Generalisation in theory and practice

Christos Dimitrakakis

September 29, 2025

Outline

Classification

The classifier as a decision rule

A decision rule $\pi(a|x)$ generates a decision $a \in [m]$. It is the conditional probability of a given x.

Classification

The classifier as a decision rule

A decision rule $\pi(a|x)$ generates a decision $a \in [m]$. It is the conditional probability of a given x.

Deterministic predictions given a model $P(y|x)$

Here, we pick the most likely class:

$$\pi(a|x_t) = \mathbb{I}\left\{a = \arg\max_y P(y|x_t)\right\}$$

Classification

The classifier as a decision rule

A decision rule $\pi(a|x)$ generates a decision $a \in [m]$. It is the conditional probability of a given x.

Deterministic predictions given a model $P(y|x)$

Here, we pick the most likely class:

$$\pi(a|x_t) = \mathbb{I}\left\{a = \arg\max_y P(y|x_t)\right\}$$

Randomised predictions given a model $P(y|x)$

Here, we randomly select a class according to our model:

$$\pi(a|x_t) = P(y_t = a|x_t)$$

Accuracy as a classification metric

Accuracy as a classification metric

The accuracy of a single decision

$$U(a_t, y_t) = \mathbb{I}\{a_t = y_t\} = \begin{cases} 1, & \text{if } a_t = y_t \\ 0, & \text{otherwise} \end{cases}$$

Accuracy as a classification metric

The accuracy of a single decision

$$U(a_t, y_t) = \mathbb{I}\{a_t = y_t\} = \begin{cases} 1, & \text{if } a_t = y_t \\ 0, & \text{otherwise} \end{cases}$$

The accuracy on a dataset

Let $D = \{(x_t, y_t) : t \in [T]\}$ be a dataset. We can measure the accuracy:

$$U(\pi, D) \triangleq \frac{1}{T} \sum_{t=1}^{T} \pi(y_t|x_t)$$

The expected accuracy of a decision rule

Accuracy as a classification metric

The accuracy of a single decision

$$U(a_t, y_t) = \mathbb{I}\{a_t = y_t\} = \begin{cases} 1, & \text{if } a_t = y_t \\ 0, & \text{otherwise} \end{cases}$$

The accuracy on a dataset

Let $D = \{(x_t, y_t) : t \in [T]\}$ be a dataset. We can measure the accuracy:

$$U(\pi, D) \triangleq \frac{1}{T} \sum_{t=1}^{T} \pi(y_t | x_t)$$

The expected accuracy of a decision rule

If $(x, y) \sim P$, the accuracy U of a rule π under P is the probability it predicts correctly

$$U(\pi, P) \triangleq \int_{\mathcal{X}} dP(x) \sum_{y=1}^{m} P(y|x) \pi(y|x)$$

Beyond classification: Generalised decision rules

Consider a spam application, where the e-mail client can decide between different action for emails. Different actions being best for each type of e-mail. The quality of each action can be captured through a utility function.

Utility of the spam decision problem

What utility function would you use for the spam detection problem?

Utility	Pass	Flag	Trash
Normal			
Spam			
Virus			

The utility function $U : \mathcal{Y} \times \mathcal{A} \to \mathbb{R}$

The utility function $U(y, a)$ is a real-valued function so that, for a label y, we prefer taking action a to a' iff $U(y, a) > U(y, a')$.

The optimal decision

- A model $P(y|x)$ of class probabilities
- A utility $U(y, a)$ for each class and action combination

Expected utility

We can calculate the expected utility of any decision

$$\mathbb{E}[U|a, x] = \sum_y P(y|x, a) U(y, a) = \sum_y P(y|x) U(y, a)$$

Here the first equality follows from the definition of conditional expectation and $P(y|x, a) = P(y|x)$ as the label does not depend on our actions.

The optimal decision

For any observation x, and P, we take the action maximising expected utility:

$$a^* = \arg\max_a \mathbb{E}_P[U|a, x]$$

This defines a function $\mathcal{X} \to \mathcal{A}$, which is the Bayes-optimal decision rule.

The optimal decision rule

- A model $P(y|x)$ of class probabilities
- A utility $U(y, a)$ for each class and action combination
- A decision rule $\pi(a|x)$ assigning probability to action a for every possible input x

Expected utility over a dataset.

We obtain the expected utility of the decision rule by marginalising over all actions

$$U(\pi, D) \triangleq \mathbb{E}[U|\pi, D] \overset{D=(x_t,y_t)_{t=1}^{T}}{=} \sum_{t=1}^{T} \mathbb{E}[U|\pi, x_t] = \sum_{t=1}^{T} \sum_{a \in \mathcal{A}} U(y_t, a)\pi(a|x_t)$$

Here the first equality follows from the definition of conditional expectation and $P(y|x, a) = P(y|x)$ as the label does not depend on our actions.

Expected utility over P

We can marginalise over possible datasets D

$$U(\pi, P) \triangleq \mathbb{E}_P[U|\pi] = \int_{\mathcal{D}} dP(D) U(\pi, D) \overset{\text{i.i.d.}}{=} \int_{\mathcal{X}} dP(x) \sum_{\mathcal{Y}} P(y|x) \sum_{a} \pi(a|x) U(y$$

Taking into account the probability

- For classification, it makes sense to look at the probability of the labels.
- If we are not very confident about our prediction, this should be taken into account:
- Define $P(y|x)$ to be our classifier's probability for label y, given features x. Then we can use two simple metrics:

Precision

The average probability of the actual class:

$$\sum_{t=1}^{T} P(y_t|x_t)/T$$

- If we always assign probability 1 to the correct label, this score is 1.
- If we always assign probability $1/m$ to all labels, the score is $1/m$.

Negative Log-Loss

Here we assign look at the logarithm of the probability. This really penalises bad guesses.

$$\sum_{t=1}^{T} \ln P(y_t|x_t)/T$$

- If we always assign probability 1 to the correct label, this score is 0.
- If we assign probability 0 to even a single label, the score is $-\infty$.

```
from sklearn.metrics import log_loss
```

in scikitlearn implements log-loss (not negative)

Regression

The regressor as a deterministic decision rule

A decision rule π generates a decision $a \in \mathbb{R}^m$.

- For deterministic rules $\pi(x)$ is the prediction for x.
- Since we can almost never guess correctly, we need to define the quality of our predictions somehow, either as a utility $U(y_t, a_t)$ or a loss function $\ell(y_t, a_t)$.

Mean-Squared Error Loss on a Dataset

This is the squared difference in predicted versus actual values:

$$\frac{1}{T} \sum_{t=1}^{T} [y_t - \pi(x_t)]^2$$

Expected MSE

If $(x, y) \sim P$, the expected MSE of a deterministic decision rule $\pi : \mathcal{X} \to \mathbb{R}$ is

$$\int_{\mathcal{X}} \int_{\mathcal{Y}} dP(x, y) [y - \pi(x)]^2.$$

Probabilistic regression

The regressor as a stochastic decision rule

A decision rule π generates a decision $a \in \mathbb{R}^m$.

- For stochastic rules $\pi(a|x)$ defines a density over predictions.
- In this case it is natural to define $\pi(y_t, x_t)$ as our metric.

Likelihood on a Dataset

The mean-square error is simply the squared difference in predicted versus actual values:

$$\prod t = 1^T \pi(y_t|x_t)$$

We will later see a link between this metric, mean-square error and estimation.

Training and overfitting

Training data

- $D = ((x_t, y_t) : t = 1, \ldots, T)$.
- $x_t \in \mathcal{X}$, $y_t \in \mathcal{Y}$.

Assumption: The data is generated i.i.d.

- $(x_t, y_t) \sim P$ for all t (identical)
- $D \sim P^T$ (independent)

The optimal decision rule for P

$$\max_\pi U(\pi, P) = \max_\pi \int_{\mathcal{X} \times \mathcal{Y}} dP(x, y) \sum_a \pi(a|x) U(a, y)$$

The optimal decision rule for D

$$\max_\pi U(\pi, D) = \max_\pi \sum_{(x,y) \in D} \sum_a \pi(a|x) U(a, y)$$

Generalisation

The fundamental problem

- We want to maximise $U(\pi, P)$
- We can only measure $U(\pi, D)$
- We have a learning algorithm $\lambda :\to \mathit{\Pi}$
- If $\pi = \lambda(D)$, then we instead measure $U(\lambda(D), D)$.

Generalisation

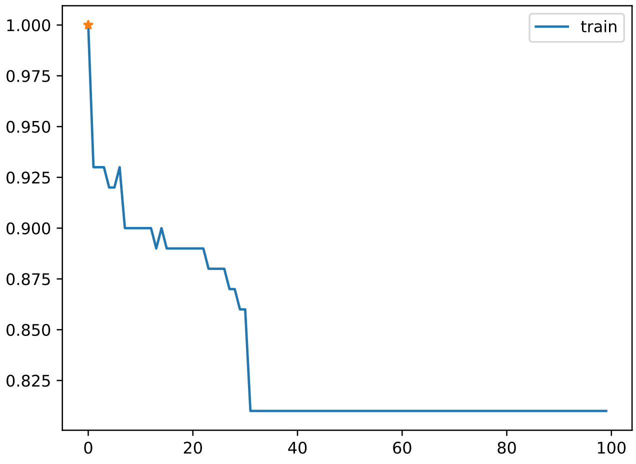
The fundamental problem

- We want to maximise $U(\pi, P)$
- We can only measure $U(\pi, D)$
- We have a learning algorithm $\lambda :\to \Pi$
- If $\pi = \lambda(D)$, then we instead measure $U(\lambda(D), D)$.

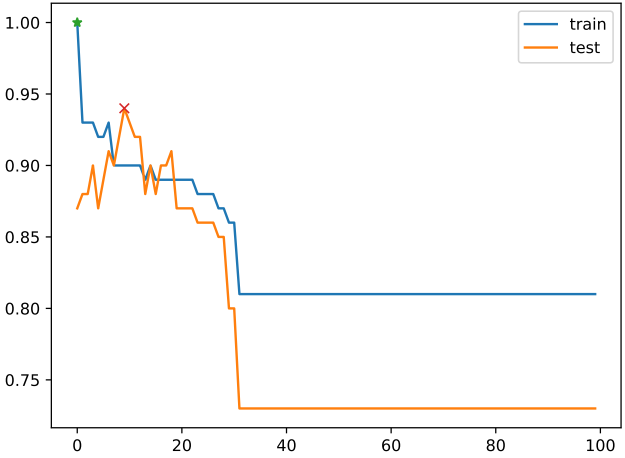
Training and testing

- Split D in $D_{\mathrm{train}}, D_{\mathrm{test}}$
- Obtain $\pi = \lambda(D_{\mathrm{train}})$
- Calculate $U(\pi, D_{\mathrm{test}})$
- $\mathbb{E}_P[U(\pi, D_{\mathrm{test}})] = U(\pi, P)$ for any fixed π (unbiased estimator)
- $\mathbb{E}_P[U(\lambda(D_{\mathrm{train}}), D_{\mathrm{train}})] \geq U(\pi, P)$ (biased estimator)

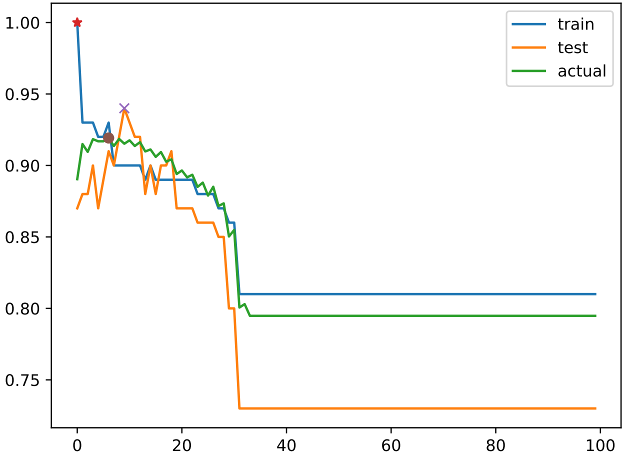
kNN Classifier Accuracy on a single dataset



kNN Classifier Accuracy on a single dataset

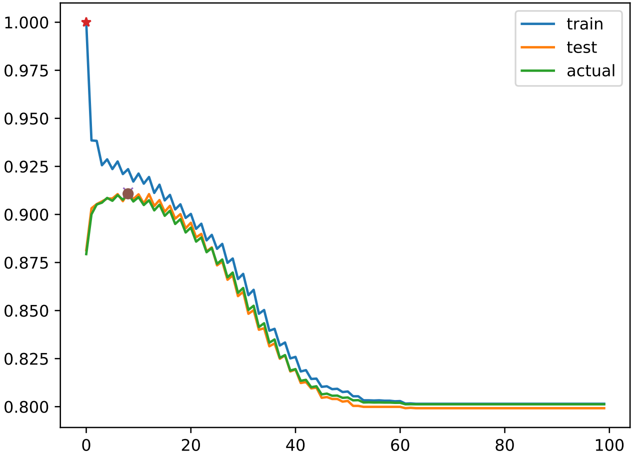


kNN Classifier Accuracy on a single dataset



Expected kNN Classifier Accuracy

Expectation approximated over 100 datasets D sampled from P.



The Train/Validation/Test methodology

Main idea
Use each piece of data once to make decisions and measure

Training set
Use to decide low-level model parameters

Validation set
Use to decide between:

- different hyperparameters (e.g. K in nearest neighbours)
- model (e.g. neural networks versus kNN)

Test set
Use to measure the final quality of a model

Cross-validation (XV)

Idea

- Use XV to select hyperparameters instead of a single train/valid test.

Methodology

- Split training set D in k different subsets
- At iteration i
- Use the i-th subset for validation
- Use all the remaining $k-1$ subsets for training
- Average results on validation sets

Cross-validation example: A mean estimate

Live coding in Python:

- Get a mean estimate
- Perform cross-validation

Hand-crafted

```
def xv_shuffler(x, estimator, scoring, n_folds):
        rng = np.random.default_rng
        # shuffle data
        T = len(x)
        indices = np.arange(T)
        shuffle(indices)
        fold_size = np.ceil(T / n_folds)
        fold_start = np.zeros(n_folds)
        fold_end = np.zeros(n_folds)
        # create folds
        for k in range(n_folds):
                fold_start = k * fold_size
                fold_end = np.min((k+1) * fold_size, T)
        # for each fold:
        # 1. Run estimator on k-1 folds
```

Bootstrapping

- Express uncertainty by resampling the data.
- Repeat your calculations for each resample

BootstrapSample(D)

```
**input** Data $D = (z_1, \ldots, z_T)$, of size $T$
for $t \in \{1, \ldots, T\}$ do
    Select $i$ uniformly in $[T]$
    Add the $i$-th point to $D_b$
end for
return $D_b$
```

BootstrapEstimate(D, λ, N)

```
input Data $D \in \mathcal{D}$, algorithm $\lambda : \mathcal{D} \to \Theta$, $N > 0$ number of samples
for $n \in \{1, \ldots, N\}$ do
    $\theta_n = \lambda(\mathrm{BootstrapSample}(D))$
end for
return $\{\theta_n : n \in [N]\}$
```

The wrong way to do XV for subset selection

1. Screen the predictors: find a subset of “good” predictors that show fairly strong (univariate) correlation with the class labels.
2. Using just this subset of predictors, build a multivariate classifier.
3. Use cross-validation to estimate the unknown tuning parameters and to estimate the prediction error of the final model.

Is this a correct application of cross-validation?

Consider a scenario with $N = 50$ samples in two equal-sized classes, and $p = 5000$ quantitative predictors (standard Gaussian) that are independent of the class labels. The true (test) error rate of any classifier is 50%.

The right way to do XV for feature selection

1. Divide the samples into K cross-validation folds (groups) at random.
2. For each fold $k = 1, 2, \ldots, K$
3. Find a subset of "good" predictors that show fairly strong (univariate) correlation with the class labels, using all of the samples except those in fold k.
4. Using just this subset of predictors, build a multivariate classifier, using all of the samples except those in fold k.
5. Use the classifier to predict the class labels for the samples in fold k.

Lab

- scikitlearn KNN
- scikitlearn performance measures
- train/test/validate plot with increasing k
- XV plot with increasing k
- Scaling and other preprocessing
- Effect of outliers on preprocessing
- Optional: Bootstrap performance evaluation

Learning and generalisation

How well can decision rule perform?

Estimation theory view

- Bias: The expected difference between the estimated value and the unknown parameter
- Variance: The expected difference between the estimated value and the unknown parameter

Learning theory view

- Approximation ability: How well a class of rules can approximate the optimal one.
- Statistical error: How easy it is to choose the best rule in the class.

Unbiased estimators

Definition (Estimator)

An estimator is a function $f : \mathcal{D} \to \Theta$, where Θ is a set of parameters. For any given dataset $D \in \mathcal{D}$, it returns a single estimate $\hat{\theta} = f(D)$.

Definition (Unbiased estimator)

An estimator is unbiased if, for the distribution $P(D|\theta)$, we have

$$\mathbb{E}[f \mid \theta] = \sum_{D} f(D) P(D|\theta) = \theta.$$

Example (Sample mean estimator)

Consider $D = (x_1, \ldots, x_T)$ with $x_t \sim P$ being i.i.d samples with $\mathbb{E}[x_t] = \theta$. The sample mean estimator $f(D) = \sum_t x_t / T$ is unbiased, as :

$$\mathbb{E}[f] = \mathbb{E}\left[\sum_{t=1} x_t / T\right] = \frac{1}{T} \sum_{t=1} \mathbb{E}[x_t] = \frac{1}{T} \sum_{t=1}^{T} \theta = \theta.$$

Example of a biased and unbiased estimator: Training error

- $U(\pi, D)$ is the measured accuracy of a classifier π on D
- $U(\pi, P) = \mathbb{E}_{D \sim P}[U(\pi, D)]$ is the actual accuracy. So $U(\pi, D)$ is unbiased.
- $\lambda(D) = \arg\max_{\pi} U(\pi, D)$ is a learning algorithm picking the best classifier for a dataset D.
- Then $U(\lambda(D), D)$ is biased, as for any π'

$$\begin{aligned}
\mathbb{E}_{D \sim P}[U(\lambda(D), D)] &= \int dP(D) U(\lambda(D), D) && (1)\\
&= \int dP(D) \max_{\pi} U(\pi, D) && (2)\\
&\geq \int dP(D) U(\pi', D) && (3)\\
&= U(\pi', P) && (4)
\end{aligned}$$

i.e. the expected value of the training accuracy is higher than the accuracy of any classifier.

The bias/variance trade-off

- Dataset $D \sim P$.
- Predictor $f_D(x)$
- Target function $y = f(x) + \epsilon$
- $\mathbb{E}\,\epsilon = 0$ zero-mean noise with variance $\sigma^2 = \mathbb{V}(\epsilon)$

MSE decomposition

$$\mathbb{E}[(f - f_D)^2] = \mathbb{V}(f_D) + \mathbb{B}(f_D)^2 + \sigma^2$$

Variance
How sensitive the estimator is to the data

$$\mathbb{V}(f_D) = \mathbb{E}[(f_D - \mathbb{E}(f_D))^2]$$

Bias
What is the expected deviation from the true function

$$\mathbb{B}(f_D) \triangleq \mathbb{E}[(f_D - f)]$$

Example: mean estimation

- Data $D = y_1, \ldots, y_T$ with $\mathbb{E}[y_t] = \mu$.
- Goal: estimate μ with some estimator f_D to minimise
- MSE: $\mathbb{E}[(y - f_D)^2]$, the expected square difference between new samples our guess.

Optimal estimate

To minimise the MSE, we use $f^* = \mu$. This gives us two ideas:

Empirical mean estimator:

- $f_D = \sum_{t=1}^{T} x_t / T$.
- $\mathbb{V}(f_D) = \mathbb{E}[f_D - \mu] = 1/\sqrt{T}$
- $\mathbb{B}(f_D) = 0$. (unbiased estimator)

Laplace mean estimator:

- $f_D = \sum_{t=1}^{T} (\lambda + x_t) / T$.
- $\mathbb{V}(f_D) = \mathbb{E}[f_D - \mu] = \frac{1}{1+\sqrt{T}}$
- $\mathbb{B}(f_D) = O(1/T)$.

A proof of the bias/variance trade-off

- RV's $y_t \sim P$, $\mathbb{E}[y_t] = \mu$, $y_t = \mu + \epsilon_t$.
- Estimator f_D, $D = y_1, \ldots, y_{t-1}$.

$$\begin{aligned}
\mathbb{E}[(f_D - y_t)^2] &= \mathbb{E}[f_D^2] - 2\,\mathbb{E}[f_D y_t] + \mathbb{E}[y_t^2] \\
&= \mathbb{V}[f_D] + \mathbb{E}[f_D]^2 - 2\,\mathbb{E}[f_D y_t] + \mathbb{E}[y_t^2] \\
&= \mathbb{V}[f_D] + \mathbb{E}[f_D]^2 - 2\,\mathbb{E}[f_D]\,\mathbb{E}[y_t] + \mathbb{E}[y_t^2] \\
&= \mathbb{V}[f_D] + \mathbb{E}[f_D]^2 - 2\,\mathbb{E}[f_D]\mu + \mathbb{E}[y_t^2] \\
&= \mathbb{V}[f_D] + \mathbb{E}[f_D]^2 - 2\,\mathbb{E}[f_D]\mu + \mathbb{E}[(\mu + \epsilon_t)^2] \\
&= \mathbb{V}[f_D] + \mathbb{E}[f_D]^2 - 2\,\mathbb{E}[f_D]\mu + \mathbb{E}[\mu^2 + 2\mu\epsilon_t + \epsilon_t^2] \\
&= \mathbb{V}[f_D] + \mathbb{E}[f_D]^2 - 2\,\mathbb{E}[f_D]\mu + \mu^2 + \sigma^2 \\
&= \mathbb{V}[f_D] + (\mathbb{E}[f_D] - \mu)^2 + \sigma^2 \\
&= \mathbb{V}(f_D) + \mathbb{B}(f_D)^2 + \sigma^2
\end{aligned}$$

Generalisation error

Regret decomposition

Let the optimal rule be $\pi^* \in \Pi$, the best approximate rule be $\hat{\pi}^* \in \Pi$ and our rule be $\hat{\pi} \in \hat{\Pi}$. We call the difference between the performance of π^* and $\hat{\pi}$ our regret:

$$\underbrace{U(\pi^*, P) - U(\hat{\pi}, P)}_{\text{regret}} = \underbrace{U(\pi^*, P) - U(\hat{\pi}^*, P)}_{\text{approximation error}} + \underbrace{U(\hat{\pi}^*, P) - U(\hat{\pi}, P)}_{\text{estimation error}}$$

We can bound the regret by bounding each term separately.

- The approximation error tells us how expressive our class of rules is, i.e. how much we lose by looking at a restricted class $\hat{\Pi}$ of rules. It is similar to estimator bias.
- The statistical error tells us how well the empirical performance on D approximates the true performance. It is similar to estimator variance.
- As a rule of thumb, the larger our class, the better the possible approximation but the higher the statistical error.

Approximation error

- Our model limits us to a set of decision rules $\hat{\Pi} \subset \Pi$.
- The most we could do is find the best rule in $\hat{\Pi}$.
- This still leaves a gap:

$$\Delta \triangleq \max_{\pi \in \Pi} U(\pi, P) - \max_{\hat{\pi} \in \hat{\Pi}} U(\pi, P)$$

The gap can be characterised in some cases.

Example: ϵ-net on Lipschitz $U(\cdot, P)$.

- Assume $U(\pi, P)$ is a Lipschitz function of π for all P, i.e. $|U(\pi, P) - U(\pi', P)| \leq L d(\pi, \pi')$ for some metric d.
- Let $\hat{\Pi}$ be an ϵ-net on Π, i.e. $\max_{\pi \in \Pi} \min_{\pi' \in \hat{\Pi}} d(\pi, \pi') = \epsilon$.
- Then $\Delta \leq L\epsilon$.

Estimation error

- First, let us bound $U(\hat{\pi}^*, P) - U(\hat{\pi}, P)$ by making an assumption.
- Then, we can prove that our assumption holds with high probability.

Lemma

Let $f, g : S \to \mathbb{R}$. If $\|f - g\|_\infty \leq \epsilon$ and $f(x) \geq f(z)$, while $g(y) \geq g(z)$, for all z, i.e. x, y maximise f, g respectively

$$f(x) - f(y) \leq 2\epsilon.$$

This holds as: $f(x) - f(y) \leq g(x) + \epsilon - f(y) \leq g(y) + \epsilon - f(y) \leq 2\epsilon$.

Corollary

If $|U(\pi, P) - U(\pi, D)| \leq \epsilon$ for all π then

$$U(\hat{\pi}^*, P) - U(\hat{\pi}, P) \leq 2\epsilon$$

- Let us now prove that, with high probability, $|U(\pi, P) - U(\pi, D)| \leq \epsilon$.

Bounding the estimation error

For any fixed rule $\pi \in \Pi$ and utility function $U : \Pi \times \mathcal{X}^T \to [0,1]$,

$$P^T(|U(\pi, D) - U(\pi, P)| \geq \epsilon) \leq 2\exp(-2T\epsilon^2).$$

This is a direct application of Hoeffding's inequality[1]. Taking the union bound over the set $\hat{\Pi}$ gives:

$$P^T(\exists \pi \in \hat{\Pi} : |U(\pi, D) - U(\pi, P)| \geq \epsilon) \leq 2|\hat{\Pi}|\exp(-2T\epsilon^2).$$

Setting the right side equal to δ and re-arranging,

$$P^T\left(\max_{\pi \in \hat{\Pi}} |U(\pi, D) - U(\pi, P)| \geq \sqrt{\frac{\ln(2|\hat{\Pi}|/\delta)}{2T}}\right) \leq \delta.$$

Example: ϵ-net.

In a n dimensional space we require $|\hat{\Pi}| = O(\epsilon^{-n})$. This means that our statistical error is $O(\sqrt{n \ln(1/\epsilon\delta)/T})$.

[1]See Hoeffding's inequality in the confidence intervals presentation

The finite hypothesis algorithm

- Input: a finite set of rules $\hat{\Pi}$, data D, utility U
- Return $\hat{\pi} \in \arg\max_{\pi \in \hat{\Pi}} U(\pi, D)$.

Regret of the finite hypothesis algorithm.

With probability $1 - \delta$

$$U(\hat{\pi}, P) \geq U(\hat{\pi}^*, P) - \sqrt{2 \ln(2|\hat{\Pi}|/\delta)/T} \tag{5}$$

$$U(\pi^*, P) - U(\hat{\pi}, P) \leq \Delta + \sqrt{2 \ln(2|\hat{\Pi}|/\delta)/T} \tag{6}$$

Examples

- ML estimation: $U(\beta, D) = P_\beta(D)$ is the data likelihood.
- Accuracy, etc: $U(\pi, D)$.

VC Dimension

Here we consider sets Π of deterministic rules $\pi : \mathcal{X} \to \{0, 1\}$.

Shattering

If a $S \subset \mathcal{X}$ can with $|S| = m$, can be assigned any labelling $y_1, \ldots, y_m$ by a $\pi \in \Pi$, then we say Π shatters S.

The VC dimension

This is the largest-size set S that Π can shatter.

Example: Perceptrons on $\mathbb{R}^2$

This class has VC dimension 3 on the plane.

Binary classification

Learning algorithm λ

- Takes data $D = \{(x_t, y_t)\}$ as input
- Generates deterministic decision rules $\pi : X \to \{0, 1\}$,

The loss of a rule π.

- Assume an existing concept class $\pi^* \in \mathit{\Pi}$
- Distribution $x_t \sim P$ is i.i.d. and $x_1, \ldots, x_T \sim P^T$.
- The loss under distribution P is

$$L(\pi) = P(\{x : \pi(x) \neq \pi^*(x)\})$$

Realisable PAC learner

- $\lambda : (\mathcal{X} \times \mathcal{Y})^* \to \mathit{\Pi}$ is (ϵ, δ)-PAC, if for any P and $\epsilon, \delta > 0$, and any concept $\pi^* \in \mathit{\Pi}$, there is T such that

$$P^T(\{D : L[\lambda(D)] > \epsilon\}) < \delta, \qquad D = (\{x_t, \pi^*(x_t)\}), x_t \sim P.$$