Machine Learning 2

B. Ensembles / B.2. Boosting

Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)
Institute for Computer Science
University of Hildesheim, Germany

Syllabus

			A. Advanced Supervised Learning
Fri.	24.4.	(1)	A.1 Generalized Linear Models
Fri.	1.5.	—	— *Labour Day* —
Fri.	8.5.	(2)	A.2 Gaussian Processes
Fri.	15.5.	(3)	A.3 Advanced Support Vector Machines
			B. Ensembles
Fri.	22.5.	(4)	B.1 Stacking
			& B.2 Boosting
Fri.	29.5.	(5)	B.3 Mixtures of Experts
Fri.	5.6.	—	— *Pentecoste Break* —
			C. Sparse Models
Fri.	12.6.	(6)	C.1 Homotopy and Least Angle Regression
Fri.	19.6.	(7)	C.2 Proximal Gradients
Fri.	26.6.	(8)	C.3 Laplace Priors
Fri.	3.7.	(9)	C.4 Automatic Relevance Determination
			D. Complex Predictors
Fri.	10.7.	(10)	D.1 Latent Dirichlet Allocation (LDA)
Fri.	17.7.	(11)	Q & A

Outline

1. Idea & L2 Loss Boosting
2. Exponential Loss Boosting (AdaBoost)
3. Functional Gradient Descent (Gradient Boosting)

Outline

Consecutive vs Joint Ensemble Learning

So far, ensembles have been constructed in two **consecutive steps**:

- 1st step: create heterogeneous models
  - learn model parameters for each model separately
- 2nd step: combine them
  - learn combination weights (stacking)

Consecutive vs Joint Ensemble Learning

So far, ensembles have been constructed in two **consecutive steps**:

- 1st step: create heterogeneous models
  - learn model parameters for each model separately
- 2nd step: combine them
  - learn combination weights (stacking)

Advantages:

- simple
- trivial to parallelize

Disadvantages:

- models are learnt in isolation

Consecutive vs Joint Ensemble Learning

So far, ensembles have been constructed in two **consecutive steps**:

- 1st step: create heterogeneous models
  - learn model parameters for each model separately
- 2nd step: combine them
  - learn combination weights (stacking)

Advantages:

- simple
- trivial to parallelize

Disadvantages:

- models are learnt in isolation

New idea: **Learn model parameters and combination weights jointly**

$$\ell(\mathcal{D}^{\text{train}}; \Theta) := \sum_{n=1}^{N} \ell(y_n, \sum_{c=1}^{C} \alpha_c \hat{y}(x_n; \theta_c)), \quad \Theta := (\alpha, \theta_1, \dots, \theta_C)$$

Boosting

Idea: fit models (and their combination weights)

- sequentially, one at a time,
- relative to the ones already fitted,
- but do not consider to change the earlier ones again.

Boosting

Idea: fit models (and their combination weights)

- sequentially, one at a time,
- relative to the ones already fitted,
- but do not consider to change the earlier ones again.

$$\begin{aligned}\hat{y}^{(C')}(x) :=& \sum_{c=1}^{C'} \alpha_c \hat{y}(x;\theta_c), \quad C' \in \{1,\ldots,C\}\\ =& \hat{y}^{(C'-1)}(x) + \alpha_{C'}\,\hat{y}(x;\theta_{C'})\end{aligned}$$

$$\ell(\mathcal{D}^{\text{train}}, \hat{y}^{(C')}) = \sum_{n=1}^{N} \ell(y_n, \hat{y}^{(C')}(x_n))$$

$$(\alpha_{C'}, \theta_{C'}) := \operatorname*{arg\,min}_{\alpha_{C'}, \theta_{C'}} \sum_{n=1}^{N} \ell(y_n, \hat{y}^{(C'-1)}(x_n) + \alpha_{C'}\,\hat{y}(x_n;\theta_{C'}))$$

Boosting

Idea: fit models (and their combination weights)

- sequentially, one at a time,
- relative to the ones already fitted,
- but do not consider to change the earlier ones again.

$$\hat{y}^{(C')}(x) := \sum_{c=1}^{C'} \alpha_c \hat{y}(x;\theta_c), \quad C' \in \{1,\ldots,C\}$$

$$= \hat{y}^{(C'-1)}(x) + \alpha_{C'}\,\hat{y}(x;\theta_{C'})$$

$$\ell(\mathcal{D}^{\text{train}}, \hat{y}^{(C')}) = \sum_{n=1}^{N} \ell(y_n, \hat{y}^{(C')}(x_n))$$

$$(\alpha_{C'}, \theta_{C'}) := \arg\min_{\alpha_{C'},\theta_{C'}} \sum_{n=1}^{N} \ell(y_n, \underbrace{\hat{y}^{(C'-1)}(x_n)}_{=:\hat{y}_n^0} + \underbrace{\alpha_{C'}}_{=:\alpha}\underbrace{\hat{y}(x_n;\theta_{C'})}_{=:\hat{y}_n})$$

L2 Loss

Assume L2 loss:

$$\ell(y, \hat{y}) := (y - \hat{y})^2$$

Q: How can we simplify

$$\ell(y_n, \hat{y}_n^0 + \alpha \hat{y}_n) = ?$$

A. approximate y_n by $\hat{y}_n^0$: $\quad = \ell(\hat{y}_n^0, \alpha \hat{y}_n)$

B. approximate α by 1: $\quad = \ell(y_n, \hat{y}_n^0 + \hat{y}_n)$

C. bring $\hat{y}_n^0$ on the other side: $\quad = \ell(y_n - \hat{y}_n^0, \alpha \hat{y}_n)$

D. divide by $\hat{y}_n^0$: $\quad = \ell(\frac{y_n}{\hat{y}_n^0}, 1 + \alpha \hat{y}_n)$

L2 Loss Boosting (Least Squares Boosting)

For L2 loss

$$\ell(y, \hat{y}) := (y - \hat{y})^2$$

we get

$$\ell(y_n, \hat{y}_n^0 + \alpha \hat{y}_n) = \ell(y_n - \hat{y}_n^0, \alpha \hat{y}_n)$$

and thus **fit the residuals**

$$\theta_{C'} := \arg\min_{\theta_{C'}} \sum_{n=1}^{N} \ell(y_n - \hat{y}_n^0, \hat{y}(x_n; \theta_{C'}))$$

$$\alpha_{C'} := 1$$

Works for any loss with

$$\ell(y, \hat{y}) = s(y - \hat{y}), \quad \text{for a function } s, \text{e.g., } s(z) = z^2$$

e.g., L2, L1 etc.

Convergence & Shrinking

Models are fitted iteratively

$$C' := 1, 2, 3, \ldots$$

- convergence is assessed via **early stopping**: once the error on a validation sample

$$\ell(\mathcal{D}^{\text{val}}, \hat{y}^{(C')})$$

  does not decrease anymore over a couple of iterations, the algorithm stops and returns the best iteration so far.

- To slow down convergence to the training data, usually **shrinking the combination weights** is applied:

$$\alpha_{C'} := \nu\, \alpha_{C'}, \quad \text{e.g., with } \nu = 0.02$$

L2 Loss Boosting / Algorithm

$\textbf{l2boost}(\mathcal{D}^{\text{train}} = \{(x_1, y_1), \ldots, (x_N, y_N)\}, C, \nu)$:

$\quad \tilde{y}_n := y_n, \quad n = 1 : N$

$\quad$ for $c := 1, \ldots, C$:

$\quad\quad \tilde{\mathcal{D}}^{\text{train}} := \{(x_n, \tilde{y}_n) \mid n = 1 : N\}$

$\quad\quad \theta_c := \arg\min_\theta \ell(\tilde{\mathcal{D}}^{\text{train}}, \hat{y}(\theta))$

$\quad\quad \alpha_c := \nu$

$\quad\quad \tilde{y}_n := \tilde{y}_n - \alpha_c \hat{y}(x_n, \theta_c), \quad n = 1 : N$

return (α, θ)

- $C \in \mathbb{N}$ number of component models
- $\nu \in (0, 1]$ step length
- $\arg\min_\theta \ell(\tilde{\mathcal{D}}^{\text{train}}, \hat{y}(\theta))$ fits a classifier to predictors x_n and residuals $\tilde{y}_n$

Outline

1. Idea & L2 Loss Boosting

2. Exponential Loss Boosting (AdaBoost)

3. Functional Gradient Descent (Gradient Boosting)

Exponential Loss Boosting (AdaBoost)

For (weighted) exponential loss

$$\ell(y, \hat{y}, w) := w\, e^{-y\hat{y}}, \quad y \in \{-1, +1\}, \hat{y} \in \mathbb{R}$$

we get

$$\ell(y_n, \hat{y}_n^0 + \alpha \hat{y}_n, w_n^0) = \ell(y_n, \hat{y}_n^0, w_n^0)\, \ell(y_n, \alpha \hat{y}_n, 1)$$

Exponential Loss Boosting (AdaBoost)

For (weighted) exponential loss

$$\ell(y, \hat{y}, w) := w\, e^{-y\hat{y}}, \quad y \in \{-1, +1\}, \hat{y} \in \mathbb{R}$$

we get

$$\begin{aligned}\ell(y_n, \hat{y}_n^0 + \alpha\hat{y}_n, w_n^0) =& \underbrace{\ell(y_n, \hat{y}_n^0, w_n^0)}_{=: w_n} \ell(y_n, \alpha\hat{y}_n, 1) \\ =& \ell(y_n, \alpha\hat{y}_n, w_n)\end{aligned}$$

Exponential Loss Boosting (AdaBoost)

Discrete models with $\hat{y} \in \{+1, -1\}$ are fitted in two steps:

1. Learn the next discrete model $\theta_{C'}$:

$$\hat{\theta}_{C'} := \arg\min_{\theta_{C'}} \sum_{n=1}^{N} \ell(y_n, \hat{y}(x_n, \theta_{C'}), w_n^{(C')})$$

2. Learn $\alpha_{C'}$:

$$\hat{\alpha}_{C'} := \arg\min_{\alpha_{C'}} \sum_{n=1}^{N} \ell(y_n, \alpha_{C'} \hat{y}(x_n, \theta_{C'}), w_n^{(C')})$$

Exponential Loss Boosting (AdaBoost) / Learning $\alpha_{C'}$

Optimal $\alpha_{C'}$ can be found analytically:

$$\min_\alpha \quad \sum_{n=1}^N \ell(y_n, \alpha\hat{y}_n, w_n) = \sum_{n=1}^N w_n e^{-\alpha y_n \hat{y}_n}$$

$$0 \stackrel{!}{=} \frac{\partial(\ldots)}{\partial\alpha} = -\sum_{n=1}^N w_n y_n \hat{y}_n e^{-\alpha y_n \hat{y}_n}$$

$$\stackrel{*)}{=} -e^{-\alpha}\sum_{n=1}^N w_n \delta(y_n = \hat{y}_n) + e^{\alpha}\sum_{n=1}^N w_n \delta(y_n \neq \hat{y}_n)$$

$$= -e^{-\alpha}\sum_{n=1}^N w_n + (e^{\alpha} + e^{-\alpha})\sum_{n=1}^N w_n \delta(y_n \neq \hat{y}_n)$$

$$\frac{e^{-\alpha}}{e^{\alpha} + e^{-\alpha}} = \frac{\sum_{n=1}^N w_n \delta(y_n \neq \hat{y}_n)}{\sum_{n=1}^N w_n} = \text{err}$$

Note: *) assuming a crisp $\hat{y}_n \in \{-1, +1\}$.

Exponential Loss Boosting (AdaBoost) / Learning $\alpha_{C'}$

$$
\begin{aligned}
\frac{e^{-\alpha}}{e^{\alpha}+e^{-\alpha}} &= \text{err} \\
\frac{e^{\alpha}+e^{-\alpha}}{e^{-\alpha}} &= \frac{1}{\text{err}} \\
e^{2\alpha}+1 &= \frac{1}{\text{err}} \\
e^{2\alpha} &= \frac{1}{\text{err}} - 1 = \frac{1-\text{err}}{\text{err}} \\
\alpha &= \frac{1}{2}\log\frac{1-\text{err}}{\text{err}}
\end{aligned}
$$

Exponential Loss Boosting (AdaBoost)

The loss in iteration C'

$$\arg\min_{\alpha,\hat{y}_n} \sum_{n=1}^{N} \ell(y_n, \alpha\hat{y}_n, w_n) = \arg\min_{\alpha_{C'},\theta_{C'}} \sum_{n=1}^{N} \ell(y_n, \alpha_{C'}\hat{y}(x_n, \theta_{C'}), w_n^{(C')})$$

is minimized sequentially:

1. Learn $\theta_{C'}$:

$$w_n^{(C')} := \ell(y_n, \hat{y}^{(C'-1)}(x_n), w_n^{(C'-1)})$$

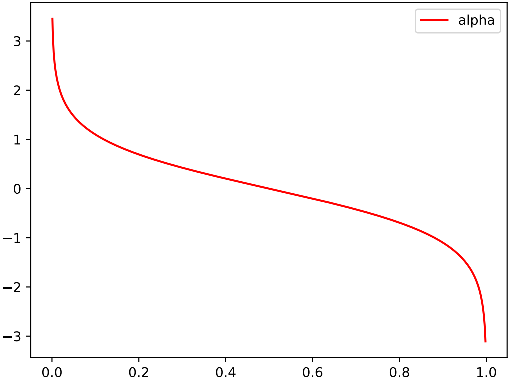
$$\hat{\theta}_{C'} := \arg\min_{\theta_{C'}} \sum_{n=1}^{N} \ell(y_n, \hat{y}(x_n, \theta_{C'}), w_n^{(C')})$$

2. Learn $\alpha_{C'}$:

$$\mathrm{err}_{C'} := \frac{\sum_{n=1}^{N} w_n^{(C')}\delta(y_n \neq \hat{y}(x_n, \theta_{C'}))}{\sum_{n=1}^{N} w_n^{(C')}}$$

$$\alpha_{C'} := \frac{1}{2}\log\frac{1-\mathrm{err}_{C'}}{\mathrm{err}_{C'}}$$

$\alpha_{C'}(\text{err}_{C'})$



AdaBoost

1 $\textbf{adaboost}(\mathcal{D}^{\text{train}} = \{(x_1, y_1), \ldots, (x_N, y_N)\}, C)$:

2 $\quad w_n := \frac{1}{N}, \quad n := 1, \ldots, N$

3 $\quad$ for $c := 1, \ldots, C$:

4 $\qquad \theta_c := \arg\min_\theta \ell(\mathcal{D}^{\text{train}}, \hat{y}(\theta), w)$

5 $\qquad \text{err}_C := \frac{\sum_{n=1}^N w_n \delta(y_n \neq \hat{y}(x_n, \theta_c))}{\sum_{n=1}^N w_n}$

6 $\qquad \alpha_c := \log \frac{1 - \text{err}_c}{\text{err}_c}$

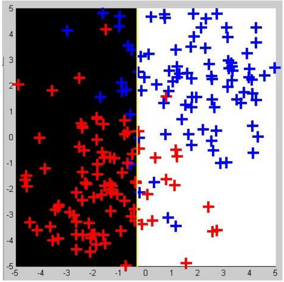
7 $\qquad w_n := w_n e^{\alpha_c \delta(y_n \neq \hat{y}(x_n, \theta_c))}, \quad n = 1, \ldots, N$

8 return (α, θ)

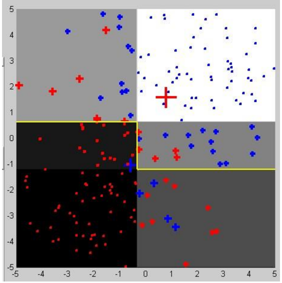
- C number of component models
- $\arg\min_\theta \ell(\mathcal{D}^{\text{train}}, \hat{y}(\theta), w)$ fits a classifier to data with case weights w

Note: Here α is inflated by a factor of 2 as in [**?**, alg. 10.1].
The error in line 7 is for a crisp $\hat{y}(x_n, \theta_c) \in \{-1, +1\}$.

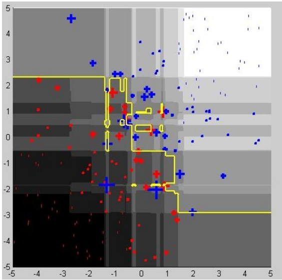
AdaBoost / Example (Decision Tree Stumps)



$C' = 1$



$C' = 3$



$C' = 120$

[?, fig. 16.10]

Outline

Functional Gradient Descent

So far, we have to derive the boosting equations **for each loss individually**.

Idea:

- compute the **gradient of the loss function** for an additional additive term and
- fit the next model that **mimicks best a gradient update step**

Advantage:

- works **for all differentiable losses**.

Functional Gradient Descent

Functional gradient:

$$\nabla_{\hat{y}} \ell(\mathcal{D}^{\text{train}}, \hat{y})|_{\hat{y}^{(C'-1)}} = \nabla_{\hat{y}} \left(\sum_{n=1}^{N} \ell(y_n, \hat{y}_n) \right) |_{\hat{y}^{(C'-1)}}$$

$$= \left(\frac{\partial \ell}{\partial \hat{y}} (y_n, \hat{y}^{(C'-1)}(x_n)) \right)_{n=1,\ldots,N}$$

Functional Gradient Descent

Functional gradient:

$$\begin{aligned}\nabla_{\hat{y}} \ell(\mathcal{D}^{\text{train}}, \hat{y})|_{\hat{y}^{(C'-1)}} =& \nabla_{\hat{y}} \left(\sum_{n=1}^{N} \ell(y_n, \hat{y}_n) \right) |_{\hat{y}^{(C'-1)}} \\ =& \left(\frac{\partial \ell}{\partial \hat{y}} (y_n, \hat{y}^{(C'-1)}(x_n)) \right)_{n=1,\ldots,N}\end{aligned}$$

A functional gradient update step would do:

$$\hat{y}^{(C')} = \hat{y}^{(C'-1)} - \eta \nabla_{\hat{y}} \ell(\mathcal{D}^{\text{train}}, \hat{y})$$

Functional Gradient Descent

Functional gradient:

$$\nabla_{\hat{y}} \ell(\mathcal{D}^{\text{train}}, \hat{y})|_{\hat{y}^{(C'-1)}} = \nabla_{\hat{y}} \left(\sum_{n=1}^{N} \ell(y_n, \hat{y}_n) \right) |_{\hat{y}^{(C'-1)}}$$
$$= \left(\frac{\partial \ell}{\partial \hat{y}} (y_n, \hat{y}^{(C'-1)}(x_n)) \right)_{n=1,\ldots,N}$$

A functional gradient update step would do:

$$\hat{y}^{(C')} = \hat{y}^{(C'-1)} - \eta \nabla_{\hat{y}} \ell(\mathcal{D}^{\text{train}}, \hat{y})$$

Boosting adds the next model:

$$\hat{y}^{(C')} = \hat{y}^{(C'-1)} + \alpha_{C'} \hat{y}(\theta_{C'})$$

Functional Gradient Descent

Functional gradient:

$$
\begin{aligned}
\nabla_{\hat{y}}\ell(\mathcal{D}^{\text{train}}, \hat{y})|_{\hat{y}^{(C'-1)}} =& \nabla_{\hat{y}} \left(\sum_{n=1}^{N} \ell(y_n, \hat{y}_n) \right) |_{\hat{y}^{(C'-1)}} \\
=& \left(\frac{\partial \ell}{\partial \hat{y}}(y_n, \hat{y}^{(C'-1)}(x_n)) \right)_{n=1,\dots,N}
\end{aligned}
$$

A functional gradient update step would do:

$$
\hat{y}^{(C')} = \hat{y}^{(C'-1)} - \eta \nabla_{\hat{y}}\ell(\mathcal{D}^{\text{train}}, \hat{y})
$$

Boosting adds the next model:

$$
\hat{y}^{(C')} = \hat{y}^{(C'-1)} + \alpha_{C'} \hat{y}(\theta_{C'})
$$

To mimick the gradient update step with steplength $\eta := 1$:

$$
\theta_{C'} := \arg\min_{\theta_{C'}} \sum_{n=1}^{N} (- \left(\nabla_{\hat{y}}\ell(\mathcal{D}^{\text{train}}, \hat{y})|_{\hat{y}^{(C'-1)}} \right)_n - \hat{y}(x_n, \theta_{C'}))^2
$$

Gradient Boosting / Algorithm

$$
\begin{array}{l}
\textbf{gradient-boost}(\mathcal{D}^{\text{train}} = \{(x_1, y_1), \ldots, (x_N, y_N)\}, C, \nu): \\
\quad \hat{y}_n := 0, \quad n := 1, \ldots, N \\
\quad g_n := y_n, \quad n := 1, \ldots, N \\
\quad \text{for } \ c := 1, \ldots, C: \\
\quad\quad \tilde{\mathcal{D}}^{\text{train}} := \{(x_n, g_n) \mid n = 1 : N\} \\
\quad\quad \theta_c := \arg\min_\theta \ell(\tilde{\mathcal{D}}^{\text{train}}, \hat{y}(\theta)) \\
\quad\quad \alpha_c := \nu \\
\quad\quad \hat{y}_n := \hat{y}_n + \alpha_c \hat{y}(x_n, \theta_c), \quad n = 1, \ldots, N \\
\quad\quad g_n := -\ell'(y_n, \hat{y}_n), \quad n = 1, \ldots, N \\
\text{return } \ (\alpha, \theta)
\end{array}
$$

- $C \in \mathbb{N}$ number of component models
- $\nu \in (0, 1]$ step length
- $\arg\min_\theta \ell(\tilde{\mathcal{D}}^{\text{train}}, \hat{y}(\theta))$ fits a classifier to predictors x_n and gradients g_n

Note: $\ell' := \frac{\partial \ell}{\partial \hat{y}}$

Performance Comparison / Low Dimensional Data

MODEL	1ST	2ND	3RD	4TH	5TH	6TH	7TH	8TH	9TH	10TH
BST-DT	0.580	0.228	0.160	0.023	0.009	0.000	0.000	0.000	0.000	0.000
RF	0.390	0.525	0.084	0.001	0.000	0.000	0.000	0.000	0.000	0.000
BAG-DT	0.030	0.232	0.571	0.150	0.017	0.000	0.000	0.000	0.000	0.000
SVM	0.000	0.008	0.148	0.574	0.240	0.029	0.001	0.000	0.000	0.000
ANN	0.000	0.007	0.035	0.230	0.606	0.122	0.000	0.000	0.000	0.000
KNN	0.000	0.000	0.000	0.009	0.114	0.592	0.245	0.038	0.002	0.000
BST-STMP	0.000	0.000	0.002	0.013	0.014	0.257	0.710	0.004	0.000	0.000
DT	0.000	0.000	0.000	0.000	0.000	0.000	0.004	0.616	0.291	0.089
LOGREG	0.000	0.000	0.000	0.000	0.000	0.000	0.040	0.312	0.423	0.225
NB	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.030	0.284	0.686

Table 16.3 Fraction of time each method achieved a specified rank, when sorting by mean performance across 11 datasets and 8 metrics. Based on Table 4 of (Caruana and Niculescu-Mizil 2006). Used with kind permission of Alexandru Niculescu-Mizil.

11 datasets, $\sim$ 10.000 instances, 9-200 variables

[?, p. 582]

Performance Comparison / High Dimensional Data

TABLE 11.3. *Performance of different methods. Values are average rank of test error across the five problems (low is good), and mean computation time and standard error of the mean, in minutes.*

Method	Screened Features		ARD Reduced Features	
	Average Rank	Average Time	Average Rank	Average Time
Bayesian neural networks	1.5	384(138)	1.6	600(186)
Boosted trees	3.4	3.03(2.5)	4.0	34.1(32.4)
Boosted neural networks	3.8	9.4(8.6)	2.2	35.6(33.5)
Random forests	2.7	1.9(1.7)	3.2	11.2(9.3)
Bagged neural networks	3.6	3.5(1.1)	4.0	6.4(4.4)

5 datasets, 100–6.000 instances, 500-100.000 variables

[?, p. 414]

Summary

- **Boosting** learns the component models of an ensemble sequentially.
- for L2 regression,
  - the next model predicts the **residuum** of the sum of the previous models (**L2 boosting**)
- for exponential loss classification,
  - the instance losses of the sum of the previous models are used as **case weights** (**AdaBoost**)
- **Gradient Boosting** uses functional gradient descent to mimick gradient update steps
  - accomplished by predicting the **loss gradient w.r.t.** $\hat{y}_{1:N}$
  - works for all differentiable losses

Further Readings

- Boosting: [**?**, chapter 16.4], [**?**, chapter 10], [**?**, chapter 14.3].
- Also interesting:
  - Xgboost [**?**].
  - DeepBoost [**?**].
  - distributed boosting [**?**]

References

Christopher M. Bishop.
Pattern recognition and machine learning, volume 1.
springer New York, 2006.

Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin.
The elements of statistical learning: data mining, inference and prediction, volume 27.
Springer, 2005.

Kevin P. Murphy.
Machine learning: a probabilistic perspective.
The MIT Press, 2012.