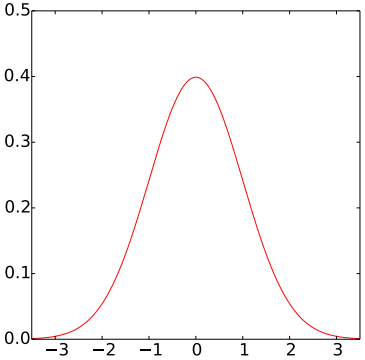
Machine Learning 2.09: Density Estimation

Tom S. F. Haines
T.S.F.Haines@bath.ac.uk

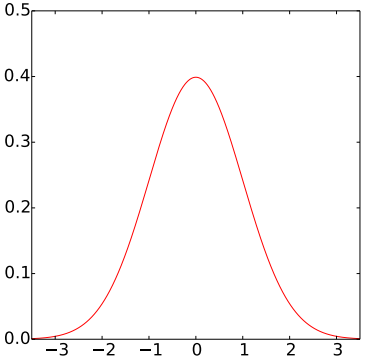
UNIVERSITY OF BATH

Density estimation?



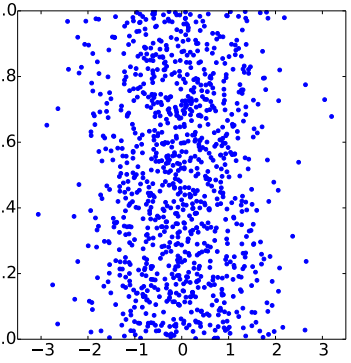
PDF

Density estimation?



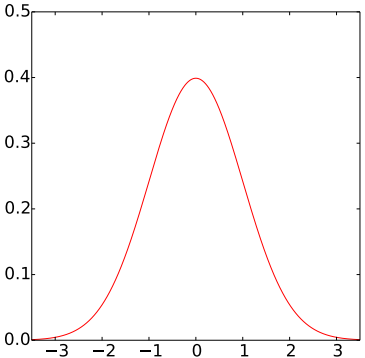
PDF

Draw/Sample/Simulate



Samples

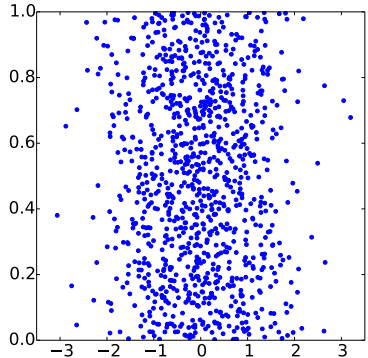
Density estimation?



PDF

Draw/Sample/Simulate

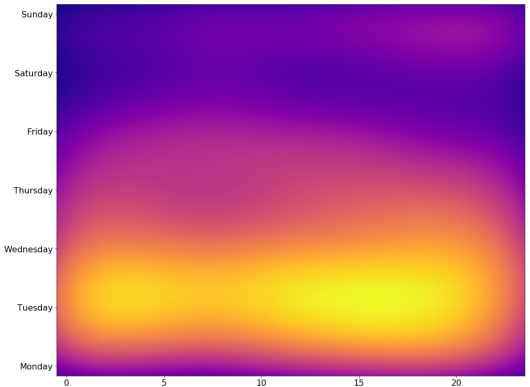
Density estimation



Samples

Real data

- Real distributions are complex!



Crime in Vancouver from 2003 to 2017, hour vs. day of week.

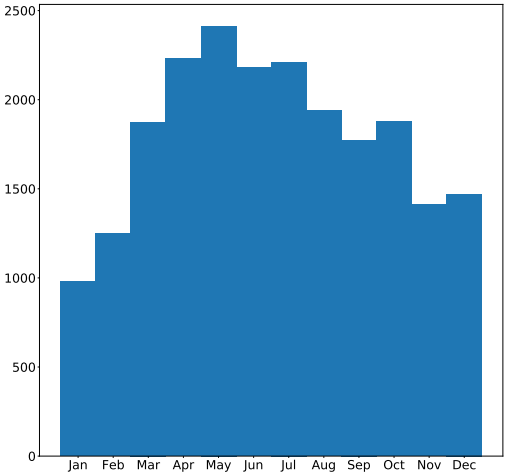
Related

- Might be a density estimate:
  - Clustering
  - Manifold learning
- Not a density estimate:
  - Discrete data
  - Generative models that use problem structure

(convention)

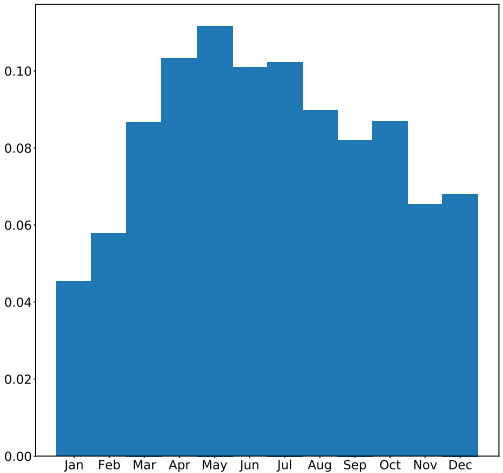
Histograms

- Histograms are a density estimate!



House sales by month (King County, USA)

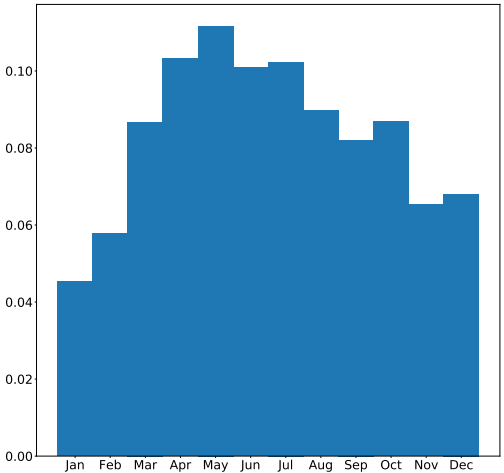
Histograms



House sales by month (King County, USA)

- Histograms are a density estimate!
  . . . if you normalise.

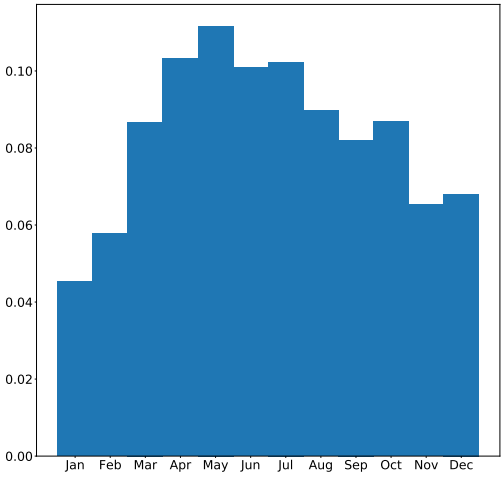
Histograms



House sales by month (King County, USA)

- Histograms are a density estimate!
  . . . if you normalise.
- Discrete: Normalise sum
- Continuous: Normalise area

Histograms



House sales by month (King County, USA)

- Histograms are a density estimate!
  . . . if you normalise.
- Discrete: Normalise sum
- Continuous: Normalise area
- Visualisation = use case
- Limitations:
  - Have to select bin size
  - Not very smooth
  - 1, 2, maybe 3, dimensions only

UNIVERSITY OF BATH

Bin count?

- Heuristics: (worst to best, probably)
  - Square root: $b = \lceil \sqrt{n} \rceil$
  - Sturges: $b = \lceil \log_2 n \rceil + 1$
  - Freedman–Diaconis: $w = \frac{2\mathrm{IQR}}{n^{1/3}}$

b = number of bins

w = bin width ($b = \lceil \frac{\mathtt{max} - \mathtt{min}}{w} \rceil$)

n = number of data points

(IQR = Interquartile range)

UNIVERSITY OF BATH

Bin count?

- Heuristics: (worst to best, probably)
  - Square root: $b = \lceil \sqrt{n} \rceil$
  - Sturges: $b = \lceil \log_2 n \rceil + 1$
  - Freedman–Diaconis: $w = \frac{2\mathtt{IQR}}{n^{1/3}}$

b = number of bins
w = bin width ($b = \lceil \frac{\mathtt{max} - \mathtt{min}}{w} \rceil$)
n = number of data points
(IQR = Interquartile range)

- Most spreadsheets use square root (copying Excel for consistency)
- numpy/matplotlib use maximum of Sturges and Freedman–Diaconis

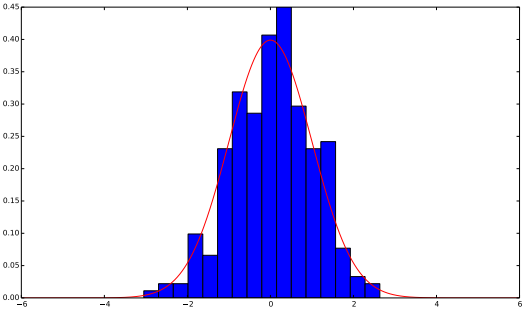
- Many more – can be smarter
- Not worth it, because...

UNIVERSITY OF BATH

Variable bin width

- Surprisingly rare

Variable bin width

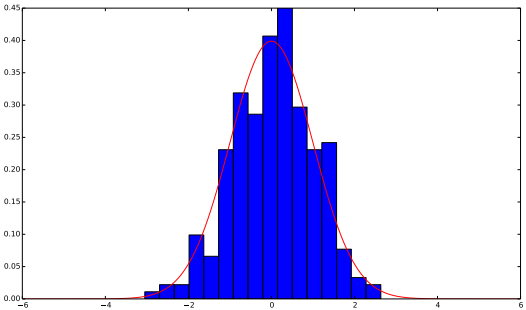


Few samples due to reduced probability mass

$\therefore$ tails have more noise

- Surprisingly rare

Variable bin width



Few samples due to reduced probability mass

$\therefore$ tails have more noise

- Surprisingly rare
- Simple approach:
  - Start with one bin
  - Keep dividing it in half
  - Stop at minimum bin size

  (random forest-ish)
- Smart approach: Bayesian blocks

UNIVERSITY OF BATH

Kernels

- Histogram = sum of rectangles (stacking bricks)

Kernels

- Histogram = sum of rectangles (stacking bricks)

Kernels

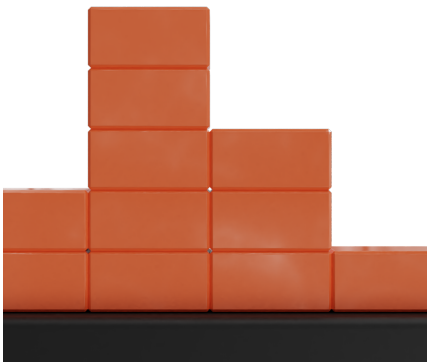
- Histogram = sum of rectangles (stacking bricks)

UNIVERSITY OF BATH

Kernels

- Histogram = sum of rectangles (stacking bricks)

Kernels



- Histogram = sum of rectangles (stacking bricks)

$$P(y) = \frac{1}{n}\sum_{i=1}^{n} \texttt{rect}\left(\frac{y}{w} - \left\lfloor \frac{x_i}{w} \right\rceil\right)$$

$$\texttt{rect}(t) = \begin{cases} 1 & -\frac{1}{2} < t < \frac{1}{2} \\ 0 & \texttt{otherwise} \end{cases}$$

(w = bin width, $\lfloor \cdot \rceil$ = round to nearest integer)

- $\texttt{rect}(t)$ = kernel (area = 1)

Kernel density estimate

KDE

- Why force kernels to be evenly spaced? – Drop $\lfloor \cdot \rceil$!

$$P(y) = \frac{1}{n} \sum_{i=1}^{n} \texttt{rect}(\frac{y - x_i}{w})$$

Kernel density estimate

KDE

- Why force kernels to be evenly spaced? – Drop $\lfloor \cdot \rceil$!

$$P(y) = \frac{1}{n} \sum_{i=1}^{n} \texttt{rect}\left(\frac{y - x_i}{w}\right)$$

Kernel density estimate

KDE

- Why force kernels to be evenly spaced? – Drop $\lfloor \cdot \rceil$!

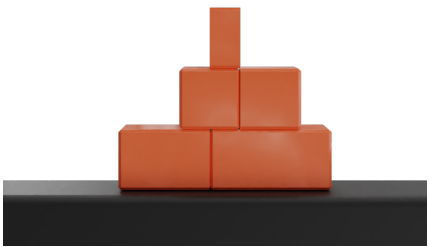
$$P(y) = \frac{1}{n} \sum_{i=1}^{n} \texttt{rect}\left(\frac{y - x_i}{w}\right)$$

Kernel density estimate

KDE

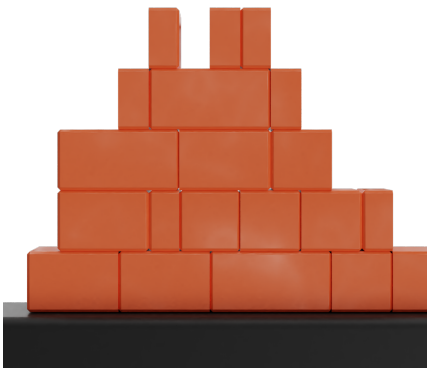
- Why force kernels to be evenly spaced? – Drop $\lfloor \cdot \rceil$!

$$P(y) = \frac{1}{n} \sum_{i=1}^{n} \texttt{rect}\left(\frac{y - x_i}{w}\right)$$



Kernel density estimate

KDE



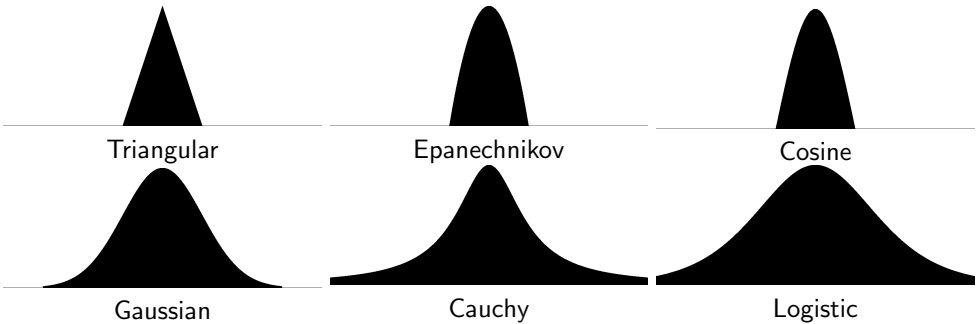
- Why force kernels to be evenly spaced? – Drop $\lfloor \cdot \rceil$!

$$P(y) = \frac{1}{n} \sum_{i=1}^{n} \texttt{rect}\left(\frac{y - x_i}{w}\right)$$

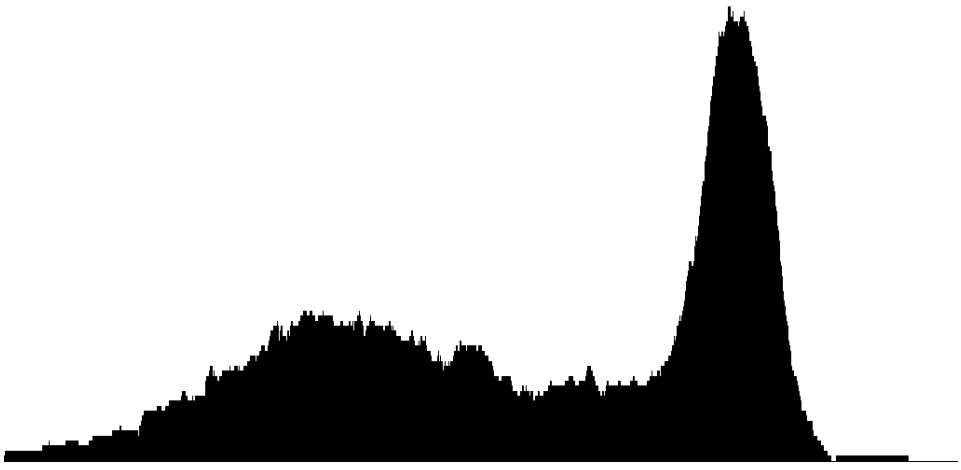
- Called a **kernel density estimate**

Rectangles are boring

- Why fix kernels to be rectangles?



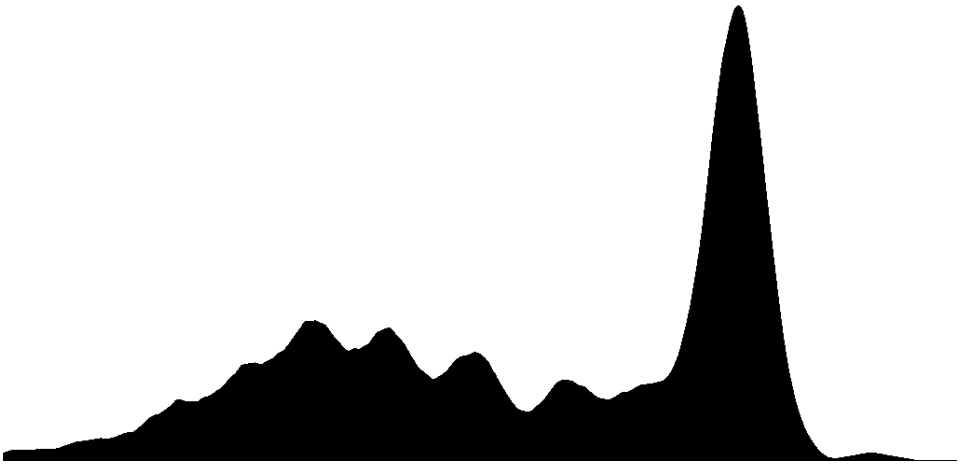
Kernels



Uniform (rect)

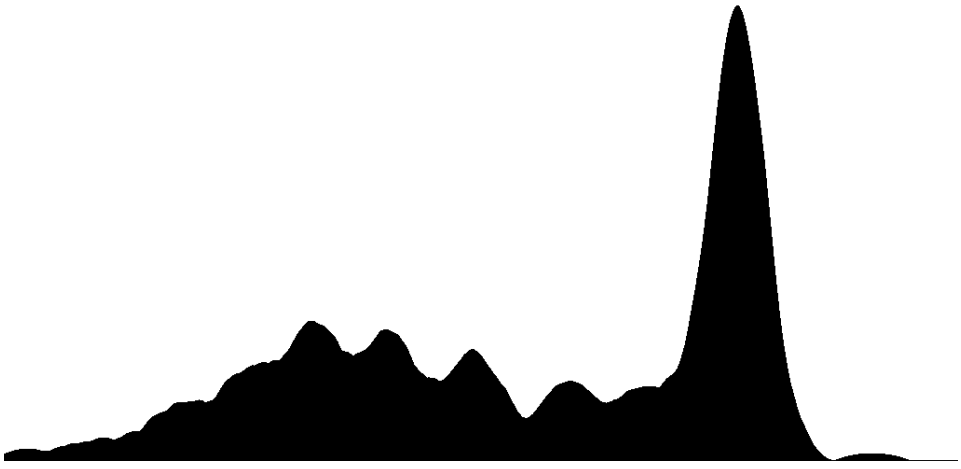
UNIVERSITY OF BATH

Kernels



Triangular

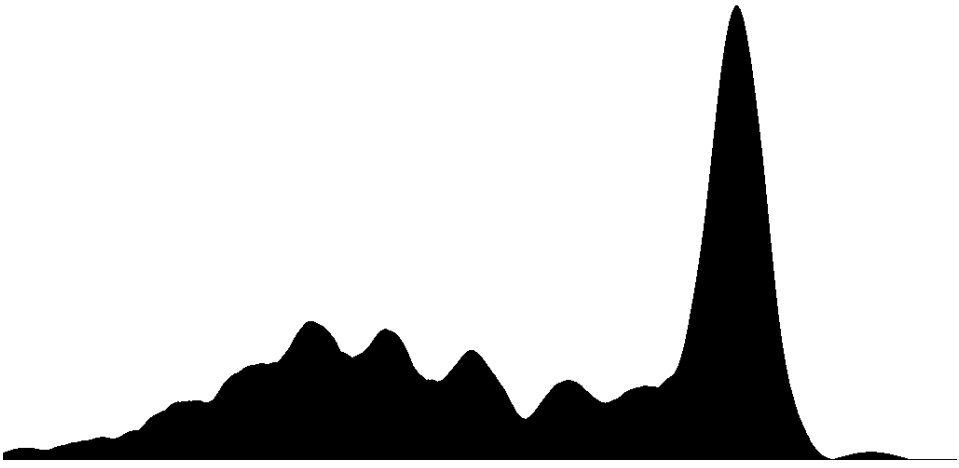
Kernels



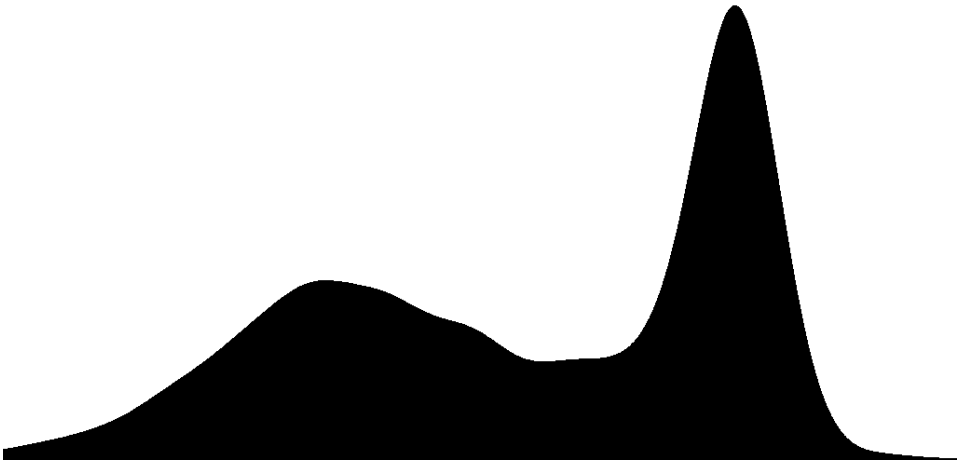
Epanechnikov

UNIVERSITY OF BATH

Kernels



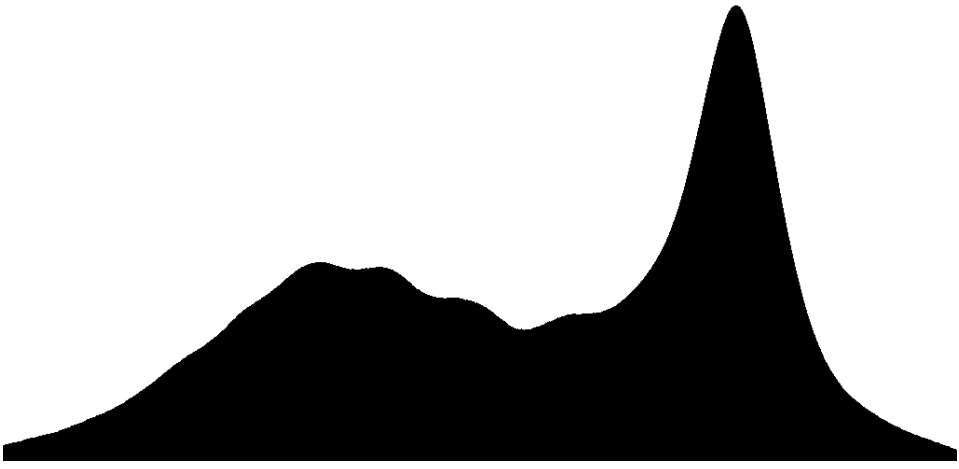
Kernels



Gaussian

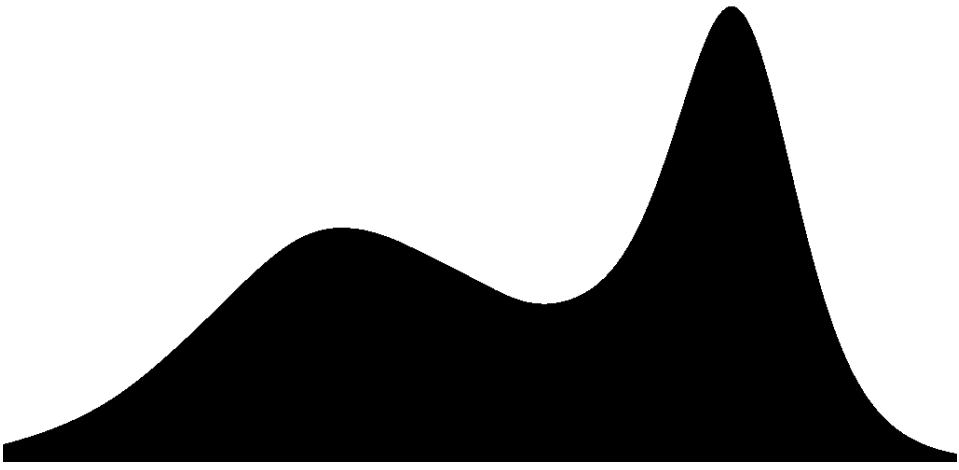
UNIVERSITY OF BATH

Kernels



Cauchy

Kernels



Logistic (best, in this case)

UNIVERSITY OF BATH

Which kernel?

- Hyperparameter optimisation...
  expensive, advantage often minimal
- Good defaults:
  Gaussian or Epanechnikov
- Implementation detail:
  - Calculating $P(x)$: $O(n)$
    Use KD-tree to accelerate
  - Some distributions have infinite domain, e.g. Gaussian
    Clip, e.g. at 3 standard deviations

Equations

$$\texttt{uniform}(t) \propto \begin{cases} 1 & t < 1 \\ 0 & \texttt{otherwise} \end{cases}$$

$$\texttt{triangular}(t) \propto \max(1 - t, 0)$$

$$\texttt{epanechnikov}(t) \propto \max(1 - t^2, 0)$$

$$\texttt{cosine}(t) \propto \cos(\frac{\pi}{2} \max(t, 1))$$

$$\texttt{gaussian}(t) \propto e^{\frac{-t^2}{2}}$$

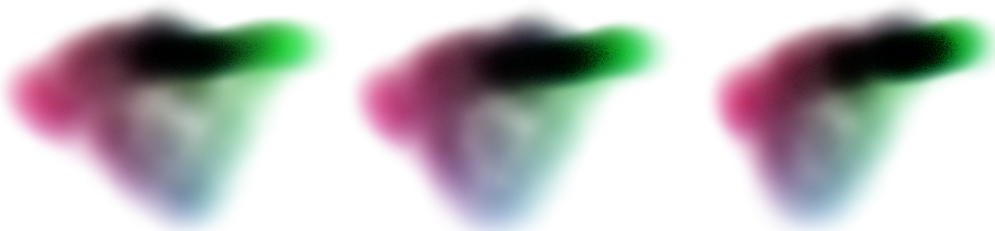
$$\texttt{cauchy}(t) \propto \frac{1}{1 + t^2}$$

$$\texttt{logistic}(t) \propto \frac{e^{-t}}{(1 + e^{-t})^2}$$

(normalisation terms omitted)

Multidimensional

- Generalises to arbitrary dimensions
  - $t = |\vec{x} - \vec{y}|$
  - Kernel normalisation gets tricky
  - Suffers curse of dimensionality, but better than most



3D, colour indicates position, darker means denser

UNIVERSITY OF BATH

Bandwidth

- Histograms have bin width...
  ...Kernel density estimates have *bandwidth* (kernel size)
- Hyperparameter to tune (there are bad heuristics, e.g. Silverman)

UNIVERSITY OF BATH

Bandwidth

- Histograms have bin width...
  ...Kernel density estimates have *bandwidth* (kernel size)
- Hyperparameter to tune (there are bad heuristics, e.g. Silverman)

- Obvious solution:
  Select bandwidth with highest data probability
  (in log space, no validation set)
- Does not work: Bandwidth $\to 0$

UNIVERSITY OF BATH

Bandwidth

- Histograms have bin width...
  ...Kernel density estimates have *bandwidth* (kernel size)
- Hyperparameter to tune (there are bad heuristics, e.g. Silverman)

- Obvious solution:
  Select bandwidth with highest data probability
  (in log space, no validation set)
- Does not work: Bandwidth $\to 0$

- Instead, do the *jackknife*:
  - Calculate probability of each exemplar *whilst excluding it from the KDE*
  - Multiply all exemplars together
    (add in log space)

  (k-fold cross-validation where $k = n$)

Simple distributions

for completeness

- You can also fit distributions, e.g.
  - Calculate mean, $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$
  - Calculate variance, $\sigma = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$
  - This fits a normal distribution: $X \sim \mathcal{N}(\mu, \sigma)$

- Works for most analytical distributions
  (multiple approaches with different answers in some cases)

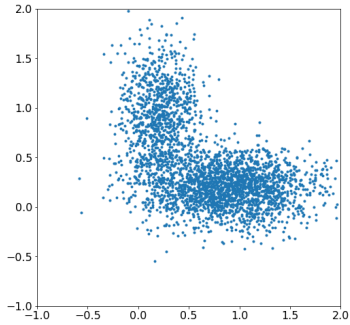
Gaussian mixture model

GMM

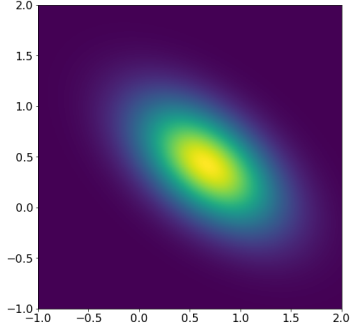
- Reminder! (Original: Lecture 12 of machine learning 1)
- Instead of one Gaussian distribution describe data with several:

$$P(x|\alpha,\mu,\Sigma) = \sum_{z=1}^{K} \alpha_z \mathcal{N}(x|\mu_z,\Sigma_z)$$

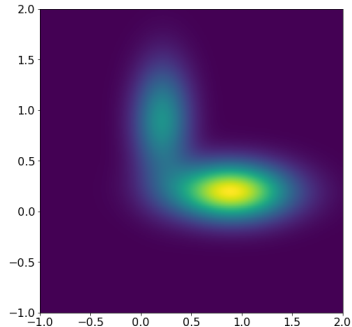
Input:



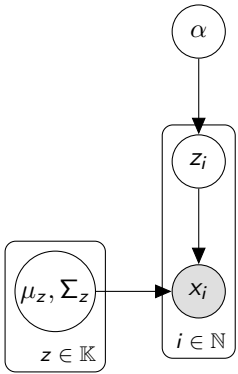
Gaussian:



GMM with 2 components:



GMM: Graphical model

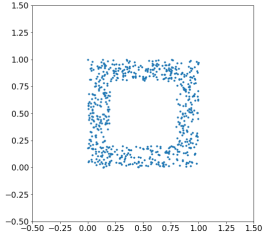


- i – Index of exemplar, $\in \mathbb{N}$
- z – Index of component, $\in \mathbb{K}$
- μ_z, Σ_z – Mean and covariance for component z
- α – Categorical distribution over component membership
- z_i – Which component feature vector i belongs to
- x_i – Feature vector for exemplar i
- Fit with EM

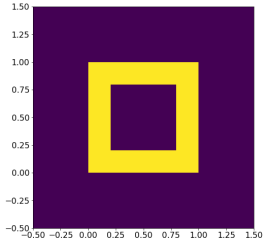
UNIVERSITY OF BATH

GMM: Demonstration

Input:

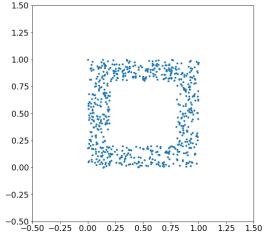


Ground Truth:

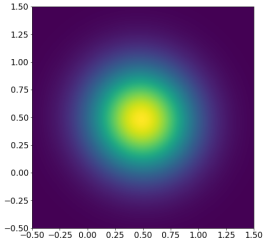


GMM: Demonstration

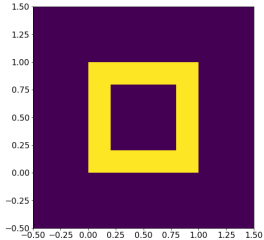
Input:



Gaussian:

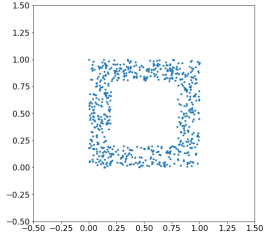


Ground Truth:

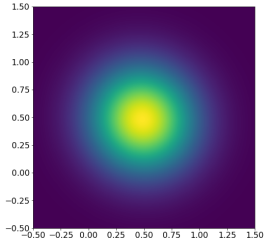


GMM: Demonstration

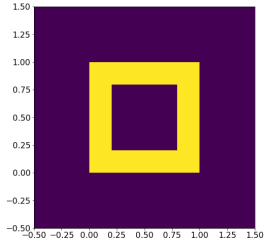
Input:



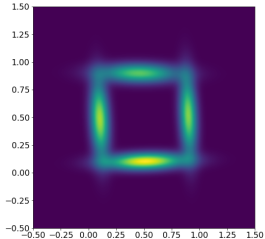
Gaussian:



Ground Truth:

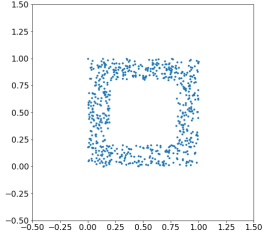


GMM with 4 components:

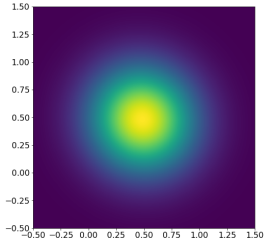


GMM: Demonstration

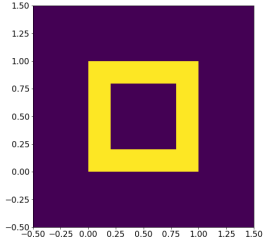
Input:



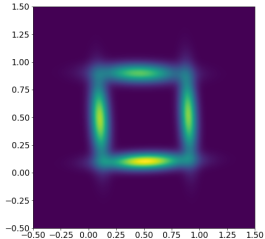
Gaussian:



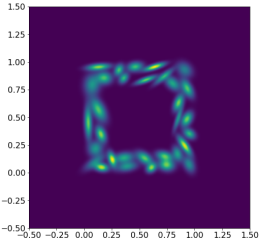
Ground Truth:



GMM with 4 components:



GMM with 32 components:



UNIVERSITY OF BATH

GMM: Component count

- Hyperparameter K = Number of Gaussian distributions
- Likelihood: $p(X|\hat{\theta})$ (X = exemplars, $\hat{\theta}$ = best model parameters)
- Maximising likelihood fails: More components $\implies$ increased likelihood

UNIVERSITY OF BATH

GMM: Component count

- Hyperparameter K = Number of Gaussian distributions
- Likelihood: $p(X|\hat{\theta})$ (X = exemplars, $\hat{\theta}$ = best model parameters)
- Maximising likelihood fails: More components $\implies$ increased likelihood

- Penalise adding extra parameters
- Bayesian Information Criterion (BIC):

$$\texttt{BIC} = \log(n)\texttt{len}(\theta) - 2\log(p(X|\hat{\theta}))$$

n = number of data points
$\texttt{len}(\theta)$ = number of parameters

GMM: Component count

- Hyperparameter K = Number of Gaussian distributions
- Likelihood: $p(X|\hat{\theta})$ (X = exemplars, $\hat{\theta}$ = best model parameters)
- Maximising likelihood fails: More components $\implies$ increased likelihood

- Penalise adding extra parameters
- Bayesian Information Criterion (BIC):

$$\texttt{BIC} = \log(n)\texttt{len}(\theta) - 2\log(p(X|\hat{\theta}))$$

n = number of data points
$\texttt{len}(\theta)$ = number of parameters

- An approximation: Only truly Bayesian as $n \to \infty$
- One of many; best approach is a *Dirichlet process prior*

UNIVERSITY OF BATH

Use case

- Visualisation (you already do this!)
- Generating data
- Classification
- Regression
- Missing values
- Abnormality detection

Generative vs discriminative

- Discriminative: Learns $P(y|x)$
  Cares about boundary between classes only
  e.g. Linear regression, support vector machine, neural network

- Generative: Learns $P(y, x)$ or $P(x|y)P(y)$
  Cares about entire distribution – harder, but more powerful
  e.g. Gaussian process, **density estimation**, variational autoencoder

UNIVERSITY OF BATH

Generative vs discriminative

- Discriminative: Learns $P(y|x)$
  Cares about boundary between classes only
  e.g. Linear regression, support vector machine, neural network

- Generative: Learns $P(y, x)$ or $P(x|y)P(y)$
  Cares about entire distribution – harder, but more powerful
  e.g. Gaussian process, **density estimation**, variational autoencoder

- Implicit: Can generate data, but doesn't model $P(y, x)$
  e.g. Generative adversarial networks

(doesn't have to be probabilistic)

UNIVERSITY OF BATH

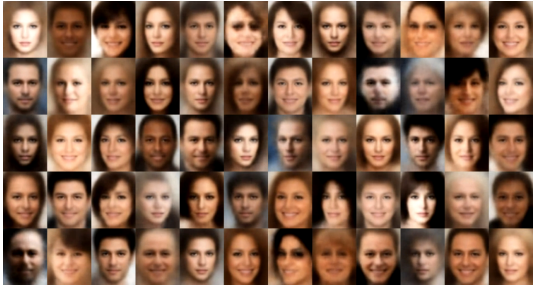
Generating data

- Learn $P(x)$ (density estimate)
- Draw new $\hat{x} \sim P(x)$

Generating data

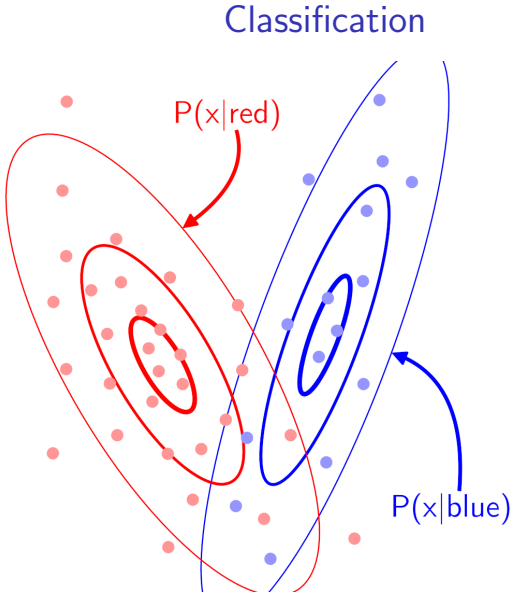
- Learn $P(x)$ (density estimate)
- Draw new $\hat{x} \sim P(x)$

- Quality of result often not great, e.g. faces from a variational autoencoder
  (which is a density estimation algorithm, though often not described as such!)



Generative classification:

1. Calculate $P(x|c)$ for each class $c \in C$ (density estimate)

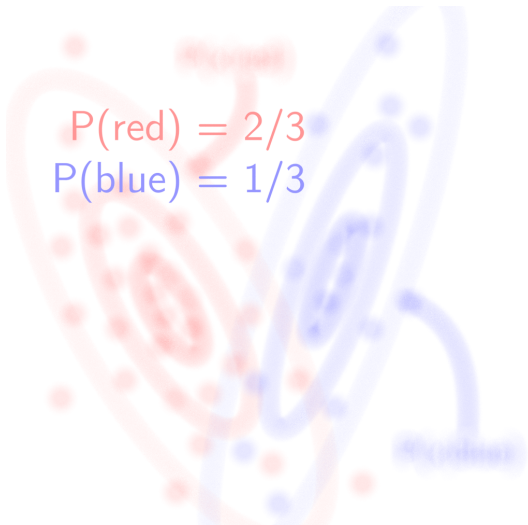


Classification

Generative classification:

1. Calculate $P(x|c)$ for each class $c \in C$
   (density estimate)

2. Fit class distribution, $P(c)$
   (usually categorical; Dirichlet prior for Bayesian)



Classification

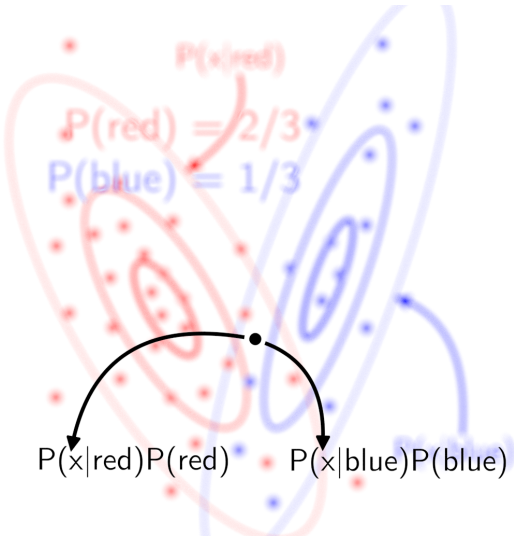
Generative classification:

1. Calculate $P(x|c)$ for each class $c \in C$
   (density estimate)

2. Fit class distribution, $P(c)$
   (usually categorical; Dirichlet prior for Bayesian)

3. Given new $\hat{x}$ apply Bayes rule:

$$P(c_i|\hat{x}) \propto P(\hat{x}|c_i)P(c_i)$$



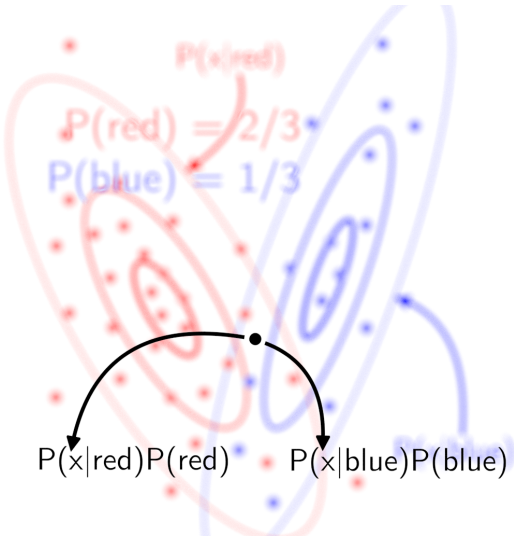
Classification

Generative classification:

1. Calculate $P(x|c)$ for each class $c \in C$
   (density estimate)
2. Fit class distribution, $P(c)$
   (usually categorical; Dirichlet prior for Bayesian)
3. Given new $\hat{x}$ apply Bayes rule:

$$P(c_i|\hat{x}) \propto P(\hat{x}|c_i)P(c_i)$$

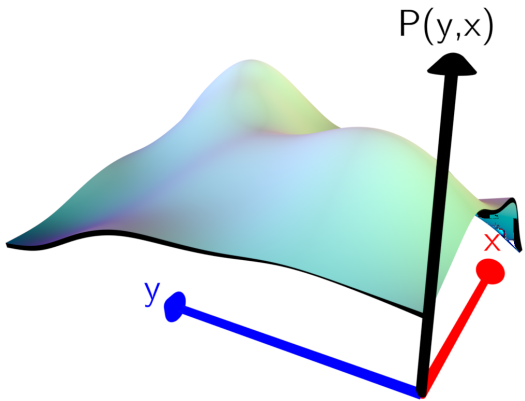
- The "right approach"
  ... but defeated by discriminative



Regression

Generative regression:

1. Fit $P(y, x)$
   (density estimate)



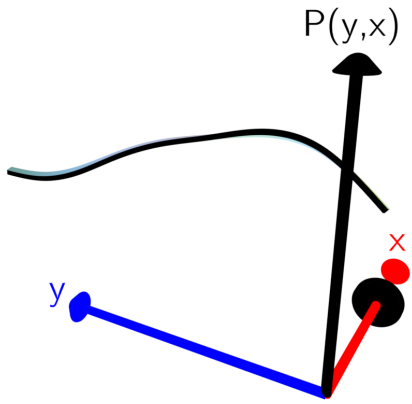
Regression

Generative regression:

1. Fit $P(y, x)$
   (density estimate)

2. Given $\hat{x}$ evaluate:

$$P(y|x = \hat{x}) \propto P(y, x = \hat{x})$$



Regression

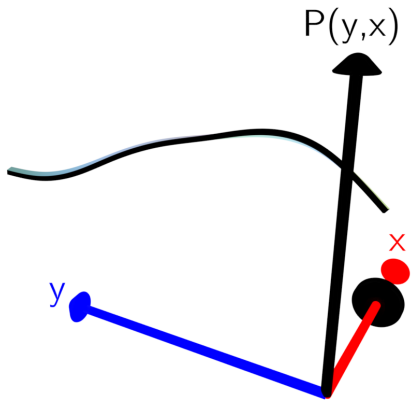
Generative regression:

1. Fit $P(y, x)$
   (density estimate)

2. Given $\hat{x}$ evaluate:

$$P(y|x = \hat{x}) \propto P(y, x = \hat{x})$$

- The "right approach"
  ... but defeated by discriminative



Missing values

- Classification:
  Marginalise out missing dimensions from $P(x|c)$
- Regression:
  Marginalise out missing dimensions from $P(y, x)$
- Generative models often handle missing data better than discriminative!

UNIVERSITY OF BATH

Abnormality detection

- Density estimate provides event probability
- Can be used to identify unusual (low probability) events, e.g.
  - Detecting fraud in bank transactions
  - Identifying unusual behaviour in CCTV

Abnormality detection

- Density estimate provides event probability
- Can be used to identify unusual (low probability) events, e.g.
  - Detecting fraud in bank transactions
  - Identifying unusual behaviour in CCTV

Background subtraction – per pixel density estimate:
(Dirichlet process Gaussian mixture model)

First row = input, second = ground truth, third = algorithm output

UNIVERSITY OF BATH

Calibrating probabilities

- Many ML models give "probabilities":
  1. K nearest neighbours: Percentage of K
  2. Logistic regression: Number between 0 and 1

  3. Random forest: Percentage of votes
  4. Regression forest: Density estimate fitted to samples

  5. SVM: Boundary distance
  6. Neural network: Class membership weight

  7. Gaussian process: Marginal Gaussian
  8. Probabilistic graphical models

UNIVERSITY OF BATH

Calibrating probabilities

- Many ML models give "probabilities":
  1. K nearest neighbours: Percentage of K –not a probability
  2. Logistic regression: Number between 0 and 1 –not a probability
  3. Random forest: Percentage of votes –not a probability
  4. Regression forest: Density estimate fitted to samples –**a probability!**
  5. SVM: Boundary distance –not a probability
  6. Neural network: Class membership weight –not a probability
  7. Gaussian process: Marginal Gaussian –**a probability!**
  8. Probabilistic graphical models –**a probability!**

UNIVERSITY OF BATH

Calibrating probabilities

- Many ML models give "probabilities":
  1. K nearest neighbours: Percentage of K –not a probability
  2. Logistic regression: Number between 0 and 1 –not a probability
  3. Random forest: Percentage of votes –not a probability
  4. Regression forest: Density estimate fitted to samples –**a probability!**
  5. SVM: Boundary distance –not a probability
  6. Neural network: Class membership weight –not a probability
  7. Gaussian process: Marginal Gaussian –**a probability!**
  8. Probabilistic graphical models –**a probability!**
- Most just some arbitrary "confidence" number...
  ...but you can calibrate! (not perfect)

UNIVERSITY OF BATH

Kolmogorov–Smirnov

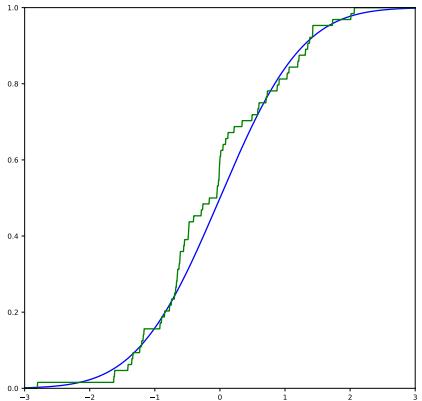
- Generative model, M; set of exemplars, X
- Assume 1D and can calculate $P(x|M)$
- How well does it fit?
  (note: modify following if M calculated from X)

Kolmogorov–Smirnov

- Two cumulative distribution functions (CDF):

$$\text{Blue}(v) = P(x < v|M)$$

$$\text{Green}(v) = \frac{|\{x \cdot x \in X \wedge x \leq v\}|}{|X|}$$



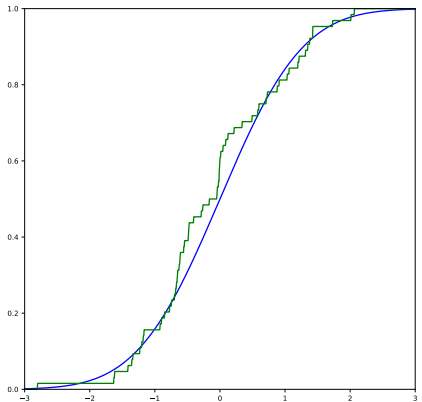
Kolmogorov–Smirnov

- Two cumulative distribution functions (CDF):

$$\text{Blue}(v) = P(x < v|M)$$

$$\text{Green}(v) = \frac{|\{x \cdot x \in X \wedge x \leq v\}|}{|X|}$$

- Find maximum vertical difference:
  $D = \sup_v |\,\text{Blue}(v) - \text{Green}(v)|$



Kolmogorov–Smirnov

- Two cumulative distribution functions (CDF):

$$\text{Blue}(v) = P(x < v|M)$$

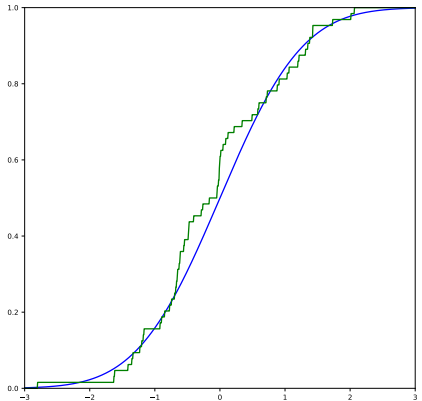
$$\text{Green}(v) = \frac{|\{x \cdot x \in X \wedge x \leq v\}|}{|X|}$$

- Find maximum vertical difference:
  $D = \sup_v |\,\text{Blue}(v) - \text{Green}(v)|$

- Accept null hypothesis (identical) at level α (0.05) if

$$\sqrt{|X|}D < \lambda$$

$$1 - \alpha = \frac{\sqrt{2\pi}}{\lambda} \sum_{k=1}^{\infty} \exp\left(\frac{-(2k-1)^2\pi^2}{8\lambda^2}\right)$$



Calibrating

- Model gives “probability” of 0.5
  . . . but it’s wrong

Calibrating

- Model gives "probability" of 0.5
  ... but it's wrong

- Choose transform of $P(x)$, e.g.

$$P'(x) \propto P(x)^{\beta}$$

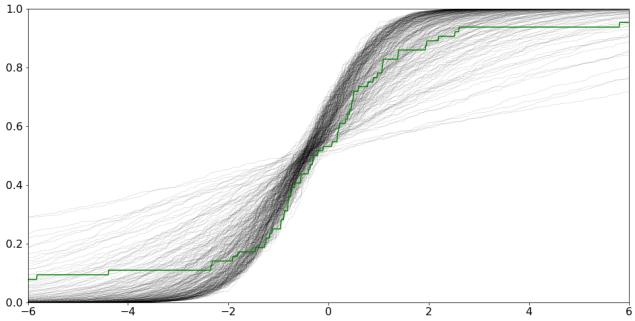
  and minimise D
  (don't need statistical test bit)

Calibrating

- Model gives "probability" of 0.5 . . . but it's wrong
- Choose transform of $P(x)$, e.g.

$$P'(x) \propto P(x)^{\beta}$$

and minimise D
(don't need statistical test bit)



Fitting Gaussian to a standard Cauchy (very silly!)

$$\mu = -0.43, \sigma = 8.1$$

$$\texttt{Best}: \beta = 21.9, D = 0.08$$

Details

- Can't evaluate CDF? Draw! (previous slide used importance sampling)

Details

- Can't evaluate CDF? Draw! (previous slide used importance sampling)
- Supervised learning:
  - CDF = Percentage of one class vs rest
  - Minimise all classes simultaneously

UNIVERSITY OF BATH

Details

- Can't evaluate CDF? Draw! (previous slide used importance sampling)
- Supervised learning:
  - CDF = Percentage of one class vs rest
  - Minimise all classes simultaneously
- Multidimensional case:
  1. Divide space into small regions
  2. Order by model probability
  3. Pretend 1D, as before

Summary

- New algorithm kind: **Density estimation**
- Unsupervised, but can be used for supervised

- Commonly used for visualisation and debugging
- Can solve most other problems!

- Calibrating "probabilities" of supervised algorithms

Further reading

- How to find the modes of a kernel density estimate:
  "*Mean Shift: A Robust Approach Toward Feature Space Analysis*",
  by Dorin & Meer (2002)

- Bayesian blocks – variable histogram bin widths:
  "*Studies in Astronomical Time Series Analysis. V. Bayesian Blocks, a New Method to Analyze Structure in Photon Counting Data*",
  by Scargle (1998)

- Incremental / real-time Gaussian mixture model:
  "*Incremental One-Class Learning with Bounded Computational Complexity*",
  by Sillito & Fisher (2007)

- Gaussian Conjugate Prior Cheat Sheet – Bayesian fitting of multivariate Gaussian:
  `http://thaines.com/content/misc/gaussian_conjugate_prior_cheat_sheet.pdf`

Sources

- Crime in Vancouver:
  `https://www.kaggle.com/wosaku/crime-in-vancouver`
- House Sales in King County, USA:
  `https://www.kaggle.com/harlfoxem/housesalesprediction`
- VAE face image:
  `https://github.com/WojciechMormul/vae`
- Background subtraction image:
  "*Background Subtraction with Dirichlet Process Mixture Models*",
  by Haines & Xiang (2014)