Human-Oriented Robotics

Supervised Learning

Part 1/3

Kai Arras
Social Robotics Lab, University of Freiburg

Winter term 2014/2015

Supervised Learning

Contents

- Introduction and basics
- Bayes Classifier
- Logistic Regression
- Support Vector Machines
- AdaBoost
- k-Nearest Neighbor
- Performance measures
- Cross-validation

Supervised Learning

Why Learning?

- An agent (a robot, an intelligent program) is **learning** if it improves its performance on future tasks after making observations about the world
- But if the design of an agent can be **improved**, why wouldn't the designer just **program** in that improvement?
- Two reasons
  - A designer **cannot anticipate all possible situations** that an autonomous agent might find itself in, particularly in a changing and dynamic world
  - For many tasks, human designers **have just no idea** how to program a solution themselves. Face recognition is an example: easy for humans, difficult to program
- Learning is typically learning a **model** from data
- Learning differs **fundamentally** from **model-based approaches** where the model is **derived from domain knowledge** (in e.g. physics, social science) or human experience

Supervised Learning

Learning Algorithms

- Machine learning algorithms can be organized into a taxonomy based on the desired **outcome** of the algorithm or the type of **input** (feedback)
  - **Supervised Learning:** Inferring a function from labelled training data
    Examples: classification, regression
  - **Unsupervised Learning:** Try to find hidden patterns in unlabeled data
    Examples: clustering, outlier detection
  - **Semi-supervised Learning:** Learn a function from both, labelled and unlabeled data
  - **Reinforcement Learning:** Learn how to act using feedback (rewards) from the world
- Machine learning has become a **key area** for robotics and AI, both as a theoretical foundation and practical toolbox
  - Examples: object recognition from sensory data, learning complex human motion from demonstrations, learning social behavior by imitation, etc.

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Supervised Learning

- The task of supervised learning is as follows: given a **training set** of N example input–output pairs

$$(\mathbf{x}_1, y_1),\ (\mathbf{x}_2, y_2),\ \cdots\ (\mathbf{x}_N, y_N)$$

  where each y was generated by an unknown function $y = f(\mathbf{x})$, **discover a function** h that approximates the true function f
- Let the inputs be vector-valued in general $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ with m **features** or **attributes**

- Function f is also called **discriminant function** or **model**, h is called a **hypothesis**
- In robotics, y often refers to a **state of the world**. Thus, we also use the notation w for y or the more general $\mathbf{w}$ when the state is vector-valued

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

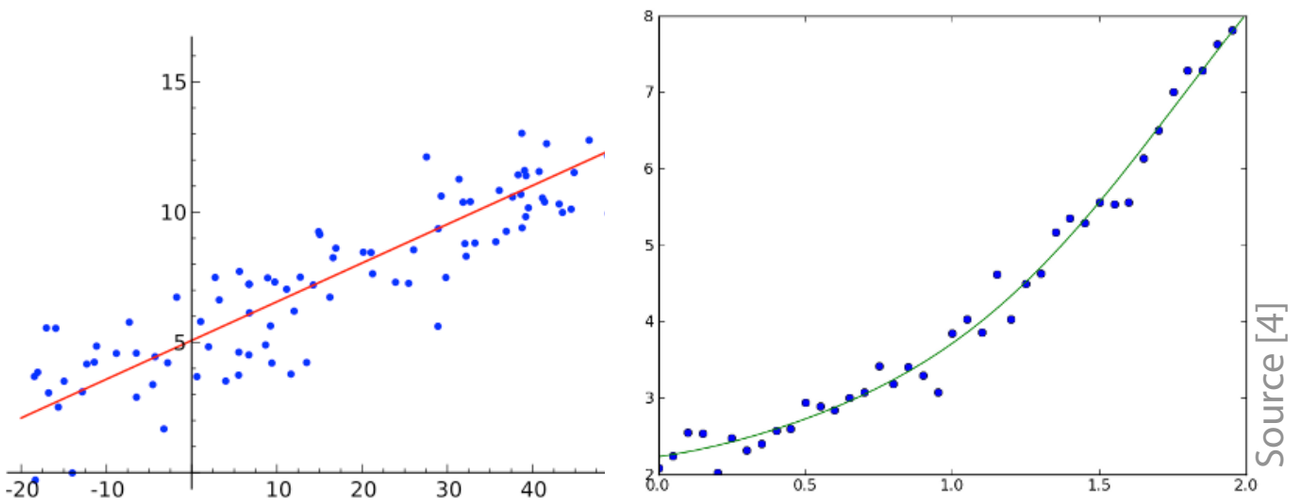
Supervised Learning

- Learning is a **search** through the space of possible hypothesis for one that will perform well, in particular on **new examples** beyond the training set
- The accuracy of a hypothesis is measured on a separate **test set** using common **performance metrics**
- We say a hypothesis **generalizes well** if it correctly predicts the value for y for novel, never seen examples
- This is the case, for example, in perception problems that consist in **measuring a sensory input** $\mathbf{x}$ and **inferring** the state of the world $\mathbf{w}$
  - Examples: an object recognized in 3d point clouds, a person detected in 2D laser data, the room that a robot is in perceived with ultrasonic sensors
- The output y (or world state $\mathbf{w}$) can be **continuous** or **discrete**
  - Example continuous state: human body pose in 3D
  - Example discrete states: presence/absence of a human, a human activity

Supervised Learning

Classification versus Regression

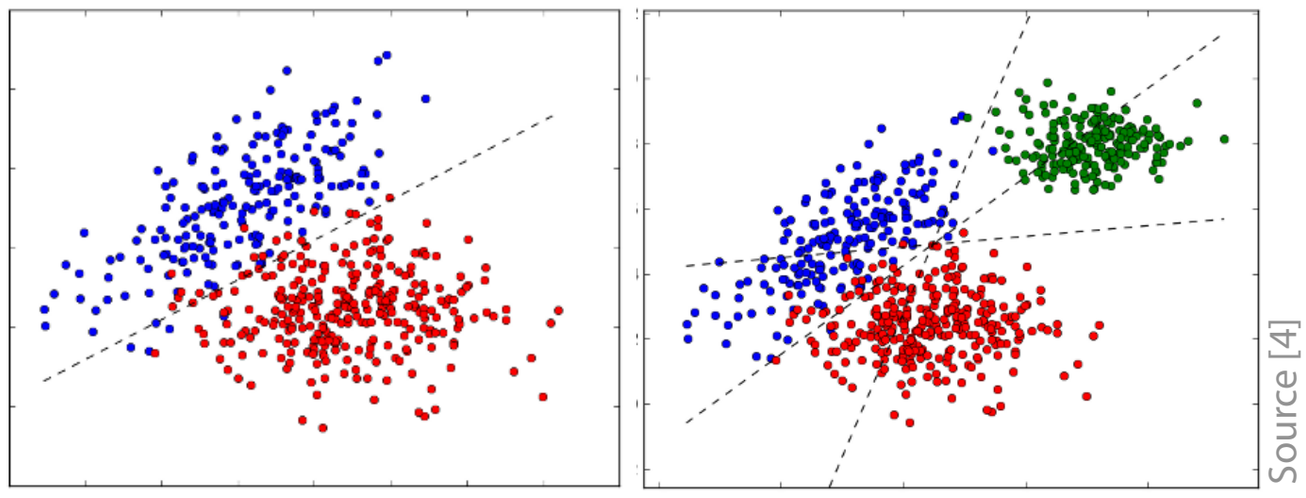
- **Regression:**
  When the world state is **continuous**, we call the inference process regression



Linear regression Nonlinear regression

Source [4]

- **Classification:**
  When the world state is **discrete**, we call the inference process classification

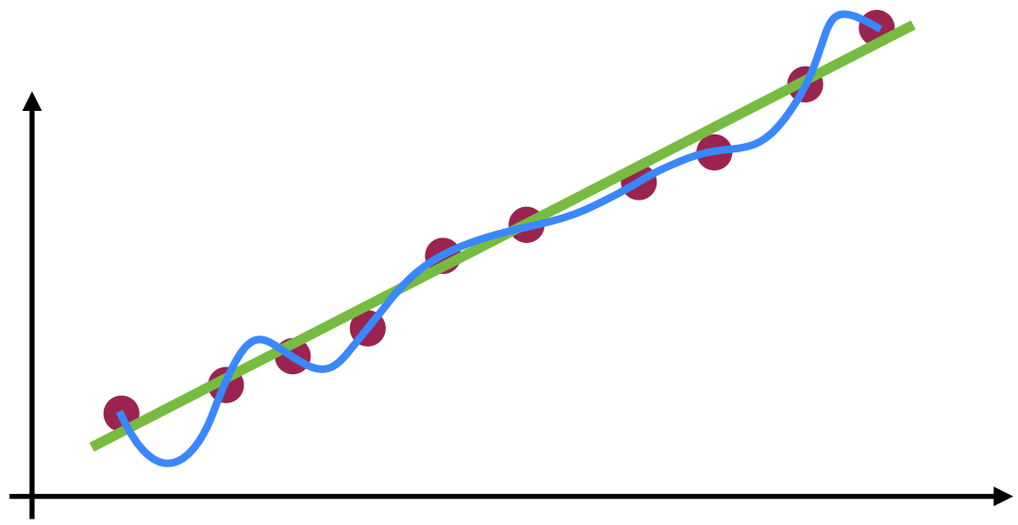


Binary classification Multiway classification

Source [4]

Overfitting

- What means that a hypothesis/model "generalizes well"?
- Overfitting occurs when a model begins to **memorize the training data** rather than learning the underlying relationship
- Occurs typically when fitting a statistical model with too many parameters (e.g. a polynomial of varying degree)
- What to do when several models explain the data perfectly? Take the simplest one according to the principle of **Occam's razor**
- Overfitted models explain training data perfectly but they **do not generalize well**
- There is a **trade-off** between model complexity/better data fit and model simplicity and generalization



Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Posterior Probability Distribution

- A major difficulty in learning is that measurements $\mathbf{x}$ may be **stochastic** and/or **compatible** with **many possible** world states $\mathbf{w}$ (i.e. $\mathbf{w}$ could be an explanation for many different $\mathbf{x}$)
- Reasons: sensory inputs corrupted by **noise** and/or highly **ambiguous**
  - Examples: 2D body pose in image data versus true 3D body pose, auditory data from different human activities, etc.
- In the light of this ambiguity, it would be great to have a **posterior probability distribution** $p(\mathbf{w}|\mathbf{x})$. It would describe everything we know about the world after observing $\mathbf{x}$
- Sometimes, computing $p(\mathbf{w}|\mathbf{x})$ is not **tractable**. In this case, we might compute only the **peak of** $p(\mathbf{w}|\mathbf{x})$, the maximum a posteriori (MAP) solution

Supervised Learning

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Model – Learning – Inference – Decision

To solve a problem of this kind, we need four things:

1. **A model.** The model f relates the (sensory) data $\mathbf{x}$ to the world state $\mathbf{w}$. This is a qualitative choice. A model has parameters $\boldsymbol{\theta}$

2. **A learning algorithm.** The learning algorithm fits the parameters $\boldsymbol{\theta}$ to the data $\mathbf{x}$ using paired training samples $(\mathbf{x}_i, \mathbf{w}_i)$

3. **An inference algorithm.** The inference algorithm takes a new observation $\mathbf{x}$ and computes the posterior $p(\mathbf{w}|\mathbf{x})$ (or approximations thereof) over the world state $\mathbf{w}$

4. **A decision rule.** Takes the posterior probability distribution and makes an optimal (class) assignment of $\mathbf{x}$ onto $\mathbf{w}$

- Sometimes, decision is **postponed** to later stages, e.g. in sensor fusion

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Model – Learning – Inference – Decision

1. **Model examples:**
   - A linear vs. a nonlinear regression model, a nonlinear SVM kernel
   - Example parameters: the coefficients of the polynomial, the kernel parameters
2. **Learning algorithm examples:**
   - Least-squares fit of parameters to data in logistic regression, convex optimization in SVMs, (trivial) storing training data in k-Nearest Neighbor classifier
3. **Inference algorithm examples:**
   - Bayes' rule in Bayesian classifier
4. **Decision rule examples:**
   - Selection of maximum a posteriori class
   - Weighted majority vote in AdaBoost
     (example of combined decision and inference, to be explained later)

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Phases and Data Sets

- Step 2 is called **learning phase**. It consists in learning the parameters $\boldsymbol{\theta}$ of the model f using paired training samples $(\mathbf{x}_i, \mathbf{w}_i)$
- The **test phase** involves steps 3 and 4 using labelled **training** samples $(\mathbf{x}_i, \mathbf{w}_i)$ to estimate how good the model has been trained, evaluated on relevant performance metrics (e.g. classification error)
- The **validation phase** compares several models, obtained, for example, by varying "extrinsic" parameters that cannot be learned. This is to determine the best model where "best" is defined in terms of the performance metrics (see also cross-validation later in this course)
- Sometimes, the term **application phase** denotes the application of the newly learned classifier to real-world data. These data are **unlabeled**
- Accordingly, the data sets that are used in the respective phases are called **training set**, **test set**, and **validation set**

Generative vs. Discriminative Approaches

There are three options for the choice of the model in step 1. In decreasing order of complexity:

- **Generative models** describe the likelihood over the data given the world. Together with a prior, they compute the joint probability over world and data

Joint distribution $p(\mathbf{x},\mathbf{w})$

- **Discriminative models** describe the posterior distribution over the world given the data. Can be used to directly predict the world state for new observations

Posterior distribution $p(\mathbf{w}|\mathbf{x})$

- **Non-probabilistic discriminant functions** map inputs $\mathbf{x}$ directly onto a class label. In this case, probabilities play no role

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Generative vs. Discriminative Approaches

- Common to both probabilistic approaches is that they compute the posterior probability distribution $p(\mathbf{w}|\mathbf{x})$ as hypothesis h to approximate the true underlying function $y = f(\mathbf{x})$
- **Generative classifiers** do this **indirectly** over the likelihood $p(\mathbf{x}|\mathbf{w})$ and the prior $p(\mathbf{w})$. We choose appropriate parametric forms for the distributions and fit the parameters using paired training samples. **Inference** is done using Bayes' rule on new data

$$p(\mathbf{w}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{w})\, p(\mathbf{w})}{\int p(\mathbf{x}|\mathbf{w})\, p(\mathbf{w})\, d\mathbf{w}}$$

- **Discriminative classifiers** do this **directly** over a well chosen parametric model for $p(\mathbf{w}|\mathbf{x})$. In learning, we fit the parameters using paired training samples, **inference** is the direct application of the model to new data

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Classifiers in this Course

- **Generative classifier**
  - (Naive) Bayes Classifier
- **Discriminative classifier**
  - Logistic Regression (a *classification* method despite its name!)
- **Non-probabilistic discriminant classifier**
  - Support Vector Machines
  - k-Nearest Neighbor classifier
  - AdaBoost
- We will mostly consider **binary** classification problems. This is enough to illustrate the essential ideas and simplifies notation
- We will not consider **regression**. Ideas are highly related to classification

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Bayes Classifier

- The Bayes classifier is a **generative** classifier
- The goal is to learn the **posterior probability distribution** $p(\mathbf{w}|\mathbf{x})$ via the **joint distribution** given by the **likelihood** $p(\mathbf{x}|\mathbf{w})$ and **prior** $p(\mathbf{w})$
- In classification, the world state is **discrete** and scalar. We assume that it can take K possible values $w \in \{1,...,K\}$ and use the notation $\mathcal{C}_k$ to denote **class** k for which $w = k$
- Thus, we are seeking to learn $p(\mathcal{C}_k|\mathbf{x})$ by applying **Bayes' rule**

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)\,p(\mathcal{C}_k)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\mathcal{C}_k)\,p(\mathcal{C}_k)}{\sum_{k=1}^{K} p(\mathbf{x}|\mathcal{C}_k)\,p(\mathcal{C}_k)}$$

- We choose **parametric distribution models** for the likelihoods $p(\mathbf{x}|\mathcal{C}_k)$ and the priors $p(\mathcal{C}_k)$ whose **parameters** are learned from the data

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Bayes Classifier

- Suppose $\mathbf{x}$ is **high dimensional** (i.e. data with many features), then the **number of parameters** to estimate $p(\mathbf{x}|\mathcal{C}_k)$ can become very **large**
- Example: if $\mathbf{x}$ is a vector of 30 discrete boolean features and $k \in \{0,1\}$ (binary classification), we need to estimate more than **2 billion** parameters
- To estimate those parameters accurately, a **huge number of training samples** is needed which is impractical in many applications
- We thus require some form of **prior assumption** about the form of the likelihood $p(\mathbf{x}|\mathcal{C}_k)$

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Naive Bayes Classifier

- This is the motivation for the ***Naive*** **Bayes classifier**
- Let us consider the numerator, expand the features/attributes of $\mathbf{x}$ as $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ and repeatedly apply the chain rule

$$
\begin{aligned}
p(\mathbf{x}|\mathcal{C}_k)\, p(\mathcal{C}_k) &= p(x_1, x_2, \ldots, x_m|\mathcal{C}_k)\, p(\mathcal{C}_k) \\
&= p(x_1|\mathcal{C}_k)\, p(x_2, \ldots, x_m|x_1, \mathcal{C}_k)\, p(\mathcal{C}_k) \\
&= p(x_1|\mathcal{C}_k)\, p(x_2|x_1, \mathcal{C}_k)\, p(x_3, \ldots, x_m|x_1, x_2, \mathcal{C}_k)\, p(\mathcal{C}_k)
\end{aligned}
$$

  and so on...

- Now comes the "naiveness" into play: we assume that each attribute is **conditionally independent** of every other attribute given class $\mathcal{C}_k$

Bayes Classifier

Naive Bayes Classifier

- Formally,

$$p(x_i|x_j, \mathcal{C}_k) = p(x_i|\mathcal{C}_k)$$
$$p(x_i|x_j, x_l, \mathcal{C}_k) = p(x_i|\mathcal{C}_k)$$

and so on for all $i \neq j, l$

- Then, the **numerator** becomes

$$p(\mathbf{x}|\mathcal{C}_k)\, p(\mathcal{C}_k) = p(x_1|\mathcal{C}_k)\, p(x_2|\mathcal{C}_k) \cdots p(x_m|\mathcal{C}_k)\, p(\mathcal{C}_k)$$
$$= \prod_{i=1}^{m} p(x_i|\mathcal{C}_k)\, p(\mathcal{C}_k)$$

Bayes Classifier

- Substituting this into the **posterior distribution** over the world state

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)\,p(\mathcal{C}_k)}{p(\mathbf{x})}$$

$$= \frac{p(\mathbf{x}|\mathcal{C}_k)\,p(\mathcal{C}_k)}{\sum_{k=1}^{K} p(\mathbf{x}|\mathcal{C}_k)\,p(\mathcal{C}_k)}$$

leads to

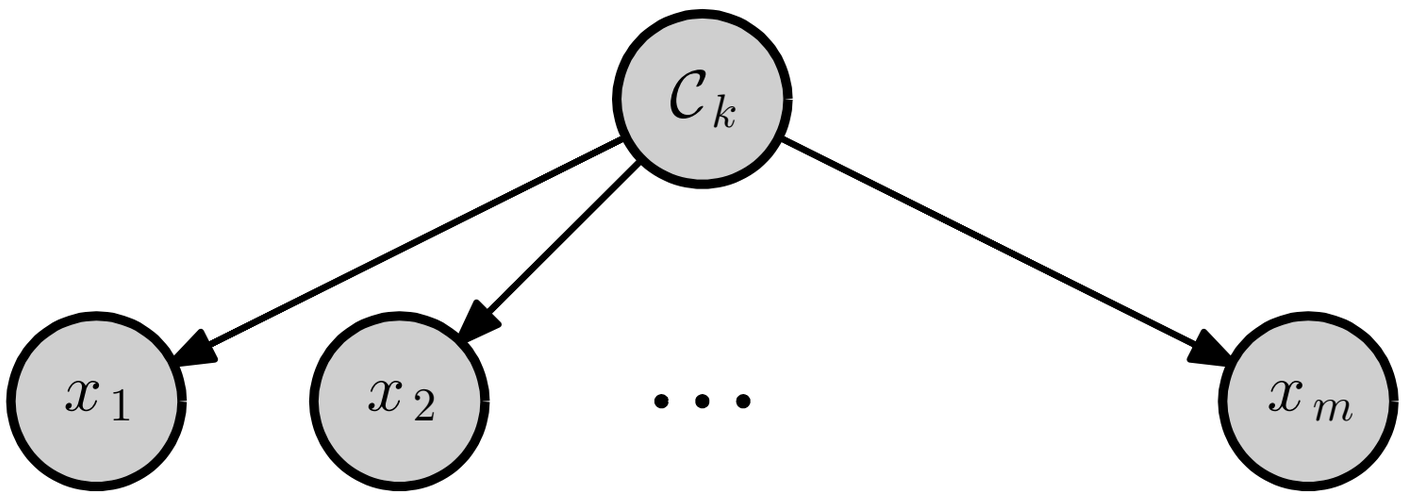
$$p(\mathcal{C}_k|\mathbf{x}) = \frac{\prod_{i=1}^{m} p(x_i|\mathcal{C}_k)\,p(\mathcal{C}_k)}{\sum_{k=1}^{K} p(\mathbf{x}|\mathcal{C}_k)\,p(\mathcal{C}_k)} = \frac{1}{Z}\prod_{i=1}^{m} p(x_i|\mathcal{C}_k)\,p(\mathcal{C}_k)$$

- The **denominator** ensures that all class probabilities sum up to 1. As it is constant and does not depend on the class, it can be **ignored** if only relative class probabilities are of interest (the typical case)

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Naive Bayes Classifier

- The corresponding **graphical model** for the assumption that each feature x_i is conditionally independent of every other feature x_j for all $i \neq j$ given class $\mathcal{C}_k$ is



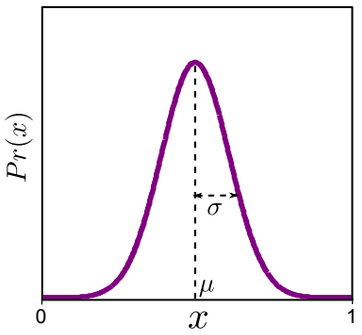
- The number of parameters scales now **linearly** with the dimension of $\mathbf{x}$
- However, the assumption is a **strong** one. It may lead to **poor approximations** of the correct class probabilities

Bayes Classifier

Likelihood and Prior

- We have to make a choice for the **parametric distribution models**
- A common **likelihood** model for real-valued data is the **Gaussian distribution**

$$p(x_i|\mathcal{C}_k) \quad = \quad \mathcal{N}_{x_i}(\mu_k, \sigma_k^2) \qquad \forall\, i \in \{1, \ldots, m\}$$



- In a binary classification problem, $w \in \{0,1\}$, we have one set of parameters $\{\mu_0, \sigma_0^2\}$ for class $\mathcal{C}_0$ and one set $\{\mu_1, \sigma_1^2\}$ for class $\mathcal{C}_1$

$$\begin{aligned} p(x_i|\mathcal{C}_0) &= \mathcal{N}_{x_i}(\mu_0, \sigma_0^2) \\ p(x_i|\mathcal{C}_1) &= \mathcal{N}_{x_i}(\mu_1, \sigma_1^2) \end{aligned}$$

- They are called **class-conditional densities**

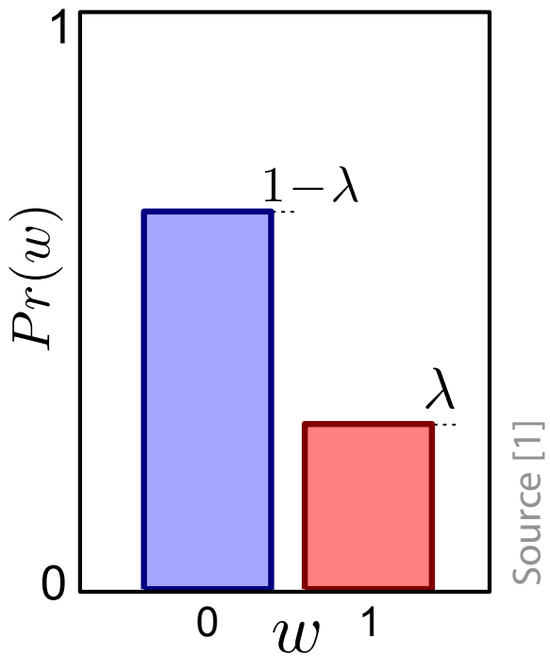
Likelihood and Prior

- For the **prior distributions** over the classes, and in light of a binary classification problem, we may choose a **Bernoulli distribution** with parameter λ

$$p(\mathcal{C}_k) = \text{Bern}_{\mathcal{C}}(\lambda_k)$$

- The Bernoulli distribution has a **single parameter lambda**, which determines the probability of success that

$$p(w = 1) = p(\mathcal{C}_1) = \lambda$$



Source [1]

Bayes Classifier

Learning

- **Learning** consists now in estimating the parameter vector $\boldsymbol{\theta} = \{\mu_0, \sigma_0^2, \mu_1, \sigma_1^2, \lambda\}$ from paired training samples
- Let's look at $\lambda = p(\mathcal{C}_1)$ first: with N_0 being the number of training samples for which $w = 0$ and N_1 the number of training samples for which $w = 1$, the **class priors** can be simply computed as

$$p(\mathcal{C}_0) = \frac{N_0}{N_0 + N_1} \qquad p(\mathcal{C}_1) = \frac{N_1}{N_0 + N_1}$$

- The likelihoods are found by **fitting the parameters** $\{\mu_0, \sigma_0^2\}$ **of each class-conditional density** $p(x_i|\mathcal{C}_0)$ to just the data x_i where the class is 0. Repeat for $\{\mu_1, \sigma_1^2\}$ and data x_i where the class is 1

Bayes Classifier

Learning

- The **maximum likelihood (ML) estimates** of the parameters are then

$$\mu_0 = \frac{1}{N_0} \sum_{j=1}^{N} \delta(w_j = 0)\, x_j \qquad \mu_1 = \frac{1}{N_1} \sum_{j=1}^{N} \delta(w_j = 1)\, x_j$$

$$\sigma_0^2 = \frac{1}{N_0} \sum_{j=1}^{N} \delta(w_j = 0)\, (x_j - \mu_0)^2 \qquad \sigma_1^2 = \frac{1}{N_1} \sum_{j=1}^{N} \delta(w_j = 1)\, (x_j - \mu_1)^2$$

where $\delta(w_j = 0)$ is 1 if $w_j = 0$ and 0 otherwise

- For other, non-Gaussian likelihood models, learning is very **similar**
- Now we have **all** required terms in the expression of the posterior probability which **closes the learning phase**
- The next step is inference, either for the **test phase** where we evaluate the performance on labelled data or for the **application phase** using new data

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Inference

- **Inference** consists in computing the **posterior probability distribution** $p(w|\mathbf{x})$ (or $p(\mathcal{C}_k|\mathbf{x})$ in our classification notation) via application of Bayes' rule for **new observations** $\mathbf{x}$

$$
\begin{aligned}
p(\mathcal{C}_k|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_k)\,p(\mathcal{C}_k)}{\sum_{k=1}^{K} p(\mathbf{x}|\mathcal{C}_k)\,p(\mathcal{C}_k)} \\
&= \frac{1}{Z}\,p(\mathbf{x}|\mathcal{C}_k)\,p(\mathcal{C}_k) \\
&= \frac{1}{Z}\,p(\mathcal{C}_k)\prod_{i=1}^{m} p(x_i|\mathcal{C}_k)
\end{aligned}
$$

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Decision

- Finally, the **decision** consists in assigning data $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ the class whose posterior probability is **maximal** (MAP inference)

$$\arg\max_k \; p(\mathcal{C}_k) \prod_{i=1}^{m} p(x_i|\mathcal{C}_k)$$

- In the binary case, we assign the label $w = 1$ to $\mathbf{x}$ if the **condition** holds

$$1 \; < \; \frac{p(\mathcal{C}_1|\mathbf{x})}{p(\mathcal{C}_0|\mathbf{x})}$$

- A decision rule **divides the space of all** $\mathbf{x}$**'s** into two **decision regions**, one for each class, separated by **decision boundaries**
- Can we say more about those boundaries? What are the **geometrical implications** of our choices in the Bayes classifier?

Bayes Classifier

Decision Boundary

- From the condition

$$1 \; < \; \frac{p(\mathcal{C}_1|\mathbf{x})}{p(\mathcal{C}_0|\mathbf{x})}$$

we can **derive the decision boundary.** Applying Bayes rule and taking the logarithm gives

$$\begin{aligned}
0 \; &< \; \log \frac{p(\mathbf{x}|\mathcal{C}_1)\, p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_0)\, p(\mathcal{C}_0)} \\
0 \; &< \; \log p(\mathbf{x}|\mathcal{C}_1)\, p(\mathcal{C}_1) - \log p(\mathbf{x}|\mathcal{C}_0)\, p(\mathcal{C}_0) \\
0 \; &< \; \log p(\mathbf{x}|\mathcal{C}_1) + \log p(\mathcal{C}_1) - \log p(\mathbf{x}|\mathcal{C}_0) - \log p(\mathcal{C}_0) \\
0 \; &< \; \log p(\mathbf{x}|\mathcal{C}_1) - \log p(\mathbf{x}|\mathcal{C}_0) + \big(\log p(\mathcal{C}_1) - \log p(\mathcal{C}_0)\big)
\end{aligned}$$

Decision Boundary

- Substituting our normally distributed class-conditional density (and assuming $\mathbf{x} = x$ for notation simplicity)

$$p(\mathbf{x}|\mathcal{C}_1) = \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(\frac{-(x-\mu_1)^2}{2\sigma_1^2}\right)$$

into $0 < \log p(\mathbf{x}|\mathcal{C}_1) - \log p(\mathbf{x}|\mathcal{C}_0) + \big(\log p(\mathcal{C}_1) - \log p(\mathcal{C}_0)\big)$ gives

$$0 < \log \frac{1}{\sqrt{2\pi\sigma_1^2}} + \left(\frac{-(x-\mu_1)^2}{2\sigma_1^2}\right) - \log \frac{1}{\sqrt{2\pi\sigma_0^2}} - \left(\frac{-(x-\mu_0)^2}{2\sigma_0^2}\right) + \big(\log p(\mathcal{C}_1) - \log p(\mathcal{C}_0)\big)$$

$$0 < -\frac{(x-\mu_1)^2}{2\sigma_1^2} + \frac{(x-\mu_0)^2}{2\sigma_0^2} + \big(\log \frac{1}{\sqrt{2\pi\sigma_1^2}} - \log \frac{1}{\sqrt{2\pi\sigma_0^2}} + \log p(\mathcal{C}_1) - \log p(\mathcal{C}_0)\big)$$

$$0 < \frac{-(x-\mu_1)^2}{2\sigma_1^2} - \frac{-(x-\mu_0)^2}{2\sigma_0^2} + \theta_0$$

where in the last step we have collected all constant terms w.r.t. x into θ_0

Bayes Classifier

Decision Boundary

- If we assume $\sigma_0 = \sigma_1 = \sigma$ (equal variance of class-conditional densities)

$$0 < \frac{-(x-\mu_1)^2 + (x-\mu_0)^2}{2\sigma^2} + \theta_0$$

$$0 < \frac{-(x^2 - 2\mu_1 x + \mu_1^2) + (x^2 - 2\mu_0 x + \mu_0^2)}{2\sigma^2} + \theta_0$$

$$0 < \frac{(\mu_1 - \mu_0)}{\sigma^2} x + \frac{\mu_0^2 - \mu_1^2}{2\sigma^2} + \theta_0$$

then this is a **linear function** of x of the form

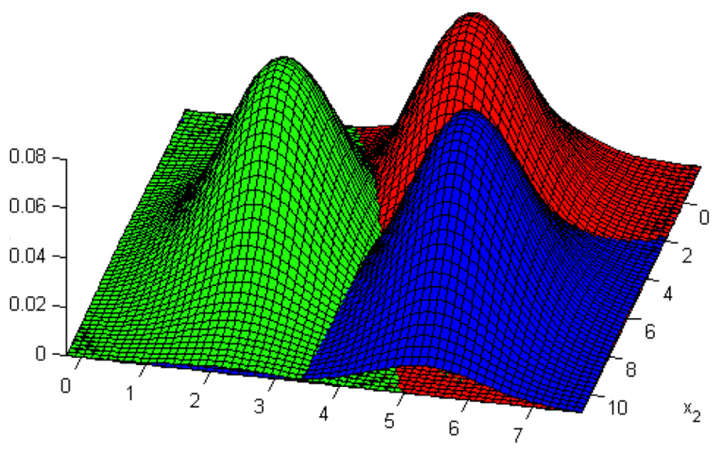
$$0 < \boldsymbol{\theta}^T \mathbf{x} + \theta_0$$

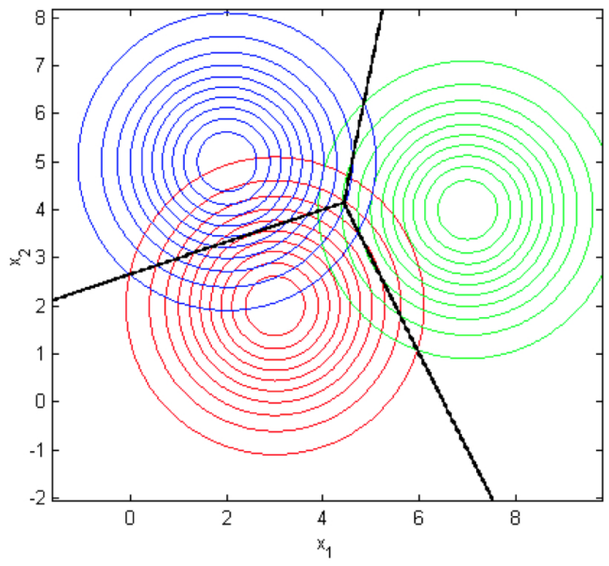
- Therefore, under the assumption of **equal variance** among classes, the Bayes classifier has a **linear decision boundary**

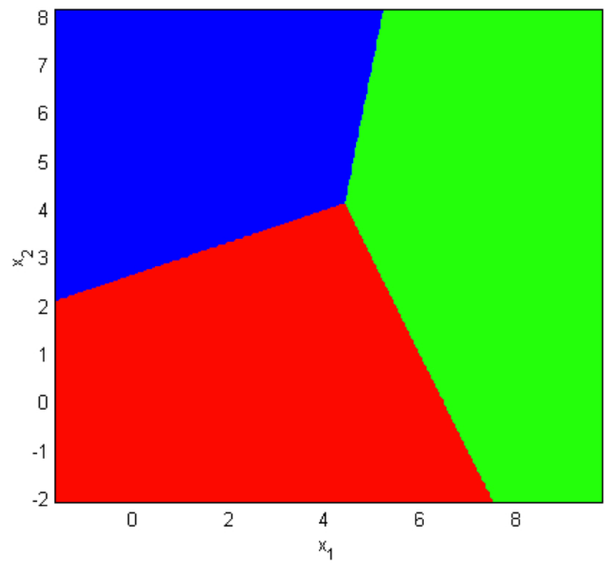
Bayes Classifier

Decision Boundary

- Visualization with three classes:







Source [5]

- If, however, we allow **individual class variances**, $\sigma_0 \neq \sigma_1$, we have

$$0 < \frac{-(x-\mu_1)^2}{2\sigma_1^2} - \frac{-(x-\mu_0)^2}{2\sigma_0^2} + \theta_0$$

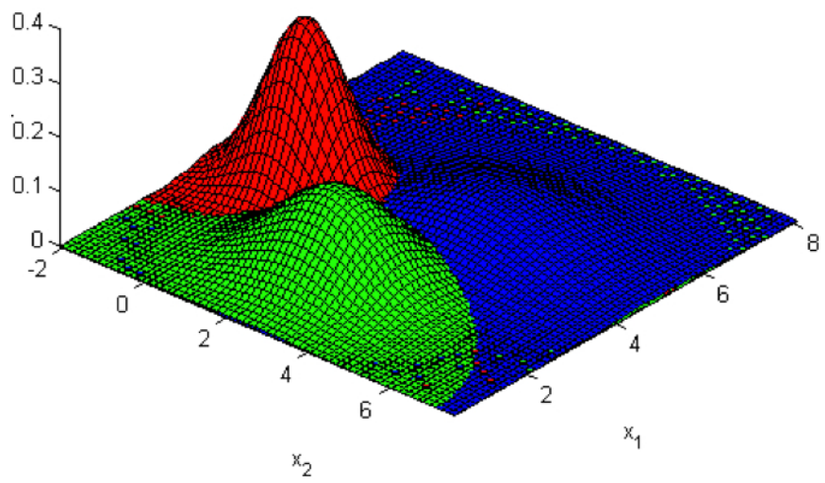
Bayes Classifier

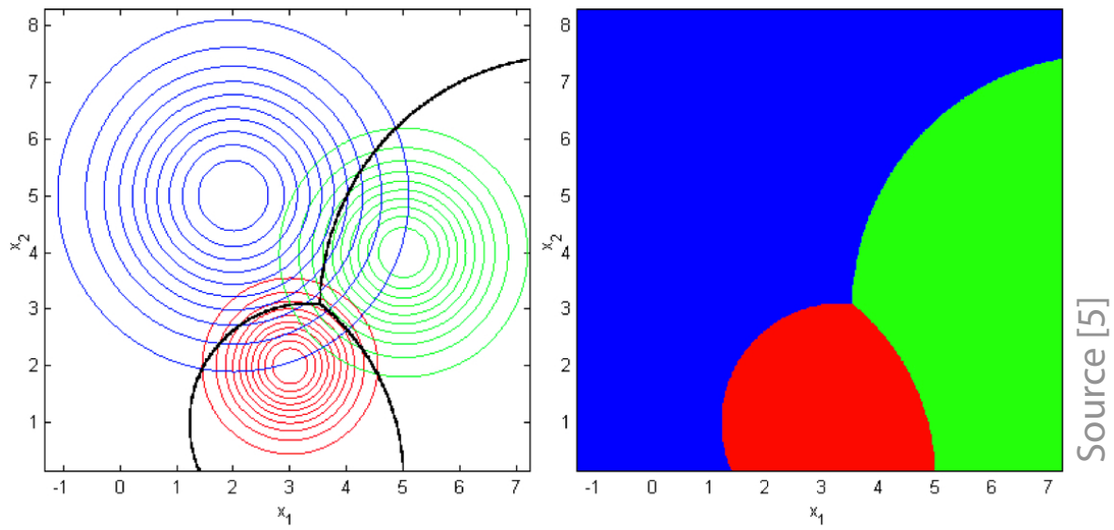
Decision Boundary

- which is a **quadratic function** in x of the form

$$0 \;<\; \mathbf{x}^T \Theta\, \mathbf{x} + \boldsymbol{\theta}^T \mathbf{x} + \theta_0$$

This decision rule induces a **quadratic decision boundary**





Source [5]

- Thus, in general, the Bayes classifier is a **quadratic classifier**

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Discussion Naive Bayes Classifier

- The decoupling of the class-conditional feature distributions means that each distribution can be independently estimated as a **one dimensional distribution**
- Simple and **much easier** than estimating high-dimensional distributions
- Due to the independence assumption, the Naive Bayes classifier will fail to produce good estimates for the correct class probabilities
- However, as long as the **correct class is more probable than any other class**, it will predict the correct class (in other words, the Naive Bayes classifier will make the **correct MAP decision**). This is even true if the probability estimates are grossly inaccurate
- This is why the Naive Bayes classifier is surprisingly useful in practice. It is a **popular baseline** for comparisons with other methods

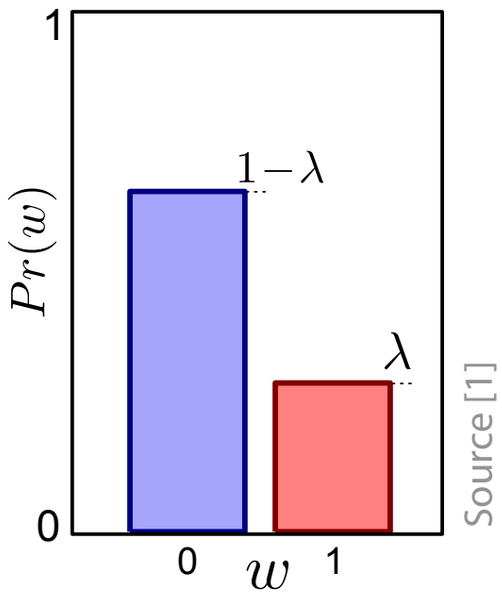
Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Logistic Regression

- Logistic Regression is a **discriminative** classifier
- The goal is to **directly** model the **posterior probability distribution** $p(\mathbf{w}|\mathbf{x})$ over a discrete world state $w \in \{1,...,K\}$ given data $\mathbf{x}$
- Let's consider a binary classification problem $w \in \{0,1\}$
- The model that we choose for the posterior probability is a **Bernoulli distribution**

$$p(w|\mathbf{x}) \;=\; \mathrm{Bern}_w(\lambda)$$

- The Bernoulli distribution has **parameter** λ, which determines the success probability $p(w=1)=\lambda$
- We now have to **estimate** λ **using data** $\mathbf{x}$ such that the constraint $0 \leq \lambda \leq 1$ is obeyed



Source [1]

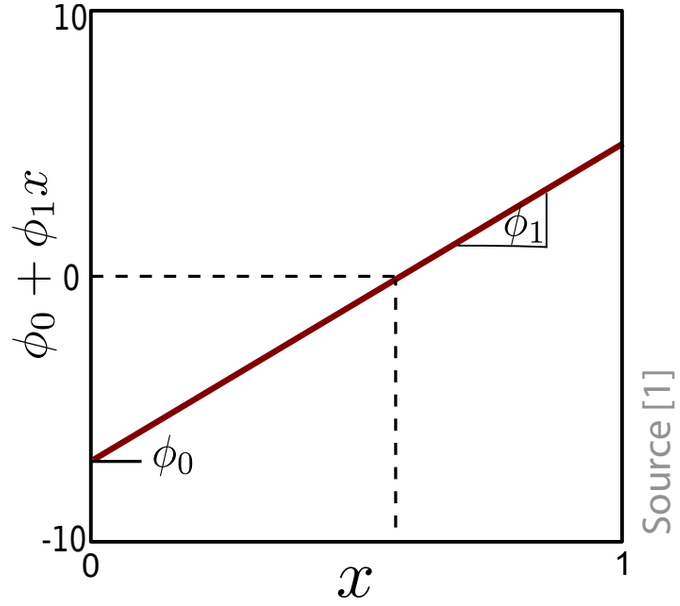
Logistic Regression

Model

- **First**, to model the probability λ we introduce the **linear predictor function**

$$a = \phi_0 + \phi_1 \, x$$

- The term is usually called **activation**
- The function has parameters $\{\phi_0, \phi_1\}$



Source [1]

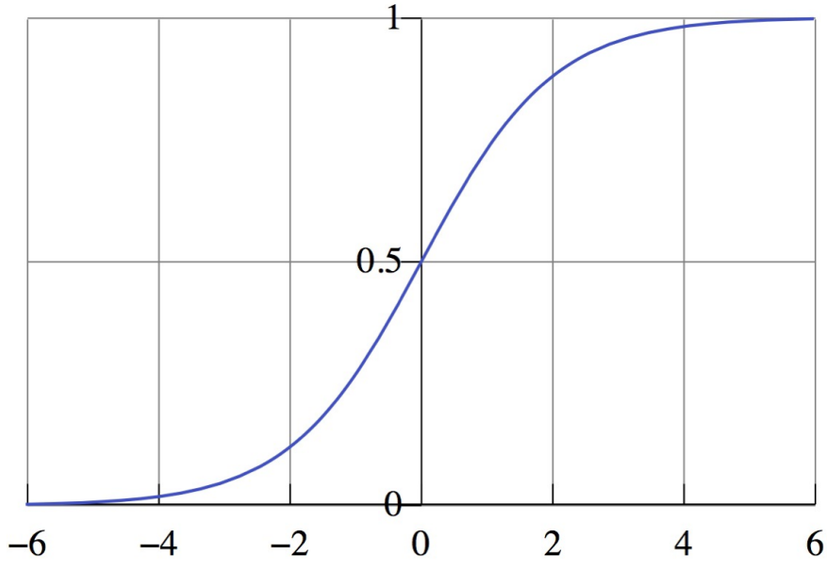
- Let us find a **more compact** notation for higher dimensions
  - Attach the y-intercept ϕ_0 to the start of the parameter vector $\boldsymbol{\phi} \leftarrow [\phi_0 \quad \boldsymbol{\phi}^T]^T$
  - Attach 1 to the start of the data vector $\mathbf{x} \leftarrow [1 \quad \mathbf{x}^T]^T$
- The activation can now be written as

$$a = \boldsymbol{\phi}^T \, \mathbf{x}$$

Model

- **Second**, we pass this function through the **logistic sigmoid function** $\mathrm{sig}(.)$ that maps the range $[-\infty..\infty]$ from the linear predictor to $[0..1]$

$$\mathrm{sig}(a) = \frac{1}{1 + \exp(-a)}$$



- The final model then becomes

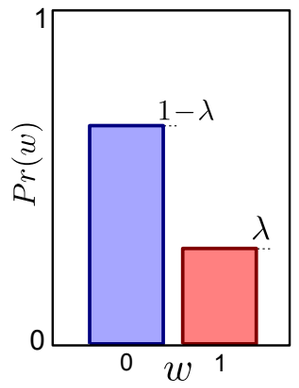
$$\begin{aligned} p(w|\mathbf{x}) &= \mathrm{Bern}_w(\lambda) \\ &= \mathrm{Bern}_w(\mathrm{sig}(a)) \\ &= \mathrm{Bern}_w\left(\frac{1}{1 + \exp(-\boldsymbol{\phi}^T \mathbf{x})}\right) \end{aligned}$$

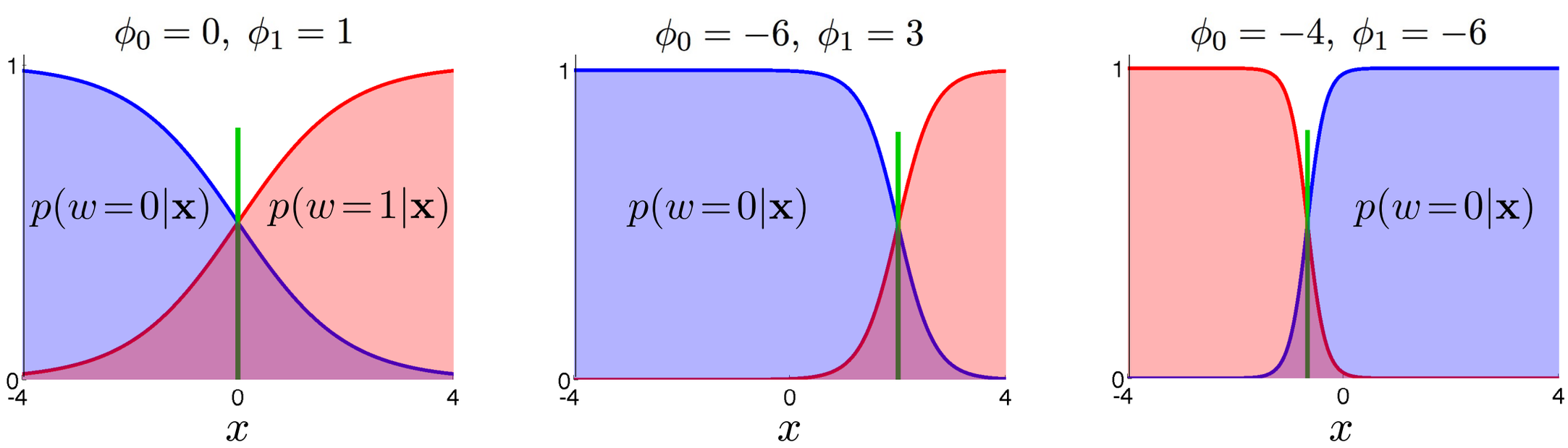
Logistic Regression

Model

- Let us **plot the model** in 1D for some values of ϕ_0 and ϕ_1

$$p(w|\mathbf{x}) = \text{Bern}_w\left(\frac{1}{1 + \exp(-\boldsymbol{\phi}^T \mathbf{x})}\right)$$





- Green line: **decision boundary**

Logistic Regression

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Learning

- For learning, we **fit the parameters** $\{\phi_0, \phi_1\}$ in a **maximum likelihood sense** by maximizing the conditional data likelihood ("goodness of fit") over the N paired training samples $(\mathbf{x}_1, w_1),\ (\mathbf{x}_2, w_2),\ \cdots\ , (\mathbf{x}_N, w_N)$
- The conditional data likelihood is the probability $p(w|\mathbf{x})$ of the (labelled) values w in the training set, conditioned on their corresponding $\mathbf{x}$ values
- Let's do this for our Bernoulli model: for a single datum, we have

$$\left.\begin{aligned} p(w=1|\mathbf{x}) &= \lambda \\ p(w=0|\mathbf{x}) &= 1-\lambda \end{aligned}\right\} \quad p(w|\mathbf{x}) = \lambda^w (1-\lambda)^{1-w}$$

- For the entire training set, let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N]$ and $\mathbf{w} = [w_1, w_2, ..., w_N]^T$, assuming independence of the pairs and applying chain rule, we obtain

$$p(\mathbf{w}|\mathbf{X}) \quad = \quad \prod_{i=1}^{N} \lambda^{w_i} (1-\lambda)^{1-w_i}$$

Logistic Regression

Learning

- Substituting the model yields

$$p(\mathbf{w}|\mathbf{X}) = \prod_{i=1}^{N} \left(\frac{1}{1 + \exp(-\boldsymbol{\phi}^T \mathbf{x}_i)} \right)^{w_i} \left(\frac{\exp(-\boldsymbol{\phi}^T \mathbf{x}_i)}{1 + \exp(-\boldsymbol{\phi}^T \mathbf{x}_i)} \right)^{1-w_i}$$

- In order to maximize this expression, it is simpler to **maximize its logarithm** L. The logarithm is a monotonic transformation that **does not change the position** of the maximum

$$L = \sum_{i=1}^{N} w_i \log \left(\frac{1}{1 + \exp(-\boldsymbol{\phi}^T \mathbf{x}_i)} \right) + \sum_{i=1}^{N} (1 - w_i) \log \left(\frac{\exp(-\boldsymbol{\phi}^T \mathbf{x}_i)}{1 + \exp(-\boldsymbol{\phi}^T \mathbf{x}_i)} \right)$$

- Finally, we set the derivate w.r.t. the parameter ϕ to zero and solve for ϕ

$$\frac{\partial L}{\partial \boldsymbol{\phi}} = -\sum_{i=1}^{N} \left(\frac{1}{1 + \exp(-\boldsymbol{\phi}^T \mathbf{x}_i)} - w_i \right) \mathbf{x}_i = -\sum_{i=1}^{N} (\text{sig}(a_i) - w_i) \mathbf{x}_i \stackrel{!}{=} 0$$

Logistic Regression

Learning

- Unfortunately, there is **no closed-form solution** for the parameters ϕ and we must rely on **nonlinear optimization** to find the maximum
- Optimization techniques start with an initial estimate of the solution, then iteratively improve it until no more progress can be made (e.g. by following the gradient). Here we can use **Newton's method**
- What about local maxima? No problem. The log likelihood for logistic regression is a **concave function of** ϕ. Concave/convex functions have no multiple maxima/minima and gradient-based methods are guaranteed to find the **global** optimum. This can be seen from the Hessian: a negative weighted sum of outer products is negative definite for all ϕ

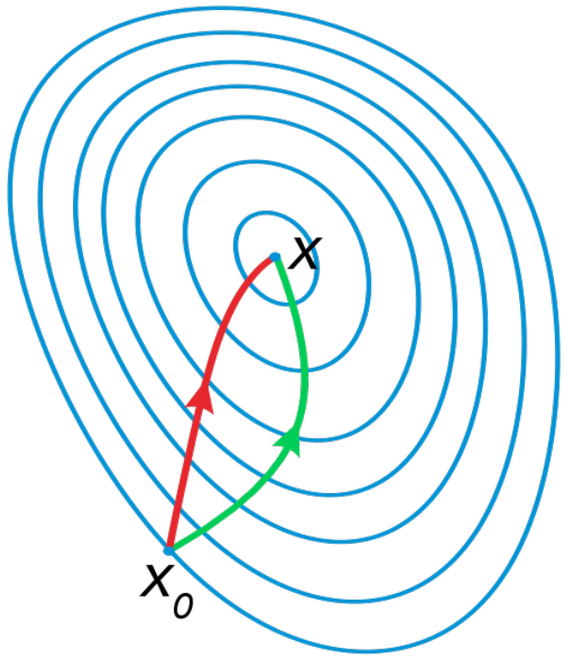
$$\frac{\partial^2 L}{\partial \boldsymbol{\phi}^2} = -\sum_{i=1}^{N} \text{sig}(a_i)(1-\text{sig}(a_i))\,\mathbf{x}_i\mathbf{x}_i^T$$

- After optimization, we have a **best parameters estimate** ϕ^*

Logistic Regression

Newton's Method

- In optimization, Newton's method finds **stationary points** of differentiable functions $f(.)$, which are the zeros of the first derivative, that may correspond to a **minimum** or **maximum** of $f(.)$
- The algorithm attempts to **iteratively** construct a sequence from an initial guess x_0 that converges towards x^* such that $f'(x^*) = 0$. This x^* is called a stationary point of $f(.)$



- Newton's method evaluates the **Hessians** (2nd derivatives) and **gradients** (1st derivatives) of the function, i.e. function is locally approximated by a quadratic
- Many more powerful algorithms exist: techniques for high dimensions, presence of constraints (equality and inequality), multi-modality, non-differential functions, etc.

Logistic Regression

Inference and Decision

- Once learning is done, we make **inference** by simply substituting a new observation $\mathbf{x}$ into our model to retrieve the posterior distribution over the state

$$p(w|\mathbf{x}) \;=\; \mathrm{Bern}_w \left(\frac{1}{1 \,+\, \exp(-\boldsymbol{\phi}^{*T}\, \mathbf{x})} \right)$$

- The **decision** consists in assigning the class to $\mathbf{x}$ whose posterior probability is maximal. Formally, we assign the label $w = 1$ if the following condition holds:

$$\frac{p(w=1|\mathbf{x})}{p(w=0|\mathbf{x})} > 1$$

From this we can derive the **decision boundary**

Logistic Regression

Decision Boundary

- We substitute the logistic regression model

$$p(w=1|\mathbf{x}) = \lambda = \frac{1}{1+\exp(-\boldsymbol{\phi}^T\mathbf{x})}$$

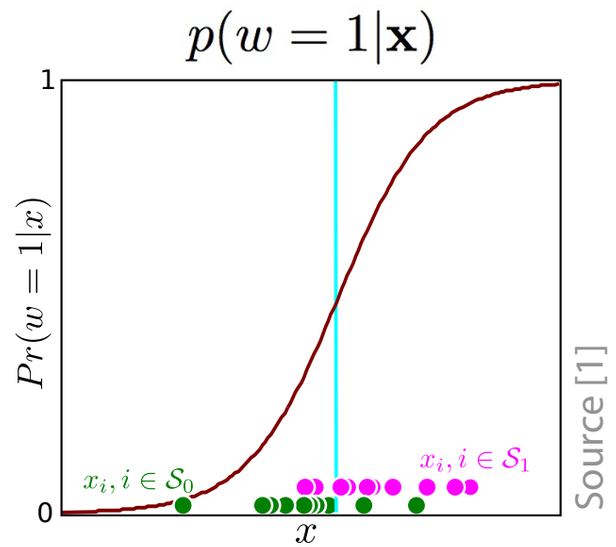
into $\frac{p(w=1|\mathbf{x})}{p(w=0|\mathbf{x})} > 1$ and obtain

$$\frac{\lambda}{1-\lambda} = \frac{\frac{1}{1+\exp(-\boldsymbol{\phi}^T\mathbf{x}_i)}}{\frac{\exp(-\boldsymbol{\phi}^T\mathbf{x}_i)}{1+\exp(-\boldsymbol{\phi}^T\mathbf{x}_i)}} = \exp(\boldsymbol{\phi}^T\mathbf{x}_i) > 1$$

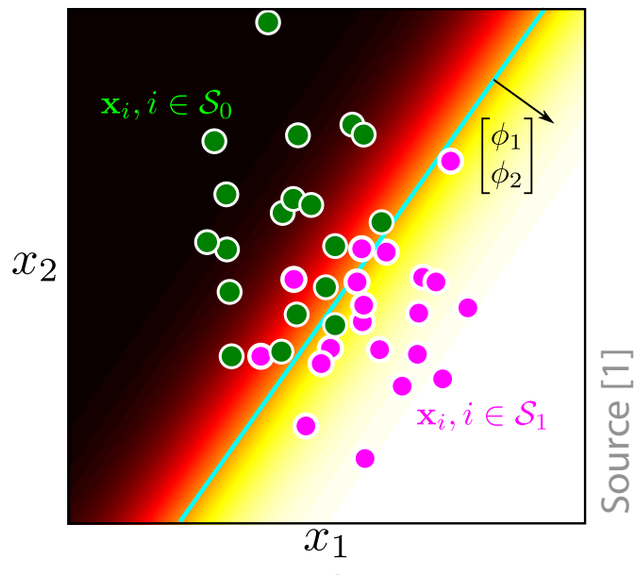
- Taking the logarithm of both sides yields

$$\boldsymbol{\phi}^T\mathbf{x}_i > 0$$

which is a **linear decision rule**



1D data



2D data

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Summary Bayes Classifier and Logistic Regression

- The problem of classification is to learn an **unknown function** $y = f(\mathbf{x})$ that maps input data to class assignments
- Bayes classifier and Logistic Regression are two methods for this function approximation task that compute a **probability distribution** $p(\mathbf{w}|\mathbf{x})$ over the world state given the input data
- Bayes classifier is a **generative classifier** that determines the likelihood $p(\mathbf{x}|\mathbf{w})$ (in our notation the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$ for each class) and the prior class probabilities $p(\mathbf{w})$ (or $p(\mathcal{C}_k)$). This amounts to learning the joint distribution. Then it uses Bayes' rule to compute the sought posterior distribution $p(\mathbf{w}|\mathbf{x})$ (or $p(\mathcal{C}_k|\mathbf{x})$)
- Learning Bayes classifiers typically requires an unrealistic number of training examples. The **Naive Bayes classifier** assumes all features in $\mathbf{x}$ are conditionally independent given $\mathcal{C}_k$. This assumption dramatically reduces the number of parameters that must be estimated to learn the classifier

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Summary Bayes Classifier and Logistic Regression

- The model is called **generative** because we can view the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$ as describing how to generate synthetic data points $\mathbf{x}$ conditioned on the target attribute $\mathcal{C}_k$ by sampling
- Bayes classifier is a **quadratic** classifier. Under the special assumption of equal variance among classes, decision boundaries are **linear**
- We have exemplified learning with a **Gaussian likelihood model** (very common for real-valued data) and **Bernoulli priors**. This is done by estimating the model parameters from the training set in a ML sense

- Logistic Regression learns $p(\mathbf{x}|\mathcal{C}_k)$ **directly** from the data
- The classifier uses a model based on a **linear activation term** passed through the **logistic sigmoid function**

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Summary Bayes Classifier and Logistic Regression

- Learning consists in estimating the parameters of this model from the training set in an ML sense
- By setting the derivative of the log likelihood w.r.t. parameters to zero, we obtain an equation system which has **no closed-form solution.** We must rely on **nonlinear optimization** to find the best parameters
- Logistic Regression is a **discriminative** classifier because we can view the distribution $p(\mathbf{x}|\mathcal{C}_k)$ as directly discriminating the value of the target $\mathcal{C}_k$ for any given instance $\mathbf{x}$
- Logistic Regression is a **linear classifier**
- Both classifiers are **simple** and **popular** (especially Naive Bayes). It is good practice to use them as **baselines** in more complex classification tasks

Sources and Further Readings

The introduction section mainly follows the books by Russell and Norvig [2] (chapter 18) and Prince [1] (chapters 6 and 9). The sections on Logistic regression, Naive Bayes mainly follow [1]. Small bits are also taken from [3] and [4]

[1] S.J.D. Prince, "Computer vision: models, learning and inference", Cambridge University Press, 2012. See www.computervisionmodels.com

[2] S. Russell, P. Norvig, "Artificial Intelligence: A Modern Approach", 3rd edition, Prentice Hall, 2009. See http://aima.cs.berkeley.edu

[3] C.M. Bischop, "Pattern Recognition and Machine Learning", Springer, 2nd ed., 2007. See http://research.microsoft.com/en-us/um/people/cmbishop/prml

[4] T. Hastie, R. Tibshirani, J. Friedman, "The Elements of Statistical Learning: Data Mining, Inference, and Prediction", 2nd Edition, Springer, 2009

[5] R. Gutierrez-Osuna, "Pattern Recognition, Lecture 5: Quadratic Classifiers", Lecture Notes, Texas A&M University, 2011

Supervised Learning

To be continued in Supervised Learning, part 2/3