Lecture Pattern Analysis

Part 01: Vocabulary, Probabilities, and Sampling

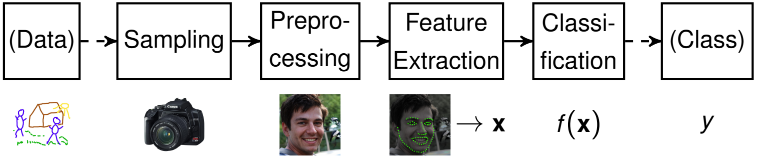
Christian Riess
IT Security Infrastructures Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
April 28, 2022

Introduction

- We require some vocabulary for our communication about Machine Learning (ML)
- We also require a formal framework to discuss ML algorithms on a scientific basis
- Arguably the most widely used framework is probability theory
- We will also introduce our first algorithm, namely how to sample from a Probability Density Function (PDF)

Pattern Recognition Recap and Classification Vocabulary

- Remember the steps of the classical pattern recognition pipeline:



- Fundamental ML assumption: good feature representations map similar objects to similar features
- Classifier training is virtually always **supervised**, i.e. a training sample is a tupel $(\mathbf{x_i}, y_i)$ (cf. lecture "Pattern Recognition")
- **Unsupervised** ML works without labels, i.e., it only operates on inputs $(\mathbf{x_i})$ Hence, unsupervised ML only works on the distribution of the features
- Fashionable variants are semi-supervised ML (some data has labels), self-supervised ML (auto-generate surrogate labels)

Recap on Probability Vocabulary

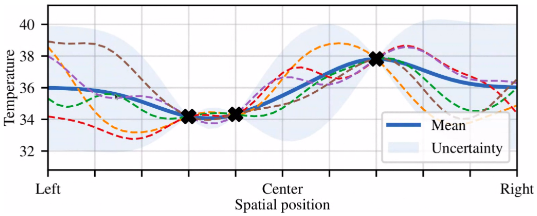
- We oftentimes operate with random variables X, Y
- Important vocabulary and equations are:

Joint distribution	$p(X, Y)$
Conditional distribution of X given Y	$p(X \vert Y)$
Sum rule / marginalization over Y	$p(X) = \sum_Y p(X, Y)$
Product rule	$p(X, Y) = p(Y \vert X) \cdot p(X)$
Bayes rule	$p(Y \vert X) = \frac{p(X \vert Y) \cdot p(Y)}{p(X)}$
Bayes rule in the language of ML	$\text{posterior} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}}$

- Please browse the book by Bishop, Sec. 1.2.3, to refresh your mind if necessary!

Sampling from a PDF

- Oftentimes, it is necessary to draw samples from a PDF
- Example:
  - Logistic Regression fits a single regression curve to the data (cf. PR)
  - Bayesian Logistic Regression fits a distribution of curves



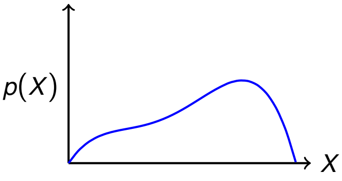
    The distribution is narrow at observations (crosses), and wider otherwise
  - Sample curves from the distribution to obtain its spread (“uncertainty”)
- Special PDFs like Gaussians have closed-form solutions for sampling
- We look now at a sampling method that works on **arbitrary PDFs**

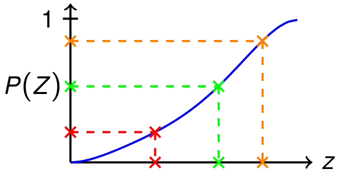
Idea of the Sampling Algorithm

- The key idea is to use the cumulative density function (CDF) $P(z)$ of $p(X)$,

$$P(z) = \int_{-\infty}^{z} p(X)\mathrm{d}X \qquad (1)$$

- A sample uniformly drawn from the CDF y-axis intersects $P(z)$ at location z
- This z position is our random draw from $p(x)$:





Sampling Algorithm

- Discretize the domain of the PDF $p(X)$
- Linearize $p(X)$ if it is multivariate
- Calculate the cumulative density function $P(z)$ of $p(X)$, the range of that CDF must be between 0 to 1
- Draw a uniformly distributed number u between 0 and 1
- The sample from the PDF is

$$z^* = \operatorname*{argmin}_z u \leq P(z) \tag{2}$$

- Additional reference / strictly optional: Bishop Chapter 11 contains many more (also advanced) sampling strategies

Lecture Pattern Analysis

Part 02: Non-Parametric Density Estimation

Christian Riess
IT Security Infrastructures Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg

April 28, 2022

Introduction

- Density Estimation = create a PDF from a set of samples
- The lecture Pattern Recognition introduces parametric density estimation:
  - Here, a parametric model (e.g., a Gaussian) is fitted to the data
  - Maximum Likelihood (ML) estimator:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\mathbf{x}_1, \ldots, \mathbf{x}_N | \boldsymbol{\theta}) \tag{1}$$

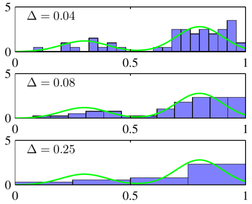
  - Maximum a Posteriori (MAP) estimator:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\boldsymbol{\theta} | \mathbf{x}_1, \ldots, \mathbf{x}_N) \overset{\text{Bayes}}{=} \frac{p(\mathbf{x}_1, \ldots, \mathbf{x}_N | \boldsymbol{\theta}) \cdot p(\boldsymbol{\theta})}{p(\mathbf{x}_1, \ldots, \mathbf{x}_N)} \tag{2}$$

  - Browse the PR slides if you like to know more
- Parametric density estimators require a good function representation
- Non-parametric density estimators can operate on arbitrary distributions

Non-Parametric Density Estimation: Histograms

- Non-parametric estimators do not use functions with a limited set of parameters
- A **simple non-parametric baseline** is to create a histogram of samples[1]
  - The number of bins is important to obtain a good fit



  - Pro: Good for a quick visualization
  - Pro: “Cheap” for many samples in low-dimensional space
  - Con: Discontinuities at bin boundaries
  - Con: Scales poorly to high dimensions (cf. curse of dimensionality later)

[1] See introduction of Bishop Sec. 2.5

Improving on the Histogram Approach

- A kernel-based method and a nearest-neighbor method are slightly better
- Both variants share their mathematical framework:
  - Let $p(\mathbf{x})$ be a PDF in D-dim. space, and R a small region around $\mathbf{x}$
    $\rightarrow$ The probability mass in R is $p = \int\limits_R p(\mathbf{x})\,\mathrm{d}\mathbf{x}$
  - Assumption 1: in R are many points $\rightarrow$ p is a relative frequency,

$$p = \frac{\#\text{ points in } R}{\text{total }\#\text{ of points}} = \frac{K}{N} \tag{3}$$

  - Assumption 2: R is small enough s.t. $p(\mathbf{x})$ is approximately constant,

$$p = \int\limits_R p(\mathbf{x})\,\mathrm{d}\mathbf{x} = p(\mathbf{x}) \int\limits_R \mathrm{d}\mathbf{x} = p(\mathbf{x}) \cdot V \tag{4}$$

  - Both assumptions together are slightly contradictory, but they yield

$$p(\mathbf{x}) = \frac{K}{N \cdot V} = \frac{\#\text{ points in } R}{\text{total }\#\text{ of points} \cdot \text{Volume of } R} \tag{5}$$

Kernel-based DE: Parzen Window Estimator (1/2)

- The Parzen window estimator fixes V and leaves K/N variable[2]
- D-dimensional Parzen window kernel function (a.k.a. “box kernel”):

$$k(\mathbf{u}) = \begin{cases} 1 & \text{if } |u_i| \leq \frac{1}{2} \quad \forall i = 1, \ldots, D \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

- Calculate K with this kernel function:

$$K(\mathbf{x}) = \sum_{i=1}^{N} k\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) \tag{7}$$

where h is a scaling factor that adjusts the box size

- Hence, the whole density is

$$p(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{h^D} k\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) \tag{8}$$

[2] See Bishop Sec. 2.5.1

Kernel-based DE: Parzen Window Estimator (2/2)

- The kernel removes much of the discretization error of the fixed-distance histogram bins, but it still leads to blocky estimates
- Replacing the box kernel by a Gauss kernel further smooths the result,

$$p(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{2\pi h^2}^{D/2} \cdot \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_i\|_2^2}{2h^2}\right) \quad , \tag{9}$$

where h^2 is the standard deviation of the Gaussian

- Mathematically, also any other kernel is possible if these conditions hold:

$$k(\mathbf{u}) \geq 0 \tag{10}$$

$$\int k(\mathbf{u})\,\mathrm{d}\mathbf{u} = 1 \tag{11}$$

K-Nearest Neighbors (k-NN) Density Estimation

- Recall our derived equation for estimating the density

$$p(\mathbf{x}) = \frac{K}{N \cdot V} = \frac{\text{\# points in } R}{\text{total \# of points} \cdot \text{Volume of } R} \tag{12}$$

- The Parzen window estimator fixes V, and K varies
- The k-Nearest Neighbors estimator fixes K, and V varies
- k-NN calculates V from the distance of the K nearest neighbors[3]
- Note that both the Parzen window estimator and the k-NN estimator are "non-parametric", but they are not free of parameters
  - The kernel scaling h and the number of neighbors k are **hyper-parameters**, i.e., some form of prior knowledge to guide the model creation
  - The model parameters are the samples themselves. Both estimators need to store all samples, which is why they are also called **memory methods**

[3] See Bishop Sec. 5.2.2

First Glance at the Model Selection Problem

- Optimizing the hyperparameters is also called **Model Selection Problem**
- Supervised methods use cross validation (CV) via Maximum Likelihood (ML)
- We can use CV to optimize the DE hyperparameters by using the **prediction of held-out samples** as objective function:
    - Split the data into J folds:
      $S^j_{\text{train}} = S \setminus \left\{ x_{\lfloor \frac{N}{J} \rfloor \cdot j}, \ldots x_{\lfloor \frac{N}{J} \rfloor \cdot (j+1) - 1} \right\}$,
      $S^j_{\text{test}} = S \setminus S^j_{\text{train}}$
    - Let $\boldsymbol{\alpha}$ be the unknown hyperparameters, and
      let $p_j(\mathbf{x}|\boldsymbol{\alpha})$ be the density estimate for samples S^j_{train} on hyperparams $\boldsymbol{\alpha}$
    - Then, the ML estimate is

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \prod_{j=1}^{J} \prod_{\mathbf{x} \in S^j_{\text{test}}} p_j(\mathbf{x}|\boldsymbol{\alpha}) \tag{13}$$

    - In practice, take the logarithm (“log likelihood”) to mitigate numerical issues
      $\rightarrow$ the product becomes a sum

Lecture Pattern Analysis

Part 03: Bias and Variance

Christian Riess
IT Security Infrastructures Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
April 28, 2022

Introduction

- We search in the exercise for good kernel parameters via cross validation
- This model selection task touches the question of **generalization** to new data:
  - Too large kernel: smears out the structure of the training data, but covers new samples
  - Too small kernel: closely follows the training data, but might miss new samples
  - "Right" kernel size: represents the structure of the training data and also covers new samples
    (to the extent possible with the given method and data)
- This is an instance of the **bias-variance tradeoff**[1]

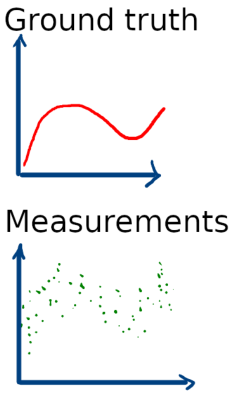
[1] See PR lecture or Hastie/Tibshirani/Friedman Sec. 7-7.3 if more details are desired

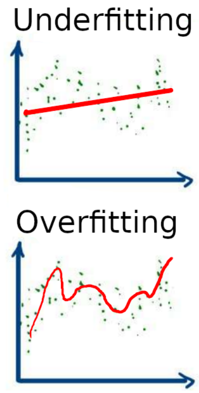
Bias and Variance in Regression

- **Bias** is the square of the *average deviation* of an estimator from the ground truth
- **Variance** denotes is the *variance of the estimates*, i.e., the expected squared deviation from the estimated mean[2]
- Informal interpretation:
  - High bias indicates **model undercomplexity**: we obtain a poor fit to the data
  - High variance indicates **model overcomplexity**: the fit also models not just the structure of the data, but also its noise
- Higher model complexity (= more model parameters) tends to lower bias and higher variance
- We will usually not be able to get bias and variance simultaneously to 0
- Regularization increases bias and lowers variance

[2] See Hastie/Tibshirani/Friedman Sec. 7.3 Eqn. (7.9) for a detailed derivation

Sketches for Model Undercomplexity and Overcomplexity





- Note that this example implicitly contains a smoothness assumption
- It does not claim that there is a universally best fit on arbitrary input distributions (because of the No-Free-Lunch Theorem)

Transferring Bias and Variance to our Density Estimators

- Our kernel framework can directly replicate these investigations by retargeting our kernels to regression or classification:
- Regression:
  - Estimate $f(\mathbf{x})$ at position $\mathbf{x}$ as a kernel-weighted sum of the neighbors or
  - as a k-NN mean of k neighbors
- Classification:
  - Estimate for classes c_1 and c_2 individual densities, evaluate $p_{c_1}(\mathbf{x})$ and $p_{c_2}(\mathbf{x})$, and select the class with higher probability or
  - Select the majority class within k nearest neighbors
- We will then observe that
  - Larger kernel support / larger k increases bias and lowers variance
  - Smaller kernel support / smaller k lowers bias and increases variance
- Analogously, we can use the notion of bias/variance also in our exercise task of unsupervised density estimation

Bonus Slide: Current Research

- Just if you are curious — this is **strictly voluntary**:

- Here is a video on the recent ICML paper “Maximum Likelihood With Bias-Corrected Calibration is Hard-To-Beat at Label Shift Adaptation” by Avanti Shrikumar from Stanford University:
  `https://www.youtube.com/watch?v=ZBXjE9QTruE`

- It adapts the prior of a deep learning model without retraining the model, using tools from our lecture
- I find the talk also generally instructive as an academic presentation

- Enjoy!