Sapienza University of Rome

Master in Artificial Intelligence and Robotics
Master in Engineering in Computer Science

Machine Learning

A.Y. 2020/2021

Prof. L. Iocchi, F. Patrizi

Sapienza University of Rome, Italy - Machine Learning (2020/2021)

6. Probabilistic models for classification

L. Iocchi, F. Patrizi

Overview

- Probabilistic generative models
- Probabilistic discriminative models
- Logistic regression

References

C. Bishop. Pattern Recognition and Machine Learning. Sect. 4.2, 4.3

Probabilistic Models for Classification

Given $f : X \rightarrow C$, $D = \{(x_i, c_i)_{i=1}^n\}$ and $\mathbf{x} \notin D$, estimate

$$P(C_i|\mathbf{x}, D)$$

Simplified notation without D in the formulas.

$P(C_i|\mathbf{x})$: posterior, $P(\mathbf{x}|C_i)$: class-conditional densities

Two families of models:

- Generative: estimate $P(\mathbf{x}|C_i)$ and then compute $P(C_i|\mathbf{x})$ with Bayes
- Discriminative: estimate $P(C_i|\mathbf{x})$ directly

Probabilistic Generative Models

Consider first the case of two classes.

Find the conditional probability:

$$P(C_1|\mathbf{x}) = \frac{P(\mathbf{x}|C_1)P(C_1)}{P(\mathbf{x})} = \frac{P(\mathbf{x}|C_1)P(C_1)}{P(\mathbf{x}|C_1)P(C_1) + P(\mathbf{x}|C_2)P(C_2)}$$
$$= \frac{1}{1+\exp(-a)} = \sigma(a)$$

with:

$$a = \ln \frac{p(\mathbf{x}|C_1)P(C_1)}{p(\mathbf{x}|C_2)P(C_2)}$$

and

$$\sigma(a) = \frac{1}{1+\exp(-a)} \text{ the } \textit{sigmoid function}.$$

Probabilistic Generative Models

Assume $p(\mathbf{x}|C_i) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma})$ - same covariance matrix

$$a = \ln \frac{p(\mathbf{x}|C_1)P(C_1)}{p(\mathbf{x}|C_2)P(C_2)} = \ln \frac{\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_1, \boldsymbol{\Sigma})P(C_1)}{\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_2, \boldsymbol{\Sigma})P(C_2)} = \ldots = \mathbf{w}^T\mathbf{x} + w_0$$

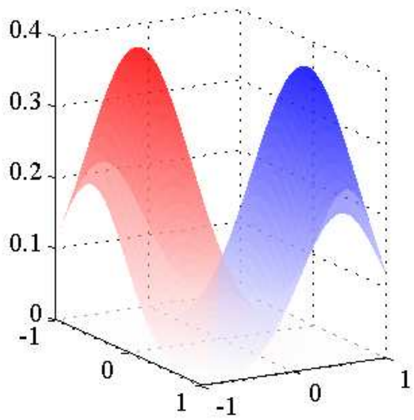
with:

$$\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2),$$
$$w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln\frac{P(C_1)}{P(C_2)}.$$

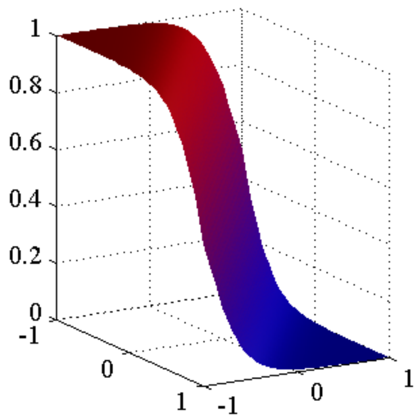
Thus

$$P(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + w_0),$$

Probabilistic Generative Models



$P(\mathbf{x}|C_1), P(\mathbf{x}|C_2)$



$P(C_1|\mathbf{x})$

Decision rule: $c = C_1 \iff P(c = C_1|\mathbf{x}) > 0.5$

Probabilistic Generative Models Multi-class

$$P(C_k|\mathbf{x}) = \frac{P(\mathbf{x}|C_k)P(C_k)}{\sum_j P(\mathbf{x}|C_j)P(C_j)} = \frac{exp(a_k)}{\sum_j exp(a_j)}$$

(normalized exponential or softmax function)

with $a_k = \ln P(\mathbf{x}|C_k)P(C_k)$

Maximum likelihood

Maximum likelihood solution for 2 classes

Assuming $P(C_1) = \pi$ (thus $P(C_2) = 1 - \pi$), $P(\mathbf{x}|C_i) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma})$

Given data set $D = \{(\mathbf{x}_n, t_n)_{n=1}^N\}$, $t_n = 1$ if $\mathbf{x}_n$ belongs to class C_1, $t_n = 0$ if $\mathbf{x}_n$ belongs to class C_2

Let N_1 be the number of samples in D belonging to C_1 and N_2 be the number of samples in C_2 ($N_1 + N_2 = N$)

Likelihood function

$$P(\mathbf{t}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}, D) = \prod_{n=1}^{N} [\pi \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_1, \boldsymbol{\Sigma})]^{t_n} [(1-\pi)\mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_2, \boldsymbol{\Sigma})]^{(1-t_n)}$$

Unknown π, $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$, $\boldsymbol{\Sigma}$

Maximum likelihood

Maximum likelihhod solution for 2 classes
Maximizing log likelihood function, we obtain

$$\pi = \frac{N_1}{N}$$

$$\boldsymbol{\mu}_1 = \frac{1}{N_1} \sum_{n=1}^{N} t_n \mathbf{x}_n \qquad \boldsymbol{\mu}_2 = \frac{1}{N_2} \sum_{n=1}^{N} (1 - t_n) \mathbf{x}_n$$

$$\boldsymbol{\Sigma} = \frac{N_1}{N} S_1 + \frac{N_2}{N} S_2$$

with $S_i = \frac{1}{N_i} \sum_{n \in C_i} (\mathbf{x}_n - \boldsymbol{\mu}_i)(\mathbf{x}_n - \boldsymbol{\mu}_i)^T$, $i = 1, 2$

Note: details in C. Bishop. PRML. Section 4.2.2

Maximum likelihood for K classes

Gaussian Naive Bayes

$P(C_k) = \pi_k$, $P(\mathbf{x}|C_k) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma})$

Data set $D = \{(\mathbf{x}_n, \mathbf{t}_n)_{n=1}^N\}$, with $\mathbf{t}_n$ 1-of-K encoding

$$\pi_k = \frac{N_k}{N}$$

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} t_{nk} \, \mathbf{x}_n$$

$$\boldsymbol{\Sigma} = \sum_{k=1}^{K} \frac{N_k}{N} S_k \ , \quad S_k = \frac{1}{N_k} \sum_{n=1}^{N} t_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T$$

Probabilistic Models

Represent posterior distributions with parametric models.

For two classes

$$P(C_1|\mathbf{x}) = \sigma(a)$$

For $k \geq 2$ classes

$$P(C_i|\mathbf{x}) = \frac{exp(a_k)}{\sum_j exp(a_j)}$$

$$a_k = \mathbf{w}^T \mathbf{x} + w_0$$

This is valid for all the class-conditional distributions in the exponential family (including Gaussians). [Bishop, Sect. 4.2.4]

Compact notation

$$\mathbf{w}^T\mathbf{x} + w_0 = (w_0 \; \mathbf{w}) \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}$$

$$\tilde{\mathbf{w}} = \begin{pmatrix} w_0 \\ \mathbf{w} \end{pmatrix}, \tilde{\mathbf{x}} = \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}$$

$$a_k = \mathbf{w}^T\mathbf{x} + w_0 = \tilde{\mathbf{w}}^T \, \tilde{\mathbf{x}}$$

Maximum Likelihood for parametric models

Likelihood for a parametric model $\mathcal{M}_\Theta$: $P(\mathbf{t}|\Theta, D)$, $D = \langle \mathbf{X}, \mathbf{t} \rangle$

Maximum likelihood solution:

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}} \ln P(\mathbf{t}|\Theta, \mathbf{X})$$

When $\mathcal{M}_\Theta$ belongs to the exponential family, likelihood $P(\mathbf{t}|\Theta, \mathbf{X})$ can be expressed in the form $P(\mathbf{t}|\tilde{\mathbf{w}}, \mathbf{X})$, with maximum likelihood

$$\tilde{\mathbf{w}}^* = \underset{\tilde{\mathbf{w}}}{\operatorname{argmax}} \ln P(\mathbf{t}|\tilde{\mathbf{w}}, \mathbf{X})$$

Probabilistic Discriminative Models

Estimate directly

$$P(C_i|\tilde{\mathbf{x}}, D) = \frac{exp(a_k)}{\sum_j exp(a_j)} \qquad a_k = \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}$$

with maximum likelihood

$$\tilde{\mathbf{w}}^* = \underset{\tilde{\mathbf{w}}}{\operatorname{argmax}} \ln P(\mathbf{t}|\tilde{\mathbf{w}}, \mathbf{X})$$

without estimating the model parameters.

Simplified notation (dataset omitted): $P(\mathbf{t}|\tilde{\mathbf{w}})$

Logistic regression

Probabilistic discriminative model based on maximum likelihhod.

Two classes

Given data set $D = \{(\tilde{\mathbf{x}}_n, t_n)_{n=1}^N\}$, with $t_n \in \{0, 1\}$

Likelihood function:

$$p(\mathbf{t}|\tilde{\mathbf{w}}) = \prod_{n=1}^{N} y_n^{t_n}(1 - y_n)^{1-t_n}$$

with $y_n = p(C_1|\tilde{\mathbf{x}}_n) = \sigma(\tilde{\mathbf{w}}^T \tilde{\mathbf{x}}_n + w_0)$

Note: t_n: value in the data set corresponding to $\mathbf{x}_n$,
y_n: posterior prediction of the current model $\tilde{\mathbf{w}}$ for $\mathbf{x}_n$.

Logistic regression

Cross-entropy error function (negative log likelihood)

$$E(\tilde{\mathbf{w}}) \equiv -\ln p(\mathbf{t}|\tilde{\mathbf{w}}) = -\sum_{n=1}^{N}[t_n \ln y_n + (1-t_n)\ln(1-y_n)]$$

Solution concept: solve the optimization problem

$$\tilde{\mathbf{w}}^* = \operatorname*{argmin}_{\tilde{\mathbf{w}}} E(\tilde{\mathbf{w}})$$

Many solvers available.

Iterative reweighted least squares

Apply *Newton-Raphson* iterative optimization for minimizing $E(\tilde{\mathbf{w}})$.

Gradient of the error with respect to $\tilde{\mathbf{w}}$

$$\nabla E(\tilde{\mathbf{w}}) = \sum_{n=1}^{N}(y_n - t_n)\tilde{\mathbf{x}}_n$$

Gradient descent step

$$\tilde{\mathbf{w}} \leftarrow \tilde{\mathbf{w}} - \boldsymbol{H}(\tilde{\mathbf{w}})^{-1}\nabla E(\tilde{\mathbf{w}})$$

$\boldsymbol{H}(\tilde{\mathbf{w}}) = \nabla\nabla E(\tilde{\mathbf{w}})$ is the Hessian matrix of $E(\tilde{\mathbf{w}})$ (second derivatives with respect to $\tilde{\mathbf{w}}$).

Iterative reweighted least squares

Given

$$\tilde{\mathbf{X}} = \begin{pmatrix} \tilde{\mathbf{x}}_1^T \\ \dots \\ \tilde{\mathbf{x}}_N^T \end{pmatrix} \quad \mathbf{t} = \begin{pmatrix} t_1 \\ \dots \\ t_N \end{pmatrix},$$

$\mathbf{y}(\tilde{\mathbf{w}}) = (y_1, \dots, y_n)^T$ posterior predictions of model $\tilde{\mathbf{w}}$

$\boldsymbol{R}(\tilde{\mathbf{w}})$: diagonal matrix with $R_{nn} = y_n(1 - y_n)$

we have

$$\nabla E(\tilde{\mathbf{w}}) = \tilde{\mathbf{X}}^T(\mathbf{y}(\tilde{\mathbf{w}}) - \mathbf{t})$$

$$\boldsymbol{H}(\tilde{\mathbf{w}}) = \nabla\nabla E(\tilde{\mathbf{w}}) = \sum_{n=1}^{N} y_n(1 - y_n)\tilde{\mathbf{x}}_n\tilde{\mathbf{x}}_n^T = \tilde{\mathbf{X}}^T \boldsymbol{R}(\tilde{\mathbf{w}})\tilde{\mathbf{X}}$$

Iterative reweighted least squares

Iterative method:

1. Initialize $\tilde{\mathbf{w}}$
2. Repeat until termination condition

$$\tilde{\mathbf{w}} \leftarrow \tilde{\mathbf{w}} - (\tilde{\mathbf{X}}^T \boldsymbol{R}(\tilde{\mathbf{w}})\, \tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T(\mathbf{y}(\tilde{\mathbf{w}}) - \mathbf{t})$$

Multiclass logistic regression

K **classes**

$$P(C_k|\tilde{\mathbf{x}}) = \frac{exp(a_k)}{\sum_j exp(a_j)} \qquad a_k = \tilde{\mathbf{w}}_k^T \tilde{\mathbf{x}} \qquad k = 1, \ldots, K$$

$$\tilde{\mathbf{X}} = \begin{pmatrix} \tilde{\mathbf{x}}_1^T \\ \ldots \\ \tilde{\mathbf{x}}_N^T \end{pmatrix} \qquad \mathbf{T} = \begin{pmatrix} t_1^T \\ \ldots \\ t_N^T \end{pmatrix}$$ 1-of-K encoding of labels

$\mathbf{y}_n^T = (y_{n1} \ldots y_{nK})^T$ posterior prediction of $\tilde{\mathbf{x}}_n$ for model $\tilde{\mathbf{w}}_1, \ldots, \tilde{\mathbf{w}}_K$

$$\mathbf{Y}(\tilde{\mathbf{w}}_1, \ldots, \tilde{\mathbf{w}}_K) = \begin{pmatrix} \mathbf{y}_1^T \\ \ldots \\ \mathbf{y}_N^T \end{pmatrix}$$ posterior predictions of model $\tilde{\mathbf{w}}_1, \ldots, \tilde{\mathbf{w}}_K$

Sapienza University of Rome, Italy - Machine Learning (2020/2021)

Multiclass logistic regression

Discriminative model

$$P(\mathbf{T}|\tilde{\mathbf{w}}_1, \ldots, \tilde{\mathbf{w}}_K) = \prod_{n=1}^{N} \prod_{k=1}^{K} P(C_k|\tilde{\mathbf{x}}_n)^{t_{nk}} = \prod_{n=1}^{N} \prod_{k=1}^{K} y_{nk}^{t_{nk}}$$

with $y_{nk} = \mathbf{Y}[n, k]$ and $t_{nk} = \mathbf{T}[n, k]$.

Multiclass logistic regression

Cross-entropy error function

$$E(\tilde{\mathbf{w}}_1, \dots \tilde{\mathbf{w}}_K) = -\ln P(\mathbf{T}|\tilde{\mathbf{w}}_1, \dots \tilde{\mathbf{w}}_K) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln y_{nk}$$

Iterative algorithm

gradient $\nabla_{\tilde{\mathbf{w}}_j} E(\tilde{\mathbf{w}}_1, \dots \tilde{\mathbf{w}}_K) = \dots$

Hessian matrix $\nabla_{\tilde{\mathbf{w}}_k} \nabla_{\tilde{\mathbf{w}}_j} E(\tilde{\mathbf{w}}_1, \dots \tilde{\mathbf{w}}_K) = \dots$

Sapienza University of Rome, Italy - Machine Learning (2020/2021)

Summary

Given a target function $f : X \to C$, and data set D

assume a parametric model for the posterior probability $P(C_k|\tilde{\mathbf{x}}, \tilde{\mathbf{w}})$
$\sigma(\tilde{\mathbf{w}}^T \tilde{\mathbf{x}})$ (2 classes) or $\frac{exp(\tilde{\mathbf{w}}_k^T \tilde{\mathbf{x}})}{\sum_{j=1}^{K} exp(\tilde{\mathbf{w}}_j^T \tilde{\mathbf{x}})}$ (k classes)

Define an error function $E(\tilde{\mathbf{w}})$ (negative log likelihood)

Solve the optimization problem

$$\tilde{\mathbf{w}}^* = \operatorname*{argmin}_{\tilde{\mathbf{w}}} E(\tilde{\mathbf{w}})$$

Classify new sample $\tilde{\mathbf{x}}'$ as C_{k^*} where $k^* = \mathrm{argmax}_{k=1,\dots,K} P(C_k|\tilde{\mathbf{x}}', \tilde{\mathbf{w}}^*)$

Generalization

Given a target function $f : X \to C$, and data set D

assume a prediction parametric model $y(\mathbf{x}; \theta)$, $\quad y(\mathbf{x}; \theta) \approx f(\mathbf{x})$

Define an error function $E(\theta)$

Solve the optimization problem

$$\theta^* = \underset{\theta}{\operatorname{argmin}} E(\theta)$$

Classify new sample $\mathbf{x}'$ as $y(\mathbf{x}'; \theta^*)$

Learning in feature space

All methods described above can be applied in a transformed space of the input (*feature space*).

Given a function $\phi : \tilde{\mathbf{x}} \mapsto \mathbf{\Phi}$ ($\mathbf{\Phi}$ is the *feature space*)
each sample $\tilde{\mathbf{x}}_n$ can be mapped to a feature vector $\boldsymbol{\phi}_n = \phi(\tilde{\mathbf{x}}_n)$

Replacing $\tilde{\mathbf{x}}_n$ with $\boldsymbol{\phi}_n$ in all the equations above, makes the learning system to work in the feature space instead of the input space.

We will see in the next lectures why this trick is useful.