Machine Learning B (2025)
Home Assignment 5

Carsten Jørgensen, student ID: skj730

Contents

2 PAC-Bayesian Aggregation

The main task of exercise 3.8 from the Lecture Notes [Sel25] is to reproduce Figure 2(a) from [Thi+17].

I am using the SVM algorithm from [Ped+11] for the classification task. This algorithm is a wrapper around LIBSVM [CL11], which is being used in [Thi+17].

Here is my version of Figure 2(a) 1.

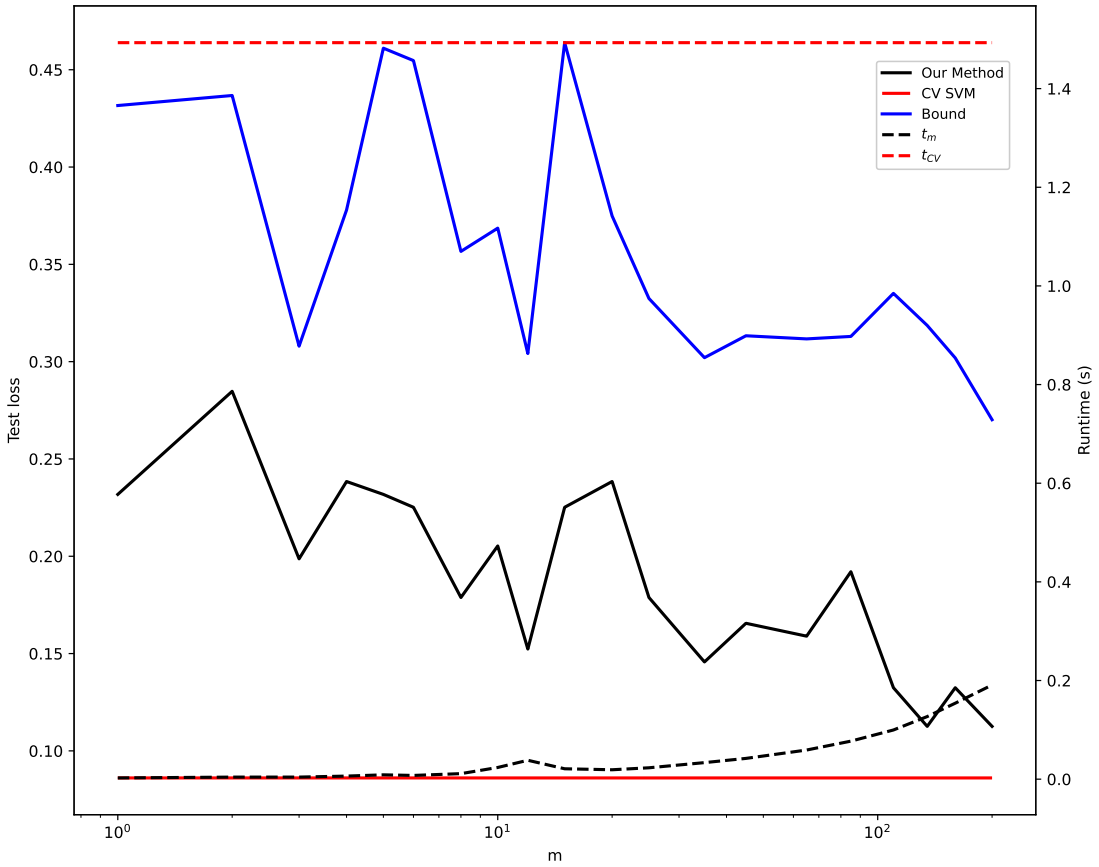

Figure 1: Comparison of PAC-Bayesian aggregation with RBF kernel SVM tuned by cross-validation.

Data Preparation

The data set used for Figure 2(a) is the *Ionosphere* data set from [Sig+89]. The original data has class labels b and g remapped to $-1, 1$. Label b is bad and g is good.

The data is then split into a training set S and test set T with $|S| = 200, |T| = 150$ as mentioned in Table 1 in [Thi+17][1]. This split was creating by scikit-learn

[1]The data set actually contains 351 rows so I am just ignoring a random point to match size of training and test sets given by [Thi+17].

method `train_test_split` with a stratify approach to ensure balance of labels.

All features have been standardized. I cannot find any where in [Thi+17] mentioning this, but standardization of features is recommended in Section 2.2 of the LIBSVM tutorial [HCL03].

Baseline cross-validated SVM

Paragraph 2 of Section 6 in [Thi+17] explains how the Baseline cross-validated SVM is constructed using a grid search with parameters C and γ.

The γ-grid is based on the Jaakkola heuristic from [JDH99] and the calculation of the γ value is basically the median distance between pairs of points with opposite labels. My implementation of the heuristic is a wrapper around the `cdist` function from [Ped+11].

Using `GridSearchCV` the best parameters across searched parameters are

$$C = 10.0, \quad \gamma = 0.0071686$$

yielding a model score of 0.945 and one-zero loss of 0.086092

Alternating minimization of the PAC-Bayes-λ bound

This is the core part of being able to reproduce Figure 2(a). The full code listing for the Python implementing the alternating minimization is listed in Section 3. Here we shall only include the most essential parts.

The input to the algorithm are the

Validation losses (Eq. (13)). For every weak learner h we need the empirical loss $\widehat{L}_{\text{val}}(h, S)$ on *validation* points $S \setminus S_h$. This is done inside the training loop, see Section 3 for the `L_val` variable.

Numerically stability of the update rule for ρ in Eq.(7). The exercise hints a way to ensure a numerical stable calculation of the Eq.(7). Here I am using the log-sum-exp trick as explained in [Gun99] and if I understand the hint correctly, then I believe that it is more or less equivalent to the log-sum-exp trick. This is implemented as:

```
# Eq. (7): computed in log-domain for numerical stability
log_pi = -np.log(m) * np.ones(m)  # uniform prior  \pi(h)=1/m (
   from page 10)
log_rho = log_pi - lambda_ * n_r * L_val
log_rho -= logsumexp(log_rho)
rho = np.exp(log_rho)
```

KL divergence and empirical loss. To compute the KL term, I am using the first equality from Eq. (9):

$$KL(\rho\|\pi) = \mathop{\mathbb{E}}_{h\sim\rho}\left[\ln \frac{\rho(h)}{\pi(h)}\right] = \sum_{h\in\mathcal{H}} \rho(h) \ln \frac{\rho(h)}{\pi(h)}.$$

to keep the calculation numerical stable in the log-domain.

```
# helpers for Eq. (8) & bound. KL divergence is computed
# in log-domain using the first equality in Eq. (9)
KL = np.sum(rho * (log_rho - log_pi))
L_emp = np.dot(rho, L_val)
```

Closed-form update of λ from Eq.(8). This is the code for updating λ:

```
# Eq. (8): lambda_new
num = 2.0
denom = (
    np.sqrt(2.0 * n_r * L_emp / (KL + np.log(2 * np.sqrt(n_r) /
   delta)))
    + 1.0
    + 1.0
)
lambda_new = num / denom
```

from Eq.(8):

$$\lambda^{\text{new}} = \frac{2}{\sqrt{\frac{2(n-r)\mathbb{E}_\rho[\widehat{L}(h,S)]}{\mathrm{KL}(\rho||\pi) + \ln\frac{2\sqrt{n-r}}{\delta}}} + 1 + 1}.$$

Upper Bound value (right-hand side of Eq.(13)).

$$F(\rho,\lambda) = \frac{E_\rho[\widehat{L}]}{1-\frac{\lambda}{2}} + \frac{\mathrm{KL}(\rho||\pi) + \ln\frac{2\sqrt{n-r}}{\delta}}{\lambda(1-\frac{\lambda}{2})(n-r)}.$$

```
# Eq. (13): value of PAC-Bayes-\lambda bound
bound = L_emp / (1 - lambda_ / 2.0) + (KL + np.log(2 * np.sqrt(n_r
   ) / delta)) / (
    n_r * lambda_ * (1 - lambda_ / 2.0)
)
```

Construction of Hypothesis Space

For each m[2] on the x-axis of Figure 2(a) we create m weak-learners SVM and compute the m empirical losses.

[2]The paper does not specify the m-value but I have eye-balled them to be $1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 20, 25, 35, 45, 65, 85, 110, 135, 160, 200,$

Majority vote evaluation

For the majority vote I use Eq. 3.31 in [Sel25]: $\mathrm{MV}_\rho(X) = \mathrm{sign}\left(\sum_{h\in\mathcal{H}} \rho(h)h(X)\right)$ implemented in the last lines of code in the training algorithm 2.

Conclusion

The main difference that I see between Figure 2(a) from [Thi+17] and my reproduction seems to be the running time for the baseline cross-validated SVM. I am using the scikit-learn SVM which is a wrapper around LIBSVM used in the paper. Maybe that can explain some of the performance loss.

In Figure 2 we see an updated version of Figure 2(a) where the experiment was repeated 10 times such that an 95% confidence level could be computed for the bounds.

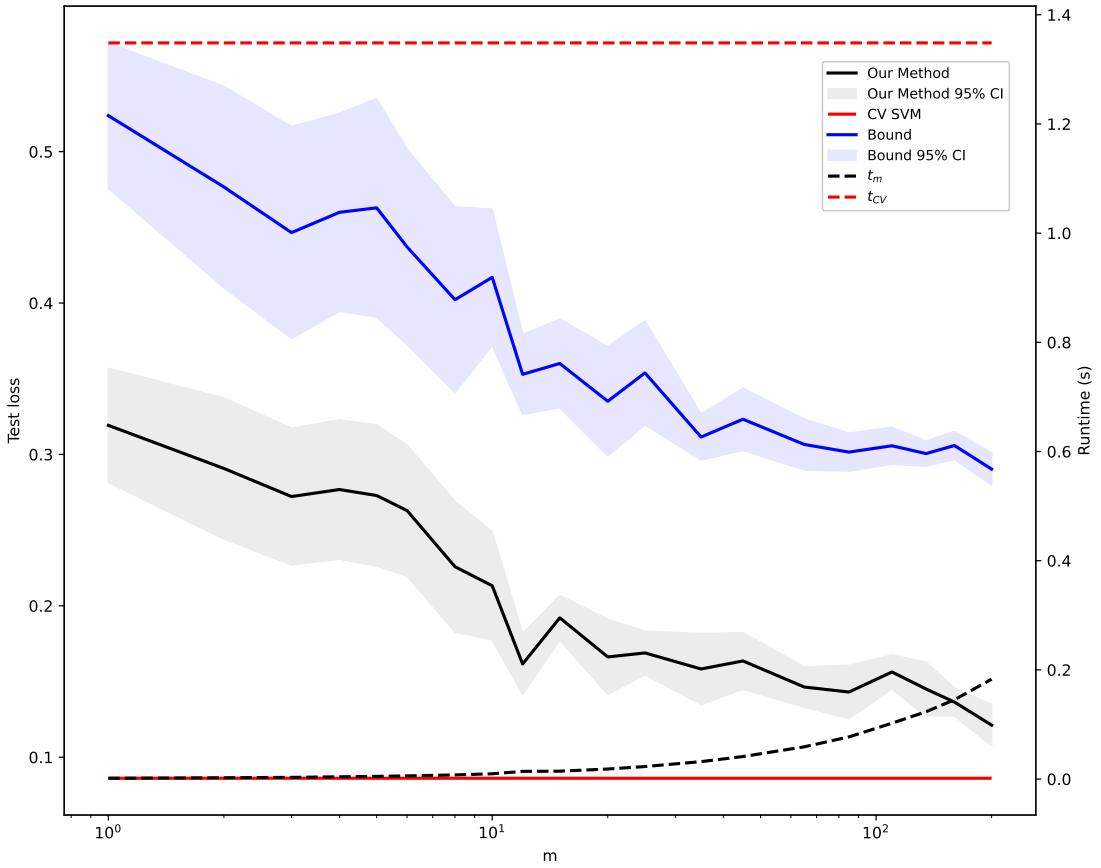

Figure 2: Figure 2(a) with 95% Confidende intervals (10 runs). Note running times are plotted for a single run.

Table 1 summarizes the statistics for 10 runs of the experiment. The details of the baseline cross-validated SVM model are: cross-validation score: 0.9450 and best parameters: $C = 10.0$, $\gamma = 0.007065$.

Method	Loss ($\mu \pm \sigma$)	Runtime ($\mu \pm \sigma$) [s]
Our Method	0.1212 ± 0.0222	0.1904 ± 0.0069
PAC-Bayes Bound	0.2903 ± 0.0172	-
CV SVM	0.0861	1.2932

Table 1: Statistics for 10 repeated experiment

3 VC Dimension

Q1 of Exercise 3.4

Finite hypothesis class Let $\mathcal{H}$ be finite, $|\mathcal{H}| = M$. If a set of n points is shattered by $\mathcal{H}$, then

$$2^n \leq |\mathcal{H}| = M.$$

Hence $n \leq \lfloor \log_2 M \rfloor$ and therefore

$$\boxed{d_{\mathrm{VC}}(\mathcal{H}) \ \leq \ \lfloor \log_2 M \rfloor}.$$

Q2 of Exercise 3.4

Two-element hypothesis class Assume $\mathcal{H} = \{h_1, h_2\}$.

- One point can be shattered: choose x, label it by $h_1(x)$ or $h_2(x)$.
- Two points cannot be shattered: on $\{x_1, x_2\}$ the two hypotheses give at most two different labelings, but four are required.

Hence

$$\boxed{d_{\mathrm{VC}}(\mathcal{H}) = 1}.$$

Q3 of Exercise 3.4

Positive circles in $\mathbb{R}^2$ Let $\mathcal{H}_+ = \{\, h_{c,r} : x \mapsto \mathbb{1}\{\|x - c\| \leq r\}\,\}$ ("positive" discs).
Take three non–collinear points A, B, C.

label $(- - -)$: disc of radius 0, no point covered;
$(+ - -)$: small disc around A;
$(+ + -)$: disc through A, B with centre on their bisector, radius small enough to exclude C
$(+ + +)$: sufficiently large disc containing all three.

The remaining $2^3 - 4$ patterns follow by symmetry. Hence the set $\{A, B, C\}$ is shattered and

$$\boxed{d_{\mathrm{VC}}(\mathcal{H}_+) \geq 3}.$$

Q4 of Exercise 3.4

Positive & negative circles Let $\mathcal{H} = \mathcal{H}_+ \cup \mathcal{H}_-$, where $\mathcal{H}_-$ contains *negative* discs ($h_{c,r}(x) = -1$ iff $\|x - c\| \leq r$).
Choose four points that are vertices of a convex quadrilateral $S = \{P_1, P_2, P_3, P_4\}$. For any labelling of S let S_+ be the $+1$ points, $S_- = S \setminus S_+$. Because $|S| = 4$ we have $\min\{|S_+|, |S_-|\} \leq 2$.

- If $|S_+| \leq 2$ choose a *positive* disc containing exactly S_+.
- If $|S_-| \leq 2$ choose a *negative* disc containing exactly S_-.

Thus every dichotomy of S is realizable; S is shattered and

$$\boxed{d_{\mathrm{VC}}(\mathcal{H}) \geq 4}.$$

(The matching upper bounds $d_{\mathrm{VC}}(\mathcal{H}_+) \leq 3$ and $d_{\mathrm{VC}}(\mathcal{H}) \leq 4$ are optional and not shown here.)

Q5 - Exercise 3.6

Construction:

Hypothesis Space $\mathcal{H}$:

$$\mathcal{H} = \{h : \mathbb{R} \to \{0, 1\} \mid h \text{ is any function}\}$$

This clearly has infinite VC-dimension since we can shatter any finite set of real numbers by defining functions that assign arbitrary binary labels to those points: $d_{\mathrm{VC}}(\mathcal{H}) = \infty$.
Data Distribution $p(X, Y)$:

$$p(X = 0, Y = 0) = 1 \tag{1}$$
$$p(X = x, Y = y) = 0 \quad \text{for all other } (x, y) \text{ pairs} \tag{2}$$

In other words, every data point is identical: $(0, 0)$.

Analysis:

With this construction:

1. **For any sample S of $n > 100$ points:** $S = \{(0, 0), (0, 0), \ldots, (0, 0)\}$ (n copies)
2. **For any hypothesis $h \in \mathcal{H}$:**
   - Empirical loss: $\hat{L}(h, S) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(0), 0) = \ell(h(0), 0)$
   - Expected loss: $L(h) = \mathbb{E}[\ell(h(X), Y)] = \ell(h(0), 0)$

3. **Therefore:** $L(h) = \hat{L}(h, S)$ for all $h \in \mathcal{H}$

4. **Generalization gap:**

$$L(h) - \hat{L}(h, S) = 0 \leq 0.01 \quad \text{for all } h \in \mathcal{H}$$

5. **Probability:** This holds with probability 1 (deterministically), which satisfies:

$$\mathbb{P}\left(\sup_{h \in \mathcal{H}} |L(h) - \hat{L}(h, S)| \leq 0.01\right) = 1 \geq 0.95$$

Key Insight:

This example demonstrates that **infinite VC-dimension doesn't automatically imply poor generalization**. VC theory provides worst-case bounds—it shows there *exists* a distribution that leads to overfitting when $d_{\text{VC}}(\mathcal{H}) = \infty$, but it doesn't claim that *all* distributions are problematic.
In our construction, the data distribution is so concentrated (on a single point) that there's no opportunity for overfitting. The empirical and expected losses coincide perfectly:

$$\hat{L}(h, S) = L(h) \quad \forall h \in \mathcal{H}$$

because they both depend only on the hypothesis's behavior at the one point that actually occurs in the data distribution.

o3. 1. Sample space and distribution. Let the input space be the real line $\mathcal{X} = \mathbb{R}$ and fix a single pair $(x_0, y_0) = (0, 1) \in \mathcal{X} \times \{0, 1\}$. Define the data–generating distribution p by

$$\Pr\big[(X, Y) = (x_0, y_0)\big] = 1.$$

Hence every sample point is *deterministically* equal to (x_0, y_0).

2. Hypothesis class with infinite VC–dimension. Let

$$\mathcal{H} \;=\; \big\{\, h_A \;\big|\; A \subseteq \mathbb{R},\; h_A(x) = \mathbb{1}\{x \in A\}\big\}.$$

For any finite set $\{x_1, \ldots, x_n\} \subset \mathbb{R}$ every labelling $(\ell_1, \ldots, \ell_n) \in \{0, 1\}^n$ is realized by the choice $A = \{x_i : \ell_i = 1\}$, so $VCdim(\mathcal{H}) = \infty$.

3. Empirical and true errors coincide. Let $S = \{(X_i, Y_i)\}_{i=1}^n$ be an i.i.d. sample of size $n > 100$ drawn from p. Necessarily

$$S = \big\{(x_0, y_0), \ldots, (x_0, y_0)\big\}.$$

For any $h \in \mathcal{H}$ put

$$L(h) = \Pr\big[h(X) \neq Y\big] = \mathbb{1}\{h(x_0) \neq y_0\}, \qquad \hat{L}(h, S) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{h(X_i) \neq Y_i\} = \mathbb{1}\{h(x_0) \neq y_0\}.$$

Hence $L(h) = \hat{L}(h, S)$ for *every* $h \in \mathcal{H}$ and every such sample S; the generalization gap is identically 0.

4. Required probability statement. Because the gap is zero, for any $n > 100$

$$\Pr\Big[\forall h \in \mathcal{H} : \; L(h) \le \hat{L}(h, S) + 0.01\Big] \; = \; 1 \; \ge \; 0.95.$$

Thus we have exhibited a hypothesis space of infinite VC–dimension together with a data distribution for which, with probability 1, the empirical errors of *all* hypotheses equal their true errors (so certainly within 0.01). Infinite VC–dimension therefore *does not* inevitably lead to overfitting; the phenomenon depends on the data distribution.

Appendix A

Alternating minimization algorithm

```
def minimize_bound(L_val):
    """
    # Perform alternating minimisation of F(rho,lambda)$
    # (RHS of Eq. (13)).
    * L_val : vector (m,) of validation losses

    Returns rho, lambda^*, bound value.
    """
    m = len(L_val)
    log_pi = -np.log(m) * np.ones(m)  # uniform prior  \pi(h)=1/m
   (from page 10)
    lambda_ = 1.0

    for i in range(50):
        # Eq. (7): computed in log-domain for numerical stability
        log_rho = log_pi - lambda_ * n_r * L_val
        log_rho -= logsumexp(log_rho)
        rho = np.exp(log_rho)

        # helpers for Eq. (8) & bound. KL divergence is computed
        # in log-domain using the first equality in Eq. (9)
        KL = np.sum(rho * (log_rho - log_pi))
        L_emp = np.dot(rho, L_val)

        # Eq. (8): lambda_new
        num = 2.0
        denom = (
            np.sqrt(2.0 * n_r * L_emp / (KL + np.log(2 * np.sqrt(
   n_r) / delta)))
            + 1.0
            + 1.0
        )
        lambda_new = num / denom
```

```
        if abs(lambda_new - lambda_) < 1e-5:  # conv. criterion
            lambda_ = lambda_new
            break
        lambda_ = lambda_new

    # Eq. (13): value of PAC-Bayes-\lambda bound (randomized
    classifier)
    bound = L_emp / (1 - lambda_ / 2.0) + (KL + np.log(2 * np.sqrt
    (n_r) / delta)) / (
        n_r * lambda_ * (1 - lambda_ / 2.0)
    )

    return rho, lambda_, bound
```

Listing 1: Alternating minimization

Training algorithm

```
# Build ensemble of m weak SVMs  (Sec. 5 construction
# of Hypothesis Space H)
for i in range(m):
    # Select r = d + 1 random indices for training
    # and the rest for validation
    train_idx = rng.choice(n, r, replace=False)
    val_idx = np.setdiff1d(all_indices, train_idx, assume_unique=
    True)

    X_tr, y_tr = X_S[train_idx], y_S[train_idx]
    X_val, y_val = X_S[val_idx], y_S[val_idx]

    # Select random gamma from grid - p. 10 in paper
    gamma = rng.choice(gamma_grid)
    clf = SVC(kernel="rbf", C=1.0, gamma=gamma).fit(X_tr, y_tr)

    L_val[i] = zero_one_loss(clf.predict(X_val), y_val)
    predsT[i] = clf.predict(X_T)

rho, _, bound = minimize_bound(L_val)

# rho-weighted Majority vote of Eq. (3.31)
# in Machine Learning: The science of selection under uncertainty
mv_raw = rho @ predsT
mv_pred = np.sign(mv_raw)  # +/- 1  (np.sign(0) returns 0 - no tie
    here)
```

Listing 2: Training algorithm

References

[Sig+89] V. Sigillito et al. *Ionosphere*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5W01B. 1989.

[Gun99] Gregory Gundersen. *The Log-Sum-Exp Trick*. 1999. URL: https://gregorygundersen.com/blog/2020/02/09/log-sum-exp/ (visited on 05/23/2025).

[JDH99] Tommi Jaakkola, Mark Diekhans, and David Haussler. "Using the Fisher Kernel Method to Detect Remote Protein Homologies". In: *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, 1999, pp. 149–158. ISBN: 1577350839.

[HCL03] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. *A Practical Guide to Support Vector Classification*. Tech. rep. Department of Computer Science, National Taiwan University, 2003. URL: http://www.csie.ntu.edu.tw/~cjlin/papers.html.

[CL11] Chih-Chung Chang and Chih-Jen Lin. "LIBSVM: A library for support vector machines". In: *ACM Transactions on Intelligent Systems and Technology* 2 (3 2011). Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm, 27:1–27:27.

[Ped+11] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[Thi+17] Niklas Thiemann et al. *A Strongly Quasiconvex PAC-Bayesian Bound*. 2017. arXiv: 1608.05610 [cs.LG]. URL: https://arxiv.org/abs/1608.05610.

[Sel25] Yevgeny Seldin. *Machine Learning The Science of Selection under Uncertainty*. Apr. 22, 2025. URL: https://sites.google.com/site/yevgenyseldin/teaching.