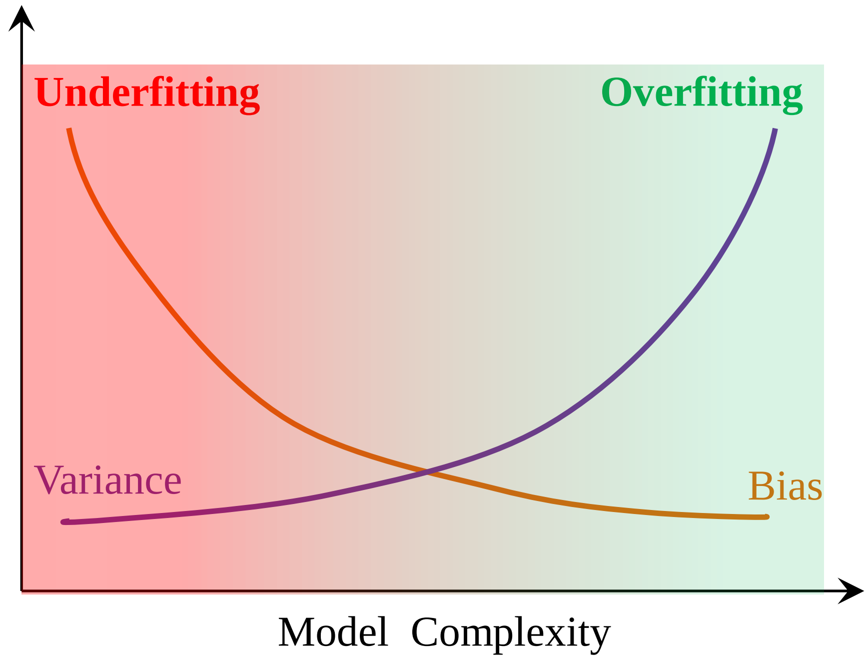
Regularization

Dripta Mj
Department of Mathematics
Ramakrishna Mission Vivekananda Educational and Research Institute
Belur Math, India

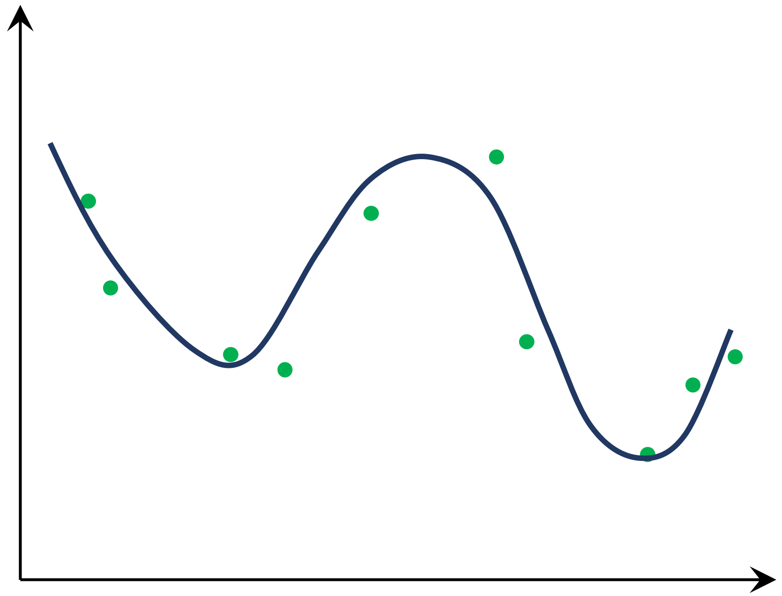
Sem 2, 2019-20

Bias-Variance trade-off



Polynomial curve fitting

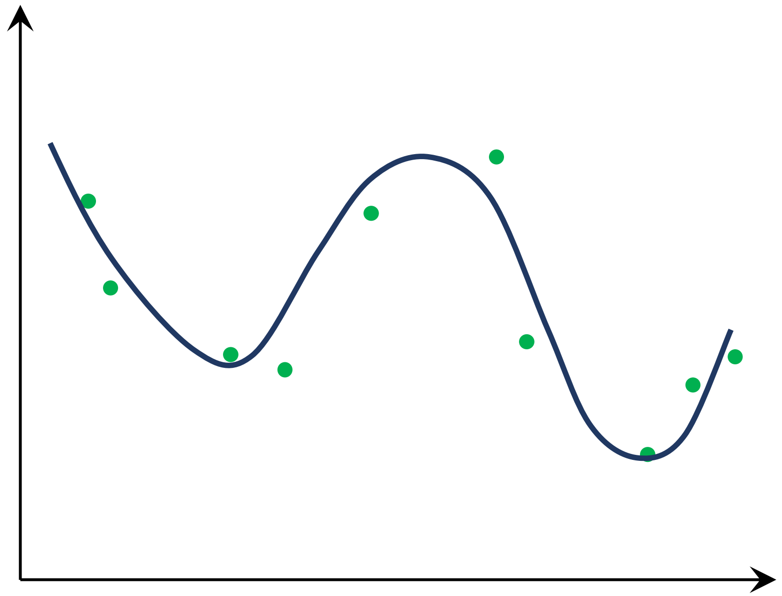
- Data generated by a Qth order polynomial + some noise



- Consider fitting data with a polynomial of order M.
- Data preprocessing:
  - Standardize the inputs.
  - Center the outputs.
- Model can be trained using linear regression with $[x^1, x^2, ..., x^M]$ as features.
- The intercept w_0 can then be ignored.

Polynomial curve fitting

- Data generated by a Qth order polynomial + some noise

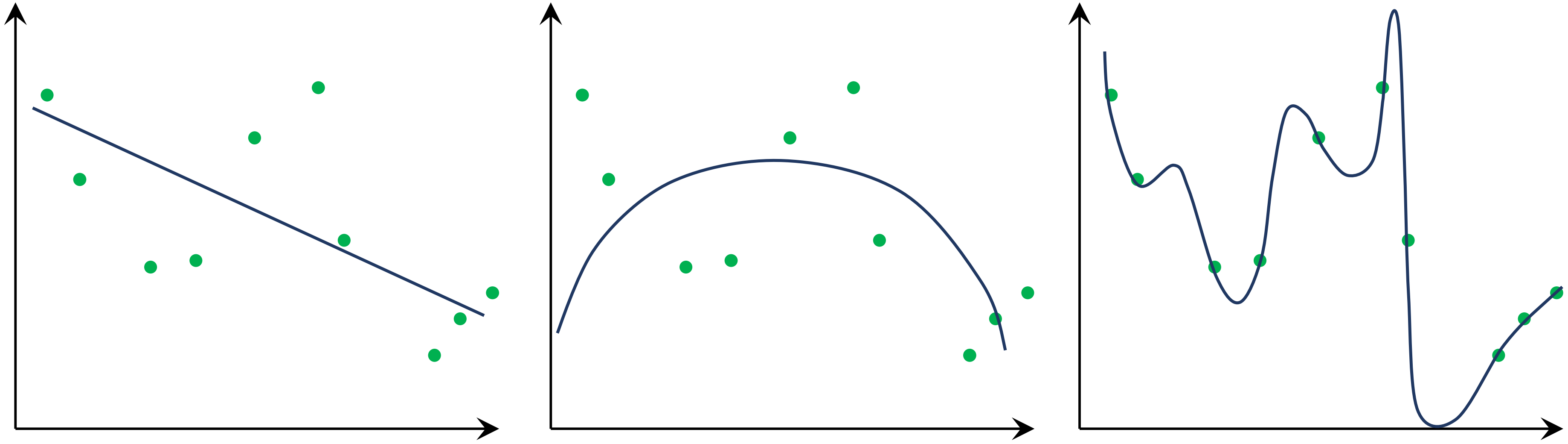


- Predictor model:

$$f(x, \mathbf{w}) = w_1 x + w_2 x^2 + w_3 x^3 + \ldots\ldots. + w_M x^M$$
$$= \sum_{i=1}^{M} w_i x^i$$
$$= \mathbf{w}^{\mathrm{T}} \phi$$

where $\mathbf{w} = [w_1, \ldots., w_M]^{\mathrm{T}}$ and $\phi = [x, \ldots., x^M]^{\mathrm{T}}$.

Polynomial curve fitting



- Complex hypotheses (richer class of models) lead to overfitting.
- A higher degree polynomial has more degrees of freedom which can lead to overfitting of the training data.
- Need to penalize the complexity in some way in the cost function.

*Figures are just for illustration.

Regularized regression

- Observations:
  - Weights $\mathbf{w}$ are unconstrained, and as such can lead to high variance.
  - Need to control the magnitude of the weights in order to control the variance.
- Modified objective:

$$\text{minimize} \quad \sum_{i=1}^{N} \left(y^{(i)} - \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}^{(i)})\right)^2 \quad \text{such that} \quad \sum_{j=1}^{M} w_j^2 \leq p$$

  - In vector form:

$$\text{minimize} \quad \left(\mathbf{y} - \Phi\mathbf{w}\right)^{\mathrm{T}}\left(\mathbf{y} - \Phi\mathbf{w}\right) \quad \text{such that} \quad ||\mathbf{w}||_2^2 \leq p$$

- Assumptions:

  Φ is standardized (zero mean and unit variance), and $\mathbf{y}$ is centered.

Regularized regression

- Can show that the problem is equivalent to:

$$\text{minimize} \qquad (\mathbf{y} - \Phi\mathbf{w})^{\mathrm{T}}(\mathbf{y} - \Phi\mathbf{w}) + \lambda||\mathbf{w}||_2^2$$

where λ is the regularization coefficient.

- λ tries to balance between the fit to the training data and the model complexity.

- Modified loss function:

$$L(\mathbf{w}) = L_E(\mathbf{w}) + \lambda L_R(\mathbf{w})$$

where

$$\boxed{\begin{aligned} L_E(\mathbf{w}) &= \sum_{i=1}^{N}\left(y^{(i)} - \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}^{(i)})\right)^2 \\ &= (\mathbf{y} - \Phi\mathbf{w})^{\mathrm{T}}(\mathbf{y} - \Phi\mathbf{w}) \end{aligned}}$$

$$\boxed{\begin{aligned} L_R(\mathbf{w}) &= \sum_{j=1}^{M} w_j^2 \\ &= ||\mathbf{w}||_2^2 \end{aligned}}$$

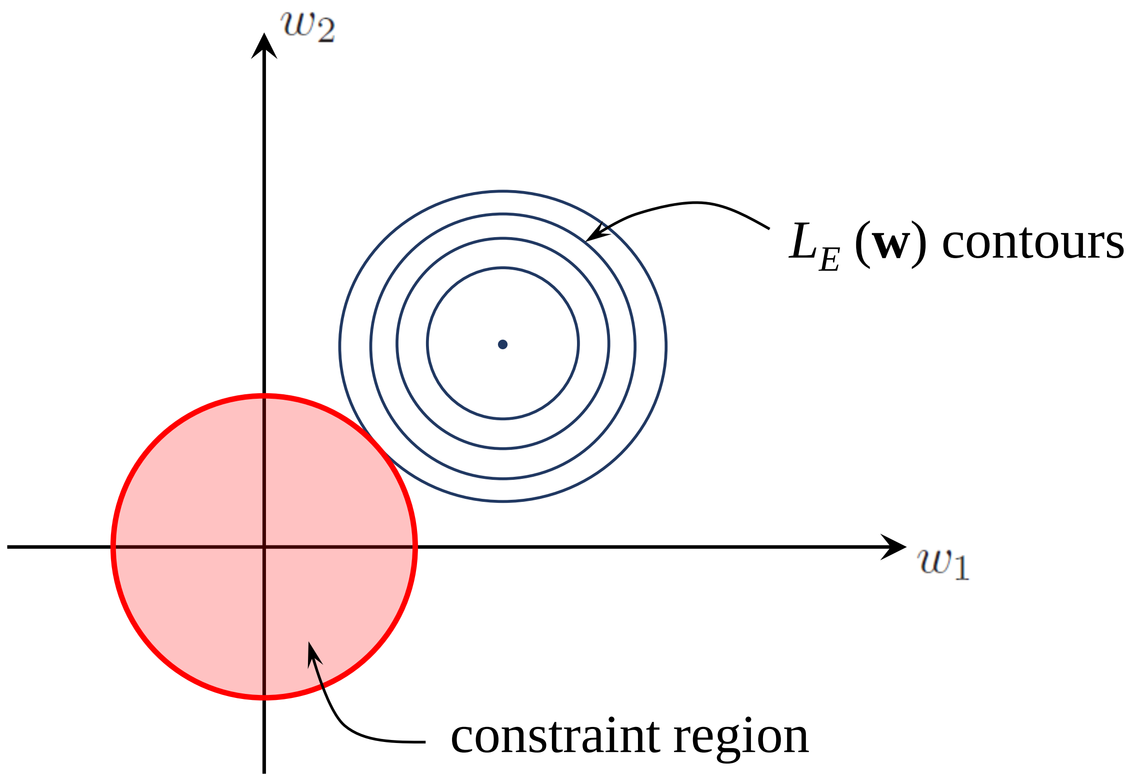
L_2 regularization

- This is known as **L_2 Regularization** and also as **Ridge Regression**.
- Goal is to minimize the loss function $L(\mathbf{w})$. Note, since $L(\mathbf{w})$ is convex it has a unique solution.
- Taking derivative of $L(\mathbf{w})$ with respect to $\mathbf{w}$ and equating it to zero $\left(\text{i.e.} \frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = 0\right)$ yields:

$$\mathbf{w} = \left(\Phi^{\mathrm{T}}\Phi + \lambda \mathbf{I}\right)^{-1}\Phi^{\mathrm{T}}\mathbf{y}$$

- If $\lambda = 0$, we get the least squares solutions.
- If $\lambda \to \infty$, we get $\mathbf{w} \to 0$.
- So $\lambda > 0$ will give weights of lower magnitudes than that obtained using least squares.

Visualization (L_2)



L_1 regularization

- Use L_1 norm of the weight vector.

$$\text{minimize} \qquad (\mathbf{y} - \Phi\mathbf{w})^{\mathrm{T}}(\mathbf{y} - \Phi\mathbf{w}) \qquad \text{such that} \qquad \sum_{j=1}^{M} |w_i| \leq p$$

- Known as the **LASSO** (least absolute shrinkage and selection operator) algorithm (*Tibshirani*, 1996).
- **LASSO** has no closed form solution unlike ridge regression.
- Can be solved using quadratic programming techniques.
- Often want some of the weights w_j's to be 0.
- **LASSO** looks for a sparse solution and so likely to yield some of the weights to be 0. But why?

L_1 regularization

- Consider a problem with two features x_1 and x_2.
- In this case we are trying to solve a optimization problem with respect to weights w_1 and w_2:

$$\text{minimize} \qquad \sum_{i=1}^{N} \left(y^{(i)} - w_1 x_1^{(i)} - w_2 x_2^{(i)} \right)^2$$

such that

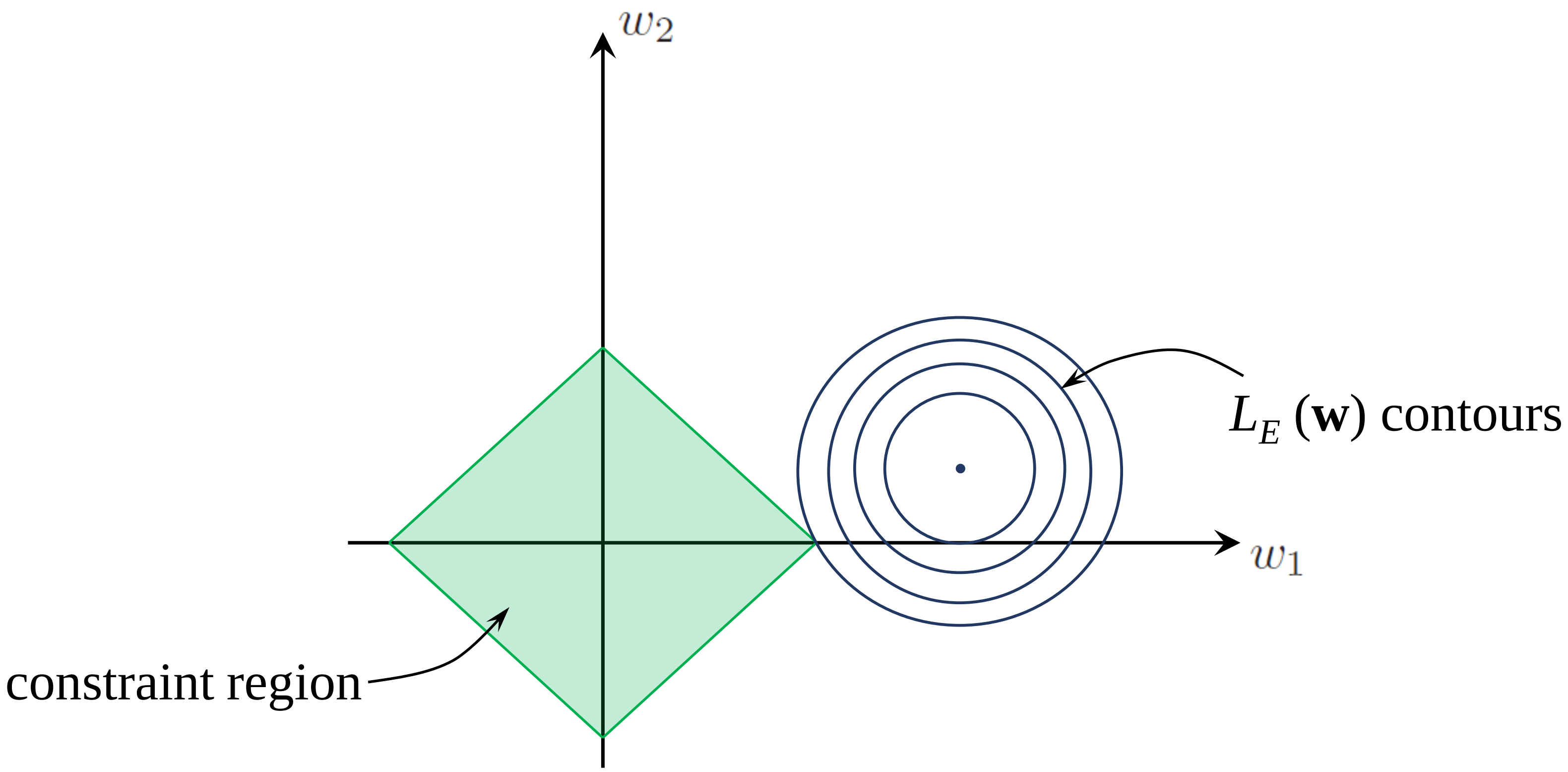
$$w_1 + w_2 \leq p$$

$$-w_1 + w_2 \leq p$$

$$w_1 - w_2 \leq p$$

$$-w_1 - w_2 \leq p$$

Visualization (L_1)



- When λ is large, then among the contours satisfying the constraints, the contour with the least value of the objective function is likely to intersect the constraints' boundary at a corner.

K-fold cross validation

- Training data is subdivided into K separate subsets – $\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_K$ of equal size (say n_K).
- For $k = 1, 2, .., K$
  - Leave out the kth fold data $\mathcal{D}_k$ and train the model on the remaining $k-1$ folds.
  - Use the trained model to make prediction on the kth fold data $\mathcal{D}_k$ and compute the (cross validation) error for this fold

$$E_k^{(\lambda)} = \frac{1}{n_K} \sum_{i=1}^{n_K} \left(y_{k,i} - f_{-k}^{(\lambda)}(\mathbf{x}_i) \right)^2$$

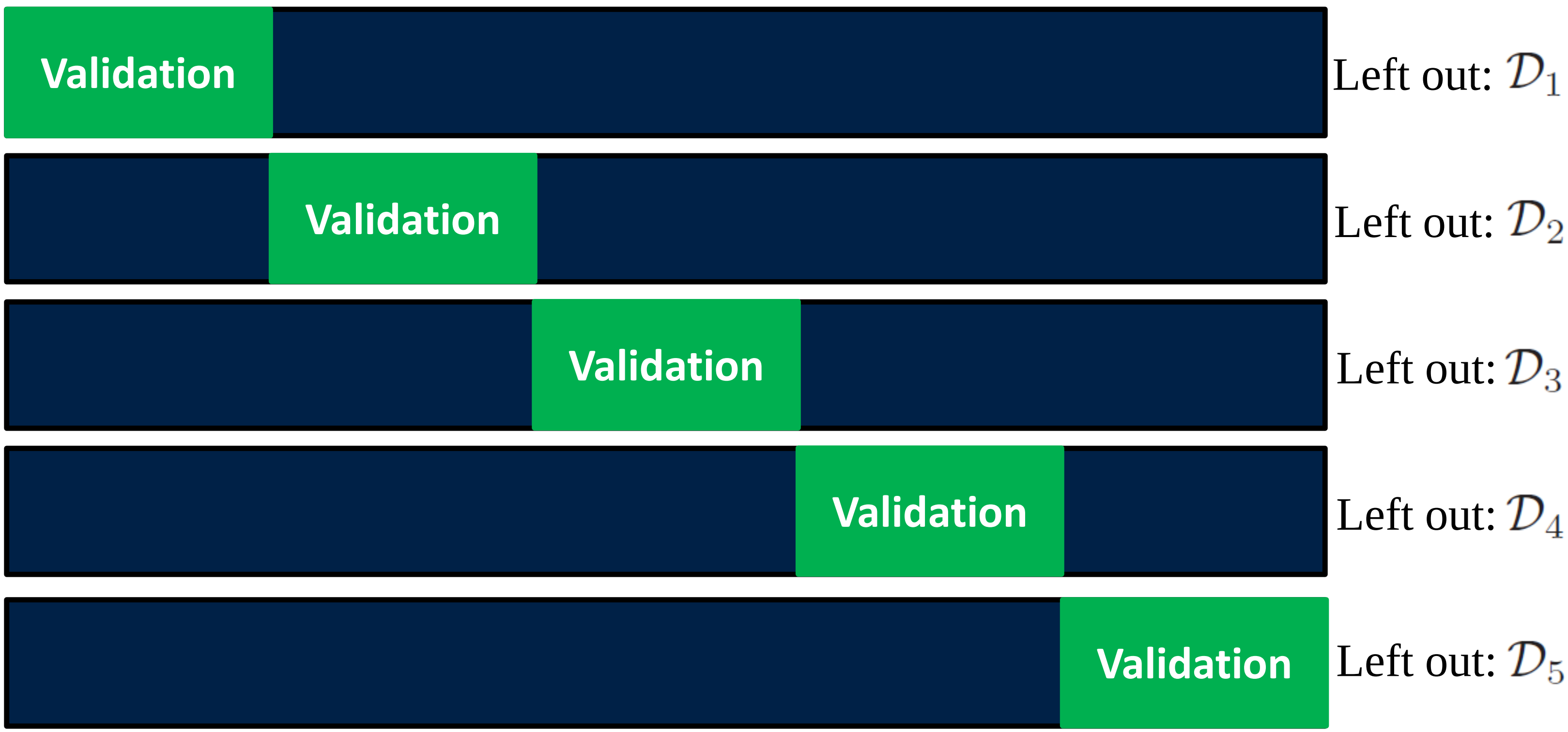
where $f_{-k}^{(\lambda)}$ is the model trained excluding the kth fold data with a specific value of λ.

K-fold cross validation

- Training data is subdivided into K separate subsets – $\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_K$ of equal size (say n_K). Let's take $K = 5$.

$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$

- Can generate K training-test datasets using the K subsets



K-fold cross validation

- For $k = 1, 2, .., K$
  - Leave out the kth fold data $\mathcal{D}_k$ and train the model on the remaining $k-1$ folds.

Left out: $\mathcal{D}_k$

  - Use the trained model to make prediction on the kth fold data $\mathcal{D}_k$ and compute the (cross validation) error for this fold

$$E_k^{(\lambda)} = \frac{1}{n_K} \sum_{i=1}^{n_K} \left(y_{k,i} - f_{-k}^{(\lambda)}(\mathbf{x}_i) \right)^2$$

where $f_{-k}^{(\lambda)}$ is the model trained excluding the kth fold data with a specific value of λ.

K-fold cross validation

- Estimated generalization error:

$$\mathbf{E}^{(\lambda)} = \frac{1}{K}\sum_{k=1}^{K} E_k^{(\lambda)}$$

- The optimal value of λ (say λ^*) is the one yielding the least value of $\mathbf{E}^{(\lambda)}$.

- Using λ^* train the model on the entire training dataset.
- When $K = N$ (size of the training dataset), the approach is known as leave-one-out cross-validation.