Lecture 8 : Cross-validation

Recap :

- We encountered 4 different methods for SUPERVISED LEARNING
  - kNN
  - Decision Trees
  - Linear regression
  - Logistic regression

  kNN and Decision Trees: Non-parametric

  Linear regression and Logistic regression: Parametric

- We also looked at regularization to keep parameters small and to prevent overfitting!

- We "train" the models to fit the training data and hope that they would give us good predictions with new, previously unseen inputs

QUESTION : Can we really expect the trained models to GENERALIZE well?

Expected new data error E_{new}

- Lets define an error function $E(y, \hat{y})$ ← it compares a prediction $\hat{y}(\underline{x})$ to a measured data point y
    - If $E(y, \hat{y})$ is small $\longrightarrow$ $\hat{y}(\underline{x})$ is a good prediction of y
    - If $E(y, \hat{y})$ is large $\longrightarrow$ " " " bad " " "

- Default choices of error functions are
    - $$E(y, \hat{y}(\underline{x})) = \begin{cases} \mathbb{I}\{\hat{y}(\underline{x}) \neq y\} & \longleftarrow \text{Misclassification (Classification)} \\ (\hat{y}(\underline{x}) - y)^2 & \longleftarrow \text{Squared error (Regression)} \end{cases}$$

- Error function $E(y, \hat{y})$ has similarities to a loss function $L(y, \hat{y})$
    - But, they are used differently
        - ↗ loss function ← used during learning
        - ↘ error function ← used after learning

Evaluating Performance

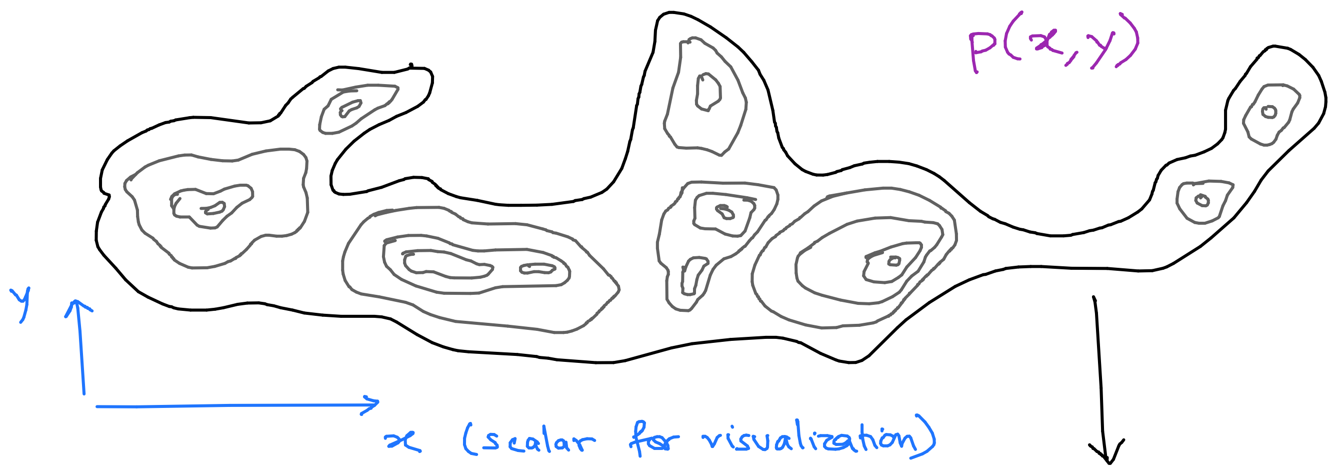
- The performance on new unseen data can be mathematically understood as the average of the error function

  e.g. how often is the classifier right?

  how well does the regression method predict?

- To be able to mathematically describe an endless stream of new unseen data, consider $p(\underline{x}, y)$ as a joint distribution over entire data $(\underline{x}, y)$

- $p(\underline{x}, y)$ can be complicated
- $\underline{x}$ is also treated as a random variable (usually not the case)
- $p(\underline{x}, y)$ remains unknown in practice



All training & test data come from this

- Let $\mathcal{T} = \{ (\underline{x}^{(i)}, y^{(i)}) \}_{i=1}^{N}$ be training data

- Average training error, $E_{train} = \frac{1}{N} \sum_{i=1}^{N} E(y^{(i)}, \hat{y}(\underline{x}^{(i)}; \mathcal{T}))$

$\hat{y}(\cdot\,; \mathcal{T})$ represents a model prediction trained using a given training dataset $\mathcal{T}$

Measures how well the learned model performs on training data

- However, we are interested in new unseen data

- So define expected new data error

$$E_{new} = \mathbb{E}_{*} \left[E(y^{*}, \hat{y}(\underline{x}^{*}; \mathcal{T})) \right]$$

$$\mathbb{E}_{x}[f(x)] = \int f(x)\, p_{x}(x)\, dx$$

$$= \int E(y^{*}, \hat{y}(\underline{x}^{*}; \mathcal{T}))\; p(\underline{x}^{*}, y^{*})\; d\underline{x}^{*}\, dy^{*}$$

- E_{new} measures how well the model generalizes from the training data $\mathcal{T}$ to new situations

- We are interested in new unseen data

- Define expected new data error

$$E_{new} = \mathbb{E}_{*}\left[E(y^*, \hat{y}(\underline{x}^*; \mathcal{T})) \right] \qquad \mathbb{E}_x[f(x)] = \int f(x)\, p_x(x)\, dx$$

$$= \int E(y^*, \hat{y}(\underline{x}^*; \mathcal{T}))\, \underset{??}{\boxed{p(\underline{x}^*, y^*)}}\, d\underline{x}^*\, dy^*$$

- E_{new} measures how well the model generalizes from the training data $\mathcal{T}$ to new situations

- E_{new} cannot be evaluated directly because $p(\underline{x}, y)$ is not known

- However, minimizing E_{new} is our ultimate goal

- Therefore, the question is : Can we approximate E_{new} is some way ?

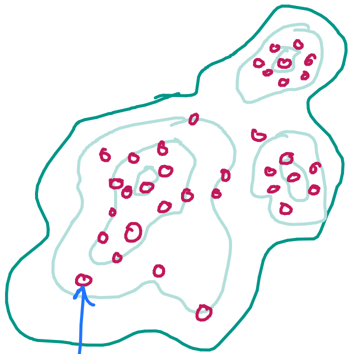
- But before that, why is estimating E_{new} so important?
  - to judge if the performance is good (whether E_{new} is small enough)
  - to choose between different ML methods
  - to choose hyperparameters → 'k' in k-NN
    → 'λ' in case of ridge regression
- Unfortunately, we cannot compute E_{new} in practice
  ↳ Therefore, we will explore a way to estimate E_{new}
    ↳ using CROSS-VALIDATION

Approximating integrals using Monte Carlo samples

- By the law of large numbers, we can approximate integrals using samples

$$\mathbb{E}_{\underline{x}}[f(\underline{x})] = \int f(\underline{x})\, p_{\underline{x}}(\underline{x})\, d\underline{x}$$

$$\approx \frac{1}{n} \sum_{i=1}^{n} f(\underline{x}^{(i)}), \qquad \underline{x}^{(i)} \overset{iid}{\sim} p(\underline{x}) \qquad i = 1, 2, \ldots, n$$



- With samples from $p(\underline{x}, y)$, we can estimate E_{new}!!

- Training data are (or should be) N samples from $p(\underline{x}, y)$
    - So can we estimate E_{new} with training data $\mathcal{T}$?

$$E_{new} = \int E(y, \hat{y}(\underline{x}; \mathcal{T}))\, p(\underline{x}, y)\, d\underline{x}\, dy$$

$$\overset{??}{\approx} \frac{1}{N} \sum_{i=1}^{N} E(y^{(i)}, \hat{y}(\underline{x}^{(i)}; \mathcal{T})) = E_{train}$$

- So can we estimate E_{new} with training data $\mathcal{T}$?

$$E_{new} = \int E(y, \hat{y}(\underline{x}; \mathcal{T}))\, p(\underline{x}, y)\, d\underline{x}\, dy$$

$$\overset{??}{\approx} \frac{1}{N} \sum_{i=1}^{N} E(y^{(i)}, \hat{y}(\underline{x}^{(i)}; \mathcal{T})) = E_{train}$$

Answer : NO

- Because the same training samples used to train the model are used to approximate the integral, hence there is an explicit dependence

$$E_{new} = \int E(y, \underbrace{\hat{y}(\underline{x}; \mathcal{T})}) \underbrace{p(\underline{x}, y)}\, d\underline{x}\, dy$$

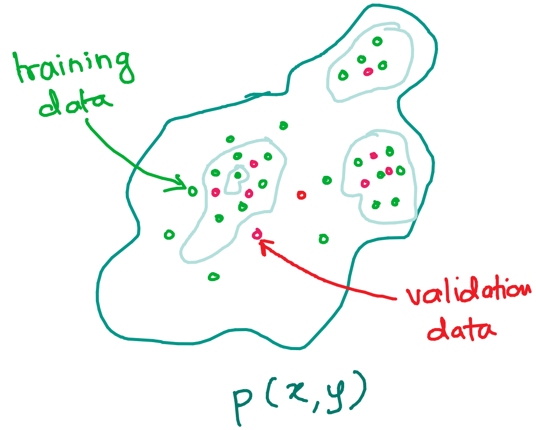
both become dependent when using training samples

- So $\boxed{E_{new} \not\approx E_{train}}$ $\Longleftrightarrow$ Performance on training data is NOT a reliable estimate of generalization

Estimating E_{new} using HOLD-OUT VALIDATION data

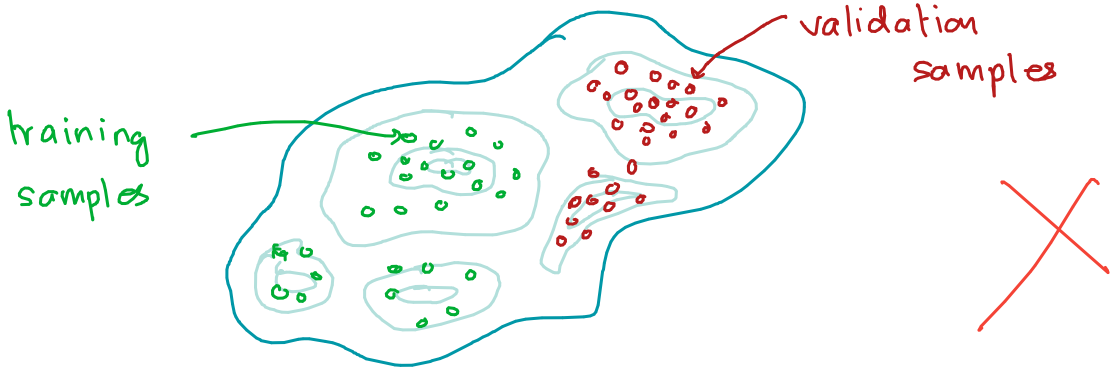
- Split data into training data and validation data

Training data, $\mathcal{T}$	Validation data
N data points	N_v data points
$\mathcal{T} = \{\underline{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$	$\{\underline{\tilde{x}}^{(j)}, \tilde{y}^{(j)}\}_{j=1}^{N_v}$



Both training & validation data are drawn from $p(\underline{x}, y)$

- When splitting the data, always do it RAMDOMLY
  - E.g., by shuffling the data points before splitting



Estimating E_{new} using HOLD-OUT VALIDATION data

- Split data into training data and validation data

$$\underbrace{\boxed{\text{Training data, } \mathcal{T}}}_{N \text{ data points}}\underbrace{\boxed{\text{Validation data}}}_{N_v \text{ data points}}$$

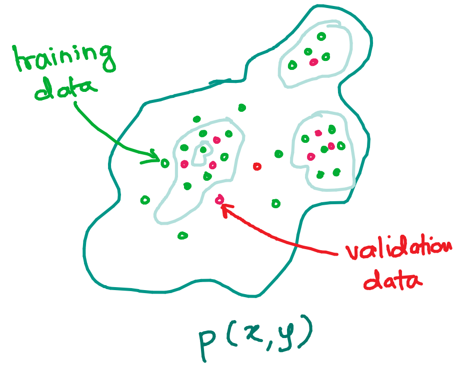
$$\mathcal{T} = \{\underline{x}^{(i)}, y^{(i)}\}_{i=1}^{N} \qquad \{\underline{\tilde{x}}^{(j)}, \tilde{y}^{(j)}\}_{j=1}^{N_v}$$

- Hold-out validation error, $E_{hold\text{-}out}$

$$E_{new} \approx E_{hold\text{-}out} = \frac{1}{N_v} \sum_{j=1}^{N_v} E\left(\tilde{y}^{(j)}, \hat{y}(\underline{\tilde{x}}^{(j)}; \mathcal{T})\right)$$

- Hold-out validation error, $E_{\text{hold-out}}$

$$E_{\text{new}} \approx E_{\text{hold-out}} = \frac{1}{N_v} \sum_{j=1}^{N_v} E\left(\tilde{y}^{(j)}, \hat{y}(\tilde{\underline{x}}^{(j)}; \mathcal{T})\right)$$



- Assuming that training and validation data points are drawn from $p(\underline{x}, y)$
  - $E_{\text{hold-out}}$ is an unbiased estimate of E_{new}
    - meaning if the entire procedure is repeated multiple times, each time with new data, the average value of $E_{\text{hold-out}}$ = E_{new}
    - However, we would not know how close $E_{\text{hold-out}}$ will be to E_{new} in a single experiment
    - The variance of $E_{\text{hold-out}}$ decreases when N_v increases; a small variance of $E_{\text{hold-out}}$ means that we can expect it to be close to E_{new}

Training data, $\mathcal{T}$	Validation data
N data points	N_v data points
$\mathcal{T} = \{\underline{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$	$\{\underline{\tilde{x}}^{(j)}, \tilde{y}^{(j)}\}_{j=1}^{N_v}$

– Hold-out validation error, $E_{\text{hold-out}}$

$$E_{new} \approx E_{\text{hold-out}} = \frac{1}{N_v} \sum_{j=1}^{N_v} E\left(\tilde{y}^{(j)}, \hat{y}(\underline{\tilde{x}}^{(j)}; \mathcal{T})\right)$$

– A good estimate of E_{new} requires a large validation set

Conflicting scenarios

– On the other hand, a good prediction requires a large training set

– Whenever there is a lot of data, the hold-out validation data approach works well

– If data is more limited, it becomes a dilemma

→ less validation data gives high variance estimate of E_{new}

→ less training data increases E_{new}

k-fold cross validation

- We would like to use all available data to train a model and at the same time have a good estimate of E_{new} for that model

$$E^{(\ell)}_{\text{hold-out}} = \frac{1}{N_v} \sum_{j=1}^{N_v} E\left(\tilde{y}^{(j,\ell)}, \hat{y}\left(\tilde{x}^{(j,\ell)}; \tau\right)\right)$$

Each $E^{(\ell)}_{\text{hold-out}}$ is an unbiased but high-variance estimate of E_{new} for the corresponding ℓth model

$$E_{k\text{-fold}} = \frac{1}{k} \sum_{\ell=1}^{k} E^{(\ell)}_{\text{hold-out}}$$

On averaging, the variance decreases and $E_{new} \approx E_{k\text{-fold}}$

- Just like hold-out validation data approach, always split the data randomly for k-fold CV to work!
    - A good approach is to first randomly permute the entire dataset and then split it into batches

- A special case when $k = N$ is called leave-one-out CV (LOO-CV)

- Advantage: Gives a very good estimate of E_{new}

- Downside of k-fold CV: Computationally demanding
    - Common choice for $k = 5, 10$

Example

Imagine we have a dataset with 6 observations

i	x	y
1	0.1	−0.3
2	0.2	−0.1
3	0.3	0.1
4	0.4	0.3
5	0.5	0.5
6	0.6	0.7

– Lets pick a value for $k = 3$ → we will use three folds to split the data

- First we shuffle the data randomly

- Then split into 3 groups

	x	y
5	0.5	0.5
2	0.2	−0.1
1	0.1	−0.3
6	0.6	0.7
4	0.4	0.3
3	0.3	0.1

Fold 1 : $\{(0.5, 0.5), (0.2, -0.1)\}$

Fold 2 : $\{(0.1, -0.3), (0.6, 0.7)\}$

Fold 3 : $\{(0.4, 0.3), (0.3, 0.1)\}$

- $k = 3$ models are trained, evaluated, and then discarded. Only scores are kept

  - Model 1 : Trained on Fold 1 + Fold 2 , Tested on Fold 3
  - Model 2 : Trained on Fold 2 + Fold 3 , Tested on Fold 1
  - Model 3 : Trained on Fold 1 + Fold 3 , Tested on Fold 2

Using a TEST Set

- An important use of $E_{k\text{-fold}}$ (or $E_{\text{hold-out}}$), in practice, is to choose between methods and select different types of hyperparameters, such as 'k' in kNN, tree depths in Decision trees, or 'λ' in ridge regression

- However, selecting $E_{k\text{-fold}}$ (or $E_{\text{hold-out}}$) for choosing hyperparameters or methods will invalidate its use as an estimator of E_{new}

- Therefore, it is wise to first set aside another hold-out dataset, which is referred to as TEST Set

Training + CV dataset	Test Set

- This test set should be used ONLY ONCE (after selecting models & hyperparameters)
    - It is used to estimate E_{new}