Classification and Representation Learning

Course 3&4 : Learning Theory

Hoel Le Capitaine

Academic year 2017-2018

Error measurement

Training vs. Generalization error

The *training error* is the error made on the training set.

- Easy to measure: number of misclassified examples divided by the total number.
- Totally irrelevant on usage: reading the training set has a training error of 0%.

What matters is the *generalization error*, which is the error that will be made on new examples (not used during learning).

- Much harder to measure (potentially infinite number of new examples, what is the correct answer?).
- Often approximated by the empirical error: one keeps a number of training examples out of the learning phase and one tests the performance on them.
- Principle of cross-validation for model selection.

More error measurements

Error may depends on the class

- False positive errors (FP, false alarm, type I) occur when the classifier predicts positive instead of negative.
- False negative errors (FN, miss, type II) occur when the classifier predicts negative instead of positive.
- is it better to fail to detect a cancer (FN), or wrongly predict one (FP) ?

Errors also include

- Recall (hit rate, sensitivity)

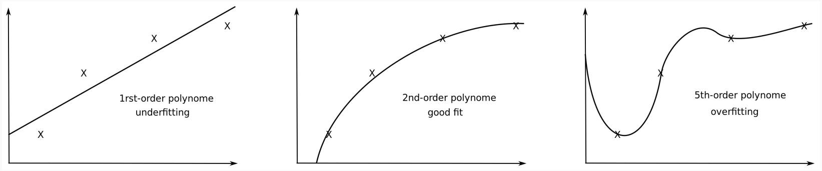
$$R = \frac{TP}{TP + FN}$$

- Precision (specificity)

$$R = \frac{TP}{TP + FP}$$

where TP = True Positive, TN = True Negative

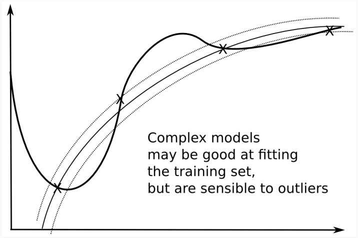
Overfitting and underfitting



The generalization error made by an hypothesis has two components

- The bias is the training error that the hypothesis would make if the training set was infinite (flexibility of the model).
- The variance is the error that will be made by the hypothesis on new examples taken from the same distribution (sensitivity of the model to outliers).

Overfitting and underfitting



Trade-off between complexity and simplicity

- The choice of the class of hypothesis is critical and should be made in relation to the data in order to optimize the bias/variance trade-off.
- The whole difficulty in machine learning is to choose a model flexible/complex enough to model the data, but not too much to avoid overfitting.
- We need an experimental procedure to select the right class of hypotheses for the given data.

Cross-validation

Cross-validation

- let us suppose we have m models $\mathcal{M} = \{M_1, \cdots, M_m\}$ that can be used to fit (classify) some data $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$
- Such a class could be the ensemble of polynomes with different orders, or different algorithms (e.g. Neural Networks, Nearest Neighbors, Naive Bayes, SVM), or the same algorithm with different values for the meta-parameters (learning rate...).
- how to find the best model ?

Cross-validation

Algorithm 1: Naive error estimation

```
forall models $M_i$:
    Train $M_i$ on $\mathcal{D}$ to obtain an hypothesis $h_i$
    Compute the training error $\epsilon_{\mathcal{D}}(h_i)$ of $h_i$ on $\mathcal{D}$
```

$$\epsilon_{\mathcal{D}}(h_i) = \sum_{(\mathbf{x},y)\in\mathcal{D}} \mathbf{1}_{h_i(\mathbf{x})\neq y}$$

Select the hypothesis $h_i^\star$ with minimal training error

$$h_i^\star = \mathrm{argmin}_{h_i\in\mathcal{M}} \epsilon_{\mathcal{D}}(h_i)$$

Cross-validation

Algorithm 1: Naive error estimation

```
forall models $M_i$:
    Train $M_i$ on $\mathcal{D}$ to obtain an hypothesis $h_i$
    Compute the training error $\epsilon_{\mathcal{D}}(h_i)$ of $h_i$ on $\mathcal{D}$
        $$\epsilon_{\mathcal{D}}(h_i) = \sum_{(\mathbf{x},y)\in\mathcal{D}} \mathbf{1}_{h_i(\mathbf{x})\neq y}$$
Select the hypothesis $h_i^\star$ with minimal training error
    $$h_i^\star = \text{argmin}_{h_i\in\mathcal{M}}\epsilon_{\mathcal{D}}(h_i)$$
```

This leads to overfitting, only the training error is used for selection

Simple hold-out cross-validation

Idea: split the training data into a *training set* (used for learning the models) and a *test set* (used to estimate the generalization error through the empirical error).

Algorithm 2: Simple hold-out cross-validation

```
Split the training data $\mathcal{D}$ into $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$ (typically 70%/30%)
forall models $M_i$:
        Train $M_i$ on $\mathcal{D}_{train}$ to obtain $h_i$
        Compute the empirical error $\epsilon_{test}(h_i)$ on $\mathcal{D}_{test}$
            $\epsilon_{test}(h_i) = \sum_{(\mathbf{x},y)\in\mathcal{D}_{test}} \mathbb{1}_{h_i(\mathbf{x})\neq y}$
Select the hypothesis $h_i^\star$ with minimal empirical error
    $h_i^\star = \mathrm{argmin}_{h_i\in\mathcal{M}} \epsilon_{test}(h_i)$
```

Simple hold-out cross-validation

Idea: split the training data into a *training set* (used for learning the models) and a *test set* (used to estimate the generalization error through the empirical error).

Algorithm 2: Simple hold-out cross-validation

```
Split the training data $\mathcal{D}$ into $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$ (typically 70%/30%)
forall models $M_i$:
        Train $M_i$ on $\mathcal{D}_{train}$ to obtain $h_i$
        Compute the empirical error $\epsilon_{test}(h_i)$ on $\mathcal{D}_{test}$
            $\epsilon_{test}(h_i) = \sum_{(\mathbf{x},y)\in\mathcal{D}_{test}} \mathbf{1}_{h_i(\mathbf{x})\neq y}$
Select the hypothesis $h_i^\star$ with minimal empirical error
    $h_i^\star = \text{argmin}_{h_i\in\mathcal{M}} \epsilon_{test}(h_i)$
```

- pros : does not use training set for method selection
- cons : some portion of the data is not used for learning, can be a problem if data is rare or expensive

k-fold cross validation

Idea: build several different test sets with the same training data, learn the models on the rest and take the hypothesis that works best on average.

Algorithm 3: k-fold cross validation

```
Randomly split the training data $\mathcal{D}$ into $k$ subsets $\{\mathcal{D}_1, \cdots, \mathcal{D}_k\}$ (e.g. $k = 10$), where $|\mathcal{D}_j| = N/k$ for all $j$
forall models $M_i$:
    forall subsets $\mathcal{D}_j$
        Train $M_i$ on $\mathcal{D} \backslash \mathcal{D}_j$ to obtain $h_{ij}$
        Compute the empirical error $\epsilon_{\mathcal{D}_j}(h_{ij})$ on $\mathcal{D}_j$
    The empirical error of the model $M_i$ on $\mathcal{D}$ is the average of empirical errors made on $\{\mathcal{D}_j\}_{j=1}^{\frac{N}{k}}$
        $$\epsilon_{\mathcal{D}}(M_i) = \frac{k}{N} \sum_{j=1}^{\frac{N}{k}} \epsilon_{\mathcal{D}_j}(h_{ij})$$
Select the model $M_i^\star$ with minimal empirical error on $\mathcal{D}$
```

k-fold cross validation

Idea: build several different test sets with the same training data, learn the models on the rest and take the hypothesis that works best on average.

Algorithm 3: k-fold cross validation

```
Randomly split the training data $\mathcal{D}$ into $k$ subsets $\{\mathcal{D}_1, \cdots, \mathcal{D}_k\}$ (e.g. $k = 10$),
where $|\mathcal{D}_j| = N/k$ for all $j$
forall models $M_i$:
        forall subsets $\mathcal{D}_j$
                Train $M_i$ on $\mathcal{D} \backslash \mathcal{D}_j$ to obtain $h_{ij}$
                Compute the empirical error $\epsilon_{\mathcal{D}_j}(h_{ij})$ on $\mathcal{D}_j$
        The empirical error of the model $M_i$ on $\mathcal{D}$ is the average of empirical
errors made on $\{\mathcal{D}_j\}_{j=1}^{\frac{N}{k}}$

        $$\epsilon_{\mathcal{D}}(M_i) = \frac{k}{N} \sum_{j=1}^{\frac{N}{k}} \epsilon_{\mathcal{D}_j}(h_{ij})$$

Select the model $M_i^\star$ with minimal empirical error on $\mathcal{D}$
```

- pros : works well
- cons : needs time ...

PAC learning theory and VC dimension

PAC learning theory

Definitions

- An hypothesis h_i belongs to a class of hypothesis (or model) $\mathcal{H} = \{h_1, \cdots, h_k\}$ that may be infinite
- Training error $\hat{\epsilon}(h)$ of an hypothesis h on a training data $(\mathbf{x}_i, y_i) \in \mathcal{D}$

$$\hat{\epsilon}(h) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}_{h(\mathbf{x}_i) \neq y_i}$$

- Generalization error $\epsilon(h)$ of an hypothesis the whole distribution $\mathcal{D}$

$$\epsilon(h) = \sum_{(\mathbf{x},y) \in \mathcal{D}} P(h(\mathbf{x}) \neq y)$$

- we say that h is probably approximately correct (PAC) on $\mathcal{D}$ with $(\epsilon > 0, \delta > 0)$ if

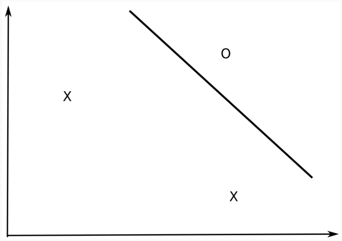
$$P(\epsilon(h) < \epsilon) = 1 - \delta$$

- ϵ is the accuracy parameter (the maximally tolerated error), δ is the confidence parameter (the probability that it fails).

Separability

How many examples can be correctly classified by a linear model in $\mathbb{R}^d$?

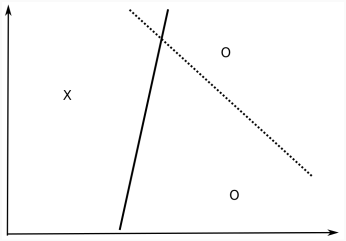
In $\mathbb{R}^2$, all dichotomies of three non-aligned examples can be correctly classified by a linear model ($\mathcal{H} = \{w_0 + w_1 x_1 + w_2 x_2\}$)



Separability

How many examples can be correctly classified by a linear model in $\mathbb{R}^d$?

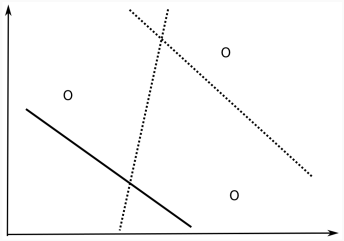
In $\mathbb{R}^2$, all dichotomies of three non-aligned examples can be correctly classified by a linear model ($\mathcal{H} = \{w_0 + w_1 x_1 + w_2 x_2\}$)



Separability

How many examples can be correctly classified by a linear model in $\mathbb{R}^d$?

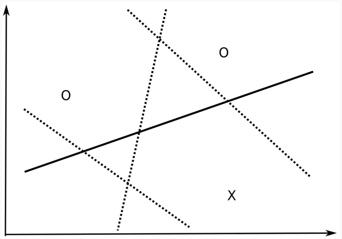
In $\mathbb{R}^2$, all dichotomies of three non-aligned examples can be correctly classified by a linear model ($\mathcal{H} = \{w_0 + w_1x_1 + w_2x_2\}$)



Separability

How many examples can be correctly classified by a linear model in $\mathbb{R}^d$?

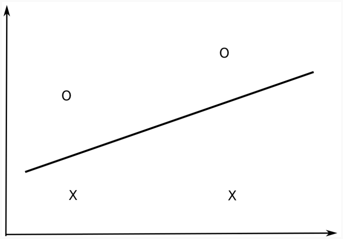
In $\mathbb{R}^2$, all dichotomies of three non-aligned examples can be correctly classified by a linear model ($\mathcal{H} = \{w_0 + w_1 x_1 + w_2 x_2\}$)



Separability

How many examples can be correctly classified by a linear model in $\mathbb{R}^d$?

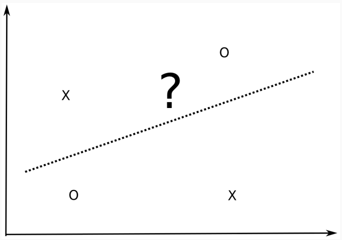
However, there exists sets of four examples in $\mathbb{R}^2$ that cannot be correctly classified by a linear model (not linearly separable)



Separability

How many examples can be correctly classified by a linear model in $\mathbb{R}^d$?

However, there exists sets of four examples in $\mathbb{R}^2$ that cannot be correctly classified by a linear model (not linearly separable)



Vapnik-Chervonenkis dimension

Non-linearly separable data

- the XOR function in $\mathbb{R}^2$ is not linearly separable (i.e. the perceptron algorithm cannot converge)

			y
$\mathbf{x}_1$	0	0	0
$\mathbf{x}_2$	0	1	1
$\mathbf{x}_3$	1	0	1
$\mathbf{x}_4$	1	1	0

- the probability that a set of 3 (non-aligned) points in $\mathbb{R}^2$is linearly separable is 1, but the probability that a set of four points is linearly separable is smaller than 1 (but not zero).
- when a class of hypotheses $\mathcal{H}$ can correctly classify all points of a training set $\mathcal{D}$, we say that $\mathcal{H}$ shatters $\mathcal{D}$.

Vapnik-Chervonenkis dimension

VC dimension of an hypothesis class

- The Vapnik-Chervonenkis dimension $VC_{dim}(\mathcal{H})$ of an hypothesis class $\mathcal{H}$ is defined as the maximal number of training examples that $\mathcal{H}$ can shatter
- we saw that in $\mathbb{R}^2$, this dimension is 3

$$VC_{dim}(Linear(\mathbb{R}^2)) = 3$$

  that can be generalized in $\mathbb{R}^d$

$$VC_{dim}(Linear(\mathbb{R}^d)) = d + 1$$

- given any set $(d + 1)$ examples in $\mathbb{R}^d$, there exists a linear classifier able to classify them perfectly
- for other types of non-linear hypotheses, the VC dimension is generally proportional to the number of free parameters

Structural risk minimization

Vapnik-Chervonenkis theorem

The generalization error of an hypothesis h taken from a class $\mathcal{H}$ of finite VC dimension and trained on N examples of $\mathcal{S}$ is bounded by

$$\epsilon(h) \leq \hat{\epsilon}_{\mathcal{S}}(h) + \sqrt{\frac{VC_{dim}(\mathcal{H}) \times (1 + \log(\frac{2N}{VC_{dim}(\mathcal{H})})) - \log(\frac{\delta}{4})}{N}}$$

with probability $1 - \delta$, if $VC_{dim}(\mathcal{H}) << N$

Structural risk minimization

$$\epsilon(h) \leq \hat{\epsilon}_{\mathcal{S}}(h) + \sqrt{\frac{VC_{dim}(\mathcal{H}) \times (1 + \log(\frac{2N}{VC_{dim}(\mathcal{H})})) - \log(\frac{\delta}{4})}{N}}$$

- The generalization error increases with the VC dimension, while the training error decreases.
- Structural risk minimization is an alternative method to cross-validation.
- The VC dimensions of various classes of hypothesis are already known. This bounds tells how many training samples are needed by a given hypothesis class in order to learn probably approximately correctly!
- A learning algorithm should only try to minimize the training error, as the VC complexity term only depends on the class.
- This term is only an upper bound: most of the time, the real bound is 100 times smaller.

Implication for linear classifiers

$$\epsilon(h) \leq \hat{\epsilon}_{\mathcal{S}}(h) + \sqrt{\frac{VC_{dim}(\mathcal{H}) \times (1 + \log(\frac{2N}{VC_{dim}(\mathcal{H})})) - \log(\frac{\delta}{4})}{N}}$$

- the VC dimension of linear classifiers in $\mathbb{R}^d$ is $d+1$
- for $N >> d$, the probability of having training errors becomes huge (the data is generally not linearly separable)
- If we project the input data onto a space with sufficiently high dimensions, it becomes then possible to find a linear classifier on this projection space that is probably approximately correct!
  This is the principle of multi-layer perceptron, radial-basis-function networks, support-vector machines...