Machine Learning 2.19: Seven Random Ideas

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

UNIVERSITY OF BATH

Ten

1. Everything matters
2. Beware the central limit theorem
3. Choice of bells
4. Calibrating distributions
5. Branch & bound
6. Stupid can be smart
7. Random kitchen sinks

1. Everything matters

UNIVERSITY OF BATH

Whole system

- Good ML does not matter if you have...
  - Messy data
  - Poorly designed features
  - Wrong cost function

Messy data

- Repetition solves:
  - Data entry errors
  - Mislabelling
  - Disagreement

  (vote on right answer)

- Outliers

UNIVERSITY OF BATH

Outlier detection

- Preprocessing:
  - Density estimate
  - Remove low probability exemplars
    (e.g bottom 10% of data)

UNIVERSITY OF BATH

Outlier detection

- Preprocessing:
  - Density estimate
  - Remove low probability exemplars
    (e.g bottom 10% of data)
- Supervised probabilistic model:
  - Fit model to all data
  - Check items with strong model/label disagreement
  - Refit model with cleaned data

UNIVERSITY OF BATH

Outlier detection

- Preprocessing:
  - Density estimate
  - Remove low probability exemplars
    (e.g bottom 10% of data)
- Supervised probabilistic model:
  - Fit model to all data
  - Check items with strong model/label disagreement
  - Refit model with cleaned data
- Modify model:
  - "is outlier" Bernoulli random variable
  - Cluster/mixture component for rare data

UNIVERSITY OF BATH

Outlier detection

- Preprocessing:
  - Density estimate
  - Remove low probability exemplars
    (e.g bottom 10% of data)
- Supervised probabilistic model:
  - Fit model to all data
  - Check items with strong model/label disagreement
  - Refit model with cleaned data
- Modify model:
  - "is outlier" Bernoulli random variable
  - Cluster/mixture component for rare data
- Outlier detection can remove good data!
  - Experimental problems (problem hiding)
  - Rare classes

UNIVERSITY OF BATH

Poorly designed features

- Useful information omitted
- Dependent on unreliable information
  (often omitted or often wrong)
- Too expensive to calculate

Wrong cost function

- Specification error: When the client asks for the wrong thing!

UNIVERSITY OF BATH

Wrong cost function

- Specification error: When the client asks for the wrong thing!
- Solution: Talking, a lot
- Be careful of language
- Consider ethics

2. Beware the central limit theorem

UNIVERSITY OF BATH

Nothing is Normal

- Central limit theorem:
  It's normal to be Normal (Gaussian distribution)

UNIVERSITY OF BATH

Nothing is Normal

- Central limit theorem:
  It's normal to be Normal (Gaussian distribution)
- Reality:
  Most things are not

UNIVERSITY OF BATH

Nothing is Normal

- Central limit theorem:
  It's normal to be Normal (Gaussian distribution)
- Reality:
  Most things are not
- Aside:
  - *Gaussian* is the original name, after Gauss
    (de Moivre got there first, then Laplace, plus Adrain also discovered it in parallel)
  - Pearson popularised *Normal*
  - Regretted doing so

UNIVERSITY OF BATH

Nothing is Normal

- Central limit theorem:
  It's normal to be Normal (Gaussian distribution)
- Reality:
  Most things are not
- Aside:
  - *Gaussian* is the original name, after Gauss
    (de Moivre got there first, then Laplace, plus Adrain also discovered it in parallel)
  - Pearson popularised *Normal*
  - Regretted doing so
- Sometimes it's correct/doesn't matter
- Sometimes it really, really matters

UNIVERSITY OF BATH

2008 financial crash

- Value at risk – one of the key contributors
  e.g. you might say you have a 1% one year VaR of 1000 units
  Means you expect to lose 1000 units every 100 years

UNIVERSITY OF BATH

2008 financial crash

- Value at risk – one of the key contributors
  e.g. you might say you have a 1% one year VaR of 1000 units
  Means you expect to lose 1000 units every 100 years

- Multiple ways to calculate, but two assumptions:
  - Risk has a Gaussian distribution
  - Assets are independent

UNIVERSITY OF BATH

2008 financial crash

- Value at risk – one of the key contributors
  e.g. you might say you have a 1% one year VaR of 1000 units
  Means you expect to lose 1000 units every 100 years

- Multiple ways to calculate, but two assumptions:
  - Risk has a Gaussian distribution
  - Assets are independent

- Underestimated risk:
  - Actually "fat tailed"
  - Markets are not independent!
    (don't make this mistake either)

Beware summary statistics

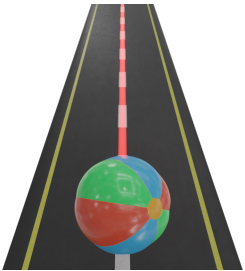
- Imagine a ball bouncing down a road. . . is it safe?

$\mu = 0$

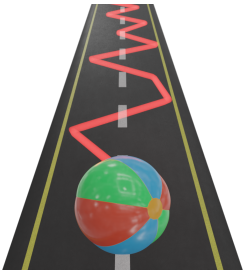
Beware summary statistics

- Imagine a ball bouncing down a road. . . is it safe?

$\mu = 0$



$\mu = 0$

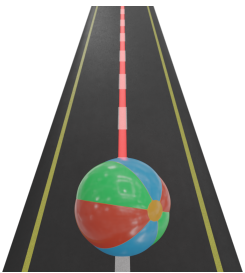


$\mu = 0$

Beware summary statistics

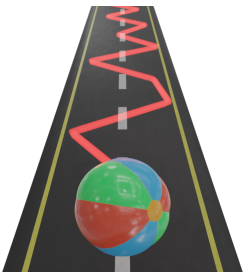
- Imagine a ball bouncing down a road. . . is it safe?

$\mu = 0$



$\mu = 0$

$\sigma^2 = 0$



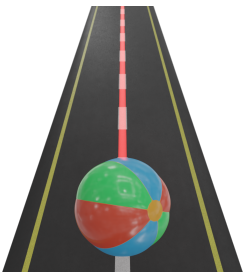
$\mu = 0$

$\sigma^2 = 1$

Beware summary statistics

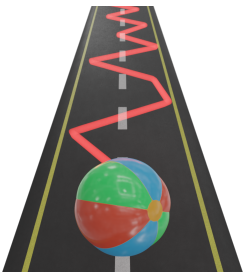
- Imagine a ball bouncing down a road. . . is it safe?

$\mu = 0$



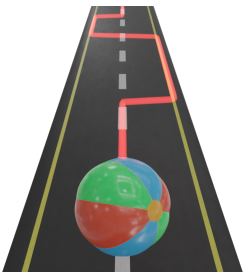
$\mu = 0$

$\sigma^2 = 0$



$\mu = 0$

$\sigma^2 = 1$



$\mu = 0$

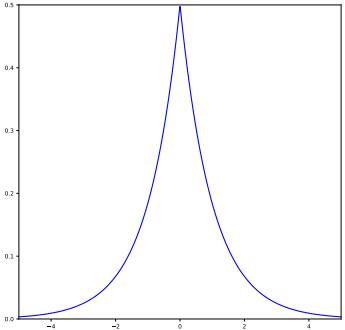
$\sigma^2 = 1$

- Evaluate cost functions on the distribution ($\mathbb{E}[C(x)]$), never the mean ($C(\mathbb{E}[c])$)

3. Choice of bells

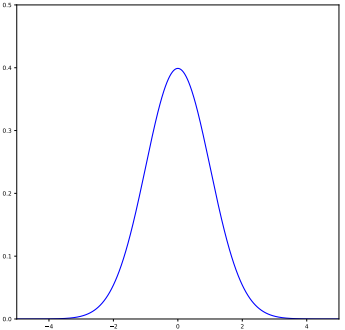
Three bells

- There are many unimodal distributions...



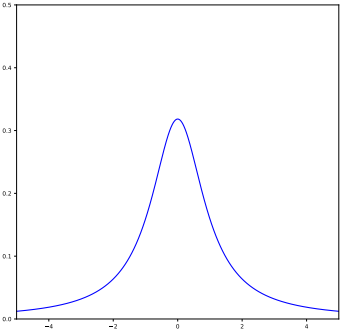
Laplace distribution

$$\frac{1}{2s}\exp\left(-\frac{|x-\mu|}{s}\right)$$



Gaussian distribution

$$\frac{1}{\sqrt{2\pi s^2}}\exp\left(-\frac{(x-\mu)^2}{2s^2}\right)$$



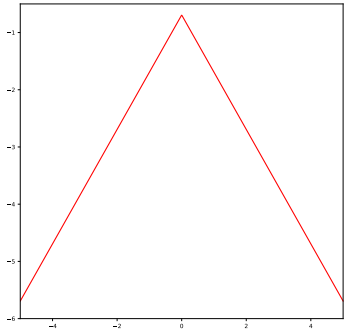
Cauchy distribution

$$\frac{1}{\pi s\left(1+\left(\frac{x-\mu}{s}\right)^2\right)}$$

(μ = location, s = scale)

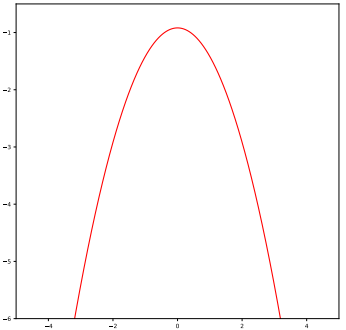
In log space

- There are many unimodal distributions...



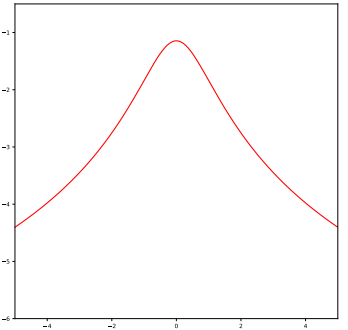
Laplace distribution

$$\frac{1}{2s}\exp\left(-\frac{|x-\mu|}{s}\right)$$



Gaussian distribution

$$\frac{1}{\sqrt{2\pi s^2}}\exp\left(-\frac{(x-\mu)^2}{2s^2}\right)$$



Cauchy distribution

$$\frac{1}{\pi s\left(1+\left(\frac{x-\mu}{s}\right)^2\right)}$$

(μ = location, s = scale)

UNIVERSITY OF BATH

Choice?

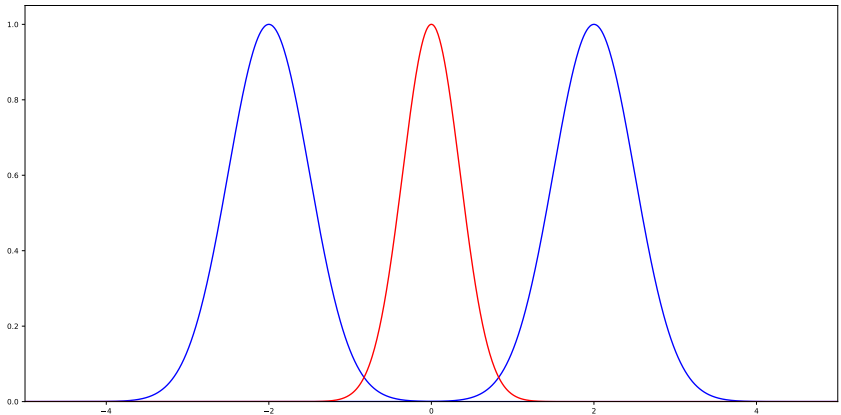
- Bagging/boosting/committees:
  Combine information from multiple estimates
- e.g. regression forests combine multiple trees
  (typically Gaussian, typically multiplication)

UNIVERSITY OF BATH

Choice?

- Bagging/boosting/committees:
  Combine information from multiple estimates
- e.g. regression forests combine multiple trees
  (typically Gaussian, typically multiplication)
- Choice should depend on desired behaviour! (not mathematical convenience)

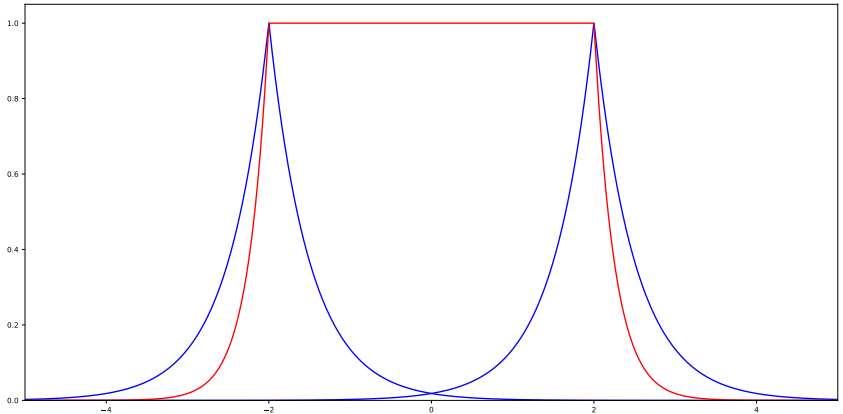
Gaussian multiplication



- Mean result, more confident (is this rational?)

UNIVERSITY OF BATH

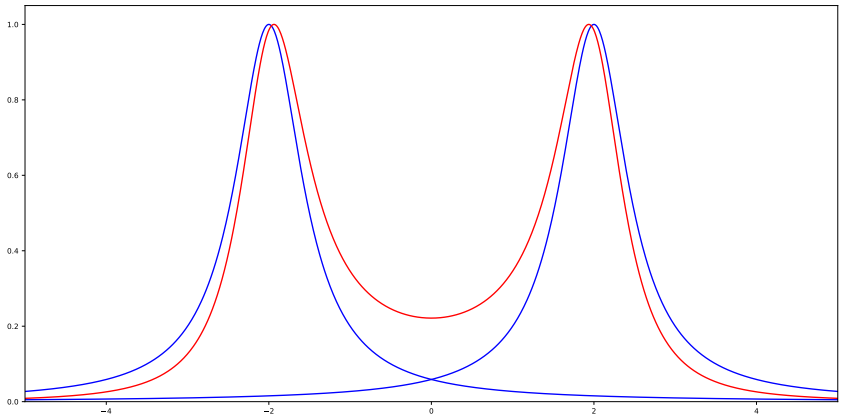
Laplace multiplication



- Somewhere in between, less confident

UNIVERSITY OF BATH

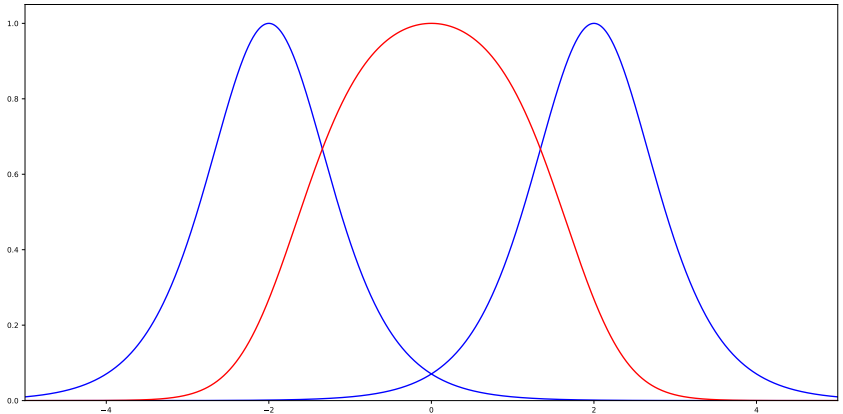
Cauchy multiplication



- One is right, the other wrong, less confident

UNIVERSITY OF BATH

Logistic multiplication



- (bonus!) Mean result, less confident

4. Calibrating distributions

Kolmogorov–Smirnov

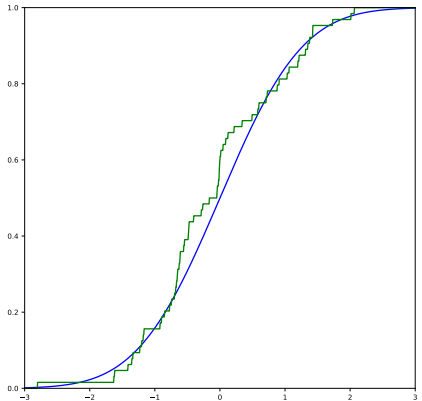
- Generative model, M; set of exemplars, X
- Assume 1D and can calculate $P(x|M)$
- How well does it fit?
  (note: modify following if M calculated from X)

Kolmogorov–Smirnov

- Two cumulative distribution functions (CDF):

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

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



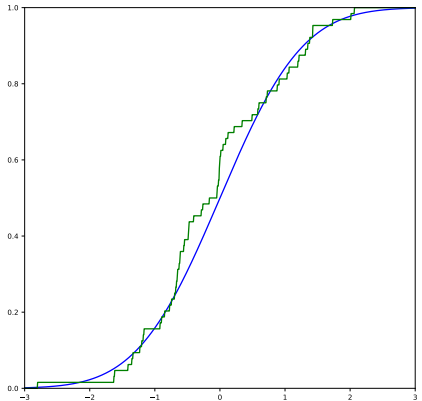
Kolmogorov–Smirnov

- 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):

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

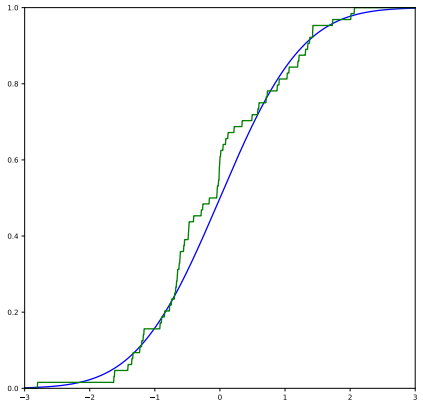
$$\text{Green}(v) = \frac{|\{x \cdot x \in X \wedge x \leq v\}|}{|X|}$$

- 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

- If a model gives a probability of 0.5, is it right?
- Kolmogorov–Smirnov answers

Calibrating

- If a model gives a probability of 0.5, is it right?
- Kolmogorov–Smirnov answers

- If not, calibrate
  - by minimising D!
- Define a transform of $P(x)$, e.g.

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

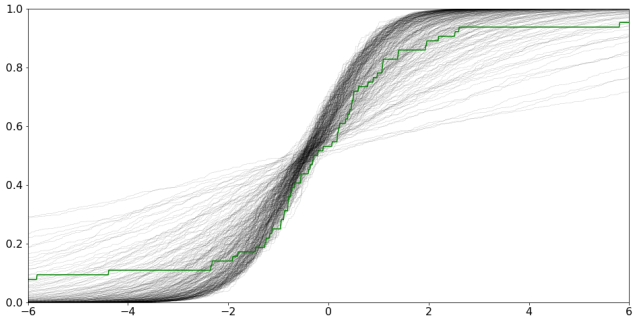
- No longer normalised!

Calibrating

- If a model gives a probability of 0.5, is it right?
- Kolmogorov–Smirnov answers
- If not, calibrate
  – by minimising D!
- Define a transform of $P(x)$, e.g.

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

- No longer normalised!



Fitting Gaussian to a standard Cauchy (bad idea!)

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

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

Notes

- Can't evaluate? Draw! (previous slide used importance sampling)
- Multidimensional case:
  1. Divide space into small regions
  2. Order by model probability
  3. Pretend 1D, as before

5. Branch & bound

Optimisation

- Many powerful approaches!
- Sometimes something simple is enough...

UNIVERSITY OF BATH

Binary search

- You should already know this, but...
- For solving $f(x) = y$ when y known, x unknown
- $f(x)$ must be increasing or decreasing

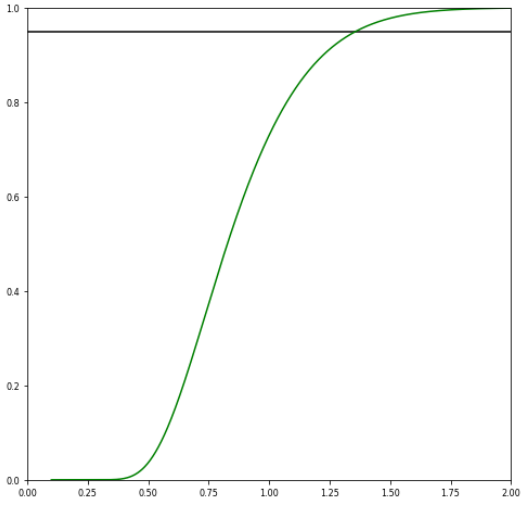
Binary search

- You should already know this, but...
- For solving $f(x) = y$ when y known, x unknown
- $f(x)$ must be increasing or decreasing

- Consider

$$0.95 = \frac{\sqrt{2\pi}}{\lambda} \sum_{k=1}^{\infty} \exp\left(\frac{-(2k-1)^2\pi^2}{8\lambda^2}\right)$$

(from Kolmogorov–Smirnov)



Binary search

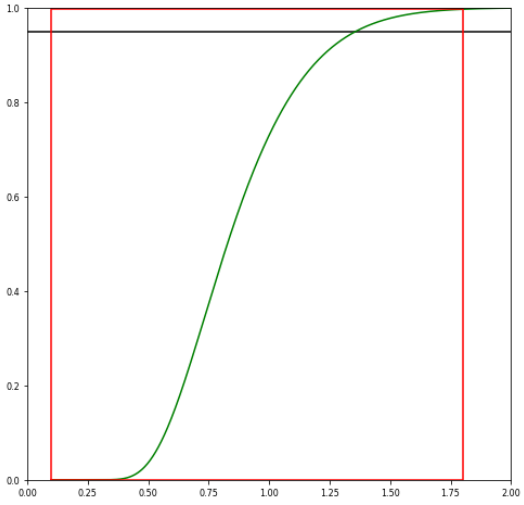
- You should already know this, but...
- For solving $f(x) = y$ when y known, x unknown
- $f(x)$ must be increasing or decreasing

- Consider

$$0.95 = \frac{\sqrt{2\pi}}{\lambda} \sum_{k=1}^{\infty} \exp\left(\frac{-(2k-1)^2\pi^2}{8\lambda^2}\right)$$

(from Kolmogorov–Smirnov)

- Initialise `low` and `high`



Binary search

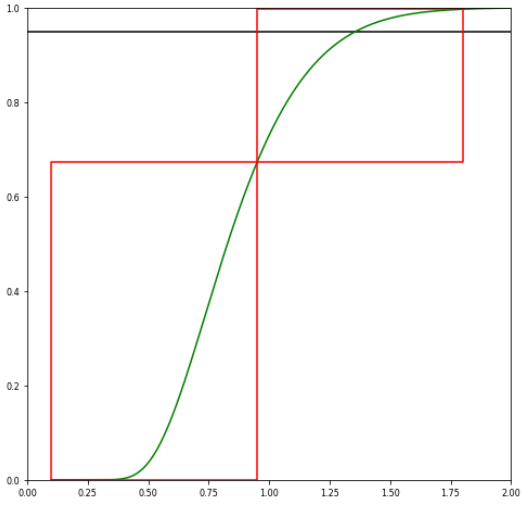
- You should already know this, but...
- For solving $f(x) = y$ when y known, x unknown
- $f(x)$ must be increasing or decreasing

- Consider

$$0.95 = \frac{\sqrt{2\pi}}{\lambda} \sum_{k=1}^{\infty} \exp\left(\frac{-(2k-1)^2\pi^2}{8\lambda^2}\right)$$

(from Kolmogorov–Smirnov)

- Initialise `low` and `high`
- Keep subdividing until range tight



Binary search

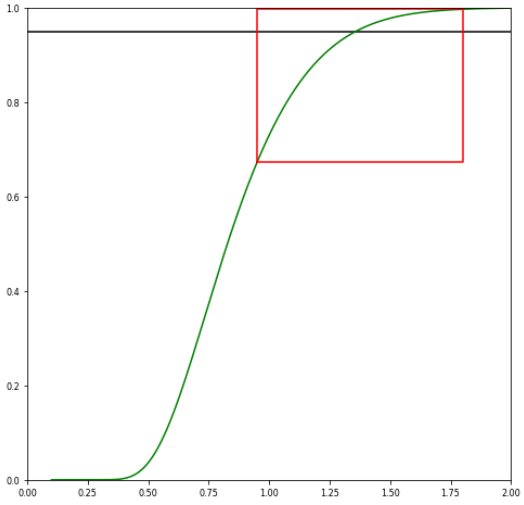
- You should already know this, but...
- For solving $f(x) = y$ when y known, x unknown
- $f(x)$ must be increasing or decreasing

- Consider

$$0.95 = \frac{\sqrt{2\pi}}{\lambda} \sum_{k=1}^{\infty} \exp\left(\frac{-(2k-1)^2\pi^2}{8\lambda^2}\right)$$

(from Kolmogorov–Smirnov)

- Initialise `low` and `high`
- Keep subdividing until range tight



Binary search

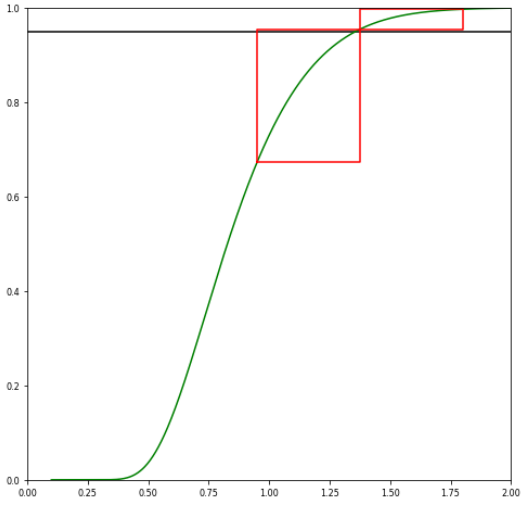
- You should already know this, but...
- For solving $f(x) = y$ when y known, x unknown
- $f(x)$ must be increasing or decreasing

- Consider

$$0.95 = \frac{\sqrt{2\pi}}{\lambda} \sum_{k=1}^{\infty} \exp\left(\frac{-(2k-1)^2\pi^2}{8\lambda^2}\right)$$

(from Kolmogorov–Smirnov)

- Initialise `low` and `high`
- Keep subdividing until range tight



Binary search

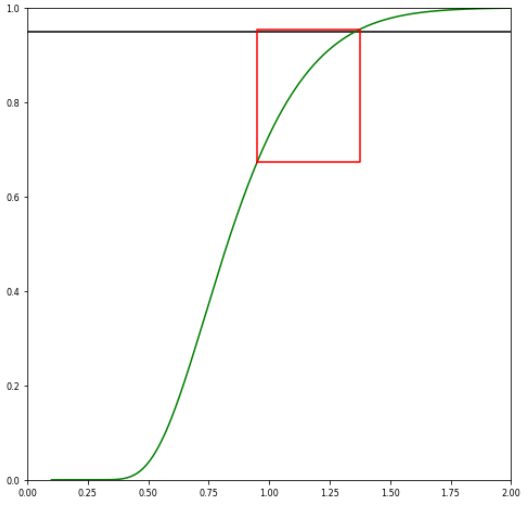
- You should already know this, but...
- For solving $f(x) = y$ when y known, x unknown
- $f(x)$ must be increasing or decreasing

- Consider

$$0.95 = \frac{\sqrt{2\pi}}{\lambda} \sum_{k=1}^{\infty} \exp\left(\frac{-(2k-1)^2\pi^2}{8\lambda^2}\right)$$

(from Kolmogorov–Smirnov)

- Initialise `low` and `high`
- Keep subdividing until range tight



Branch & bound I

- Extends idea to

$$\underset{x}{\operatorname{argmin}}\, f(x)$$

- No constraint on $f(x)$, global optimum

UNIVERSITY OF BATH

Branch & bound I

- Extends idea to

$$\underset{x}{\operatorname{argmin}}\, f(x)$$

- No constraint on $f(x)$, global optimum
- Need (loose) bounds

$$f(x) \leq f^H(x_l, x_h), x^L \leq x \leq x^H$$
$$f(x) \geq f^L(x_l, x_h), x^L \leq x \leq x^H$$

(tighter is better)

- Tree search, bounds prune branches

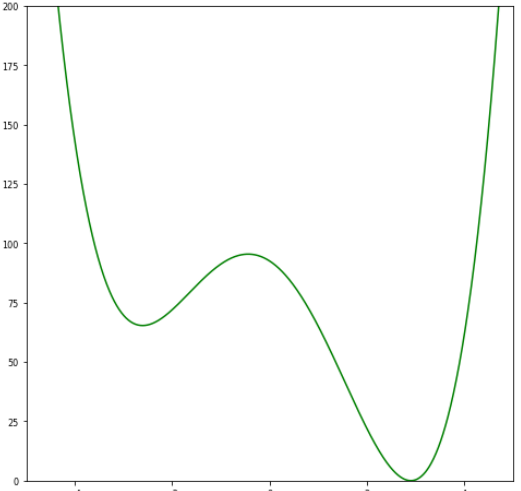
Branch & bound II

- Optimise:

$$f(x) = (x-3)^2 * (2.0 + (x+3)^2)$$

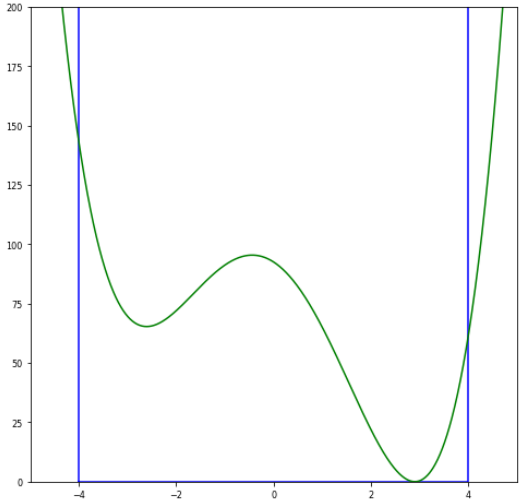
$$f^H(x_l, x_h) = \max((x_l - 3)^2, (x_h - 3)^2) \times (2 + \max((x_l + 3)^2, (x_h + 3)^2))$$

$$f^L(x_l, x_h) = \min((x_l - 3)^2, (x_h - 3)^2, 0 \text{ if } x_l < 3 < x_h) \times (2 + \min((x_l + 3)^2, (x_h + 3)^2, 0 \text{ if } x_l < -3 < x_h))$$



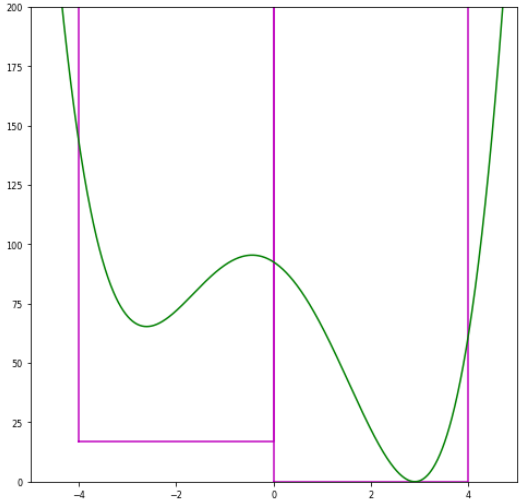
Branch & bound II

- Choose search range



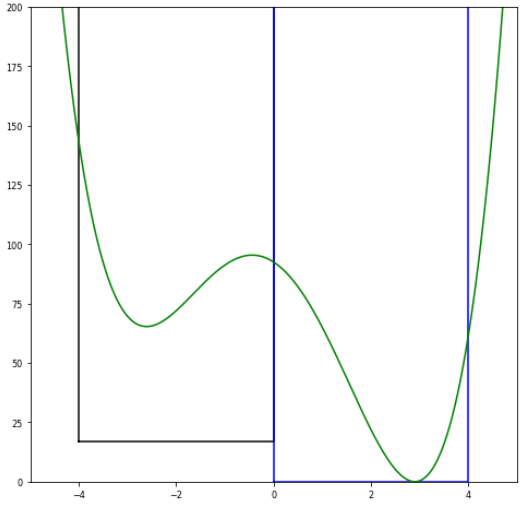
Branch & bound II

- Choose search range
- Subdivide



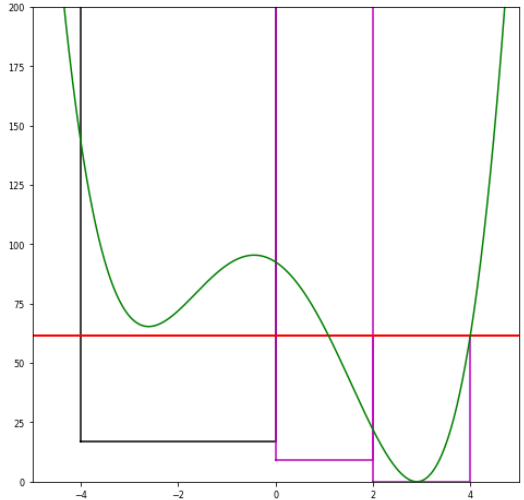
Branch & bound II

- Choose search range
- Subdivide
- Select region with best lower bound



Branch & bound II

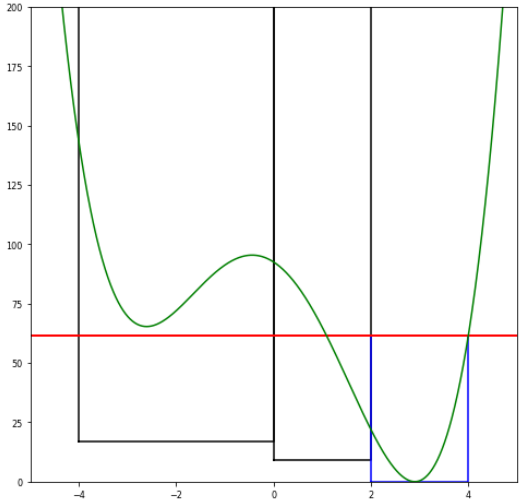
- Choose search range
- Subdivide
- Select region with best lower bound
- Subdivide. Red line is best upper bound



Branch & bound II

- Choose search range
- Subdivide
- Select region with best lower bound
- Subdivide. Red line is best upper bound

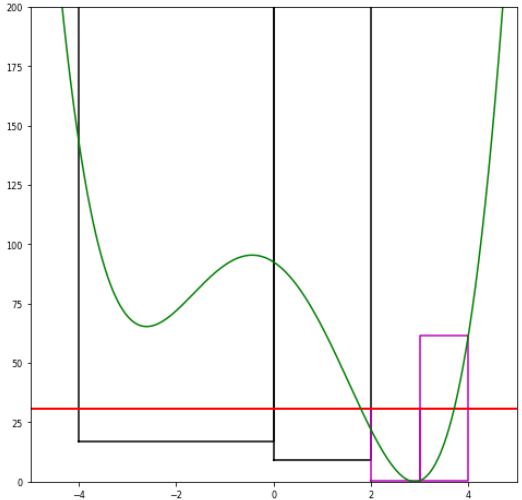
- Keep going ...



Branch & bound II

- Choose search range
- Subdivide
- Select region with best lower bound
- Subdivide. Red line is best upper bound

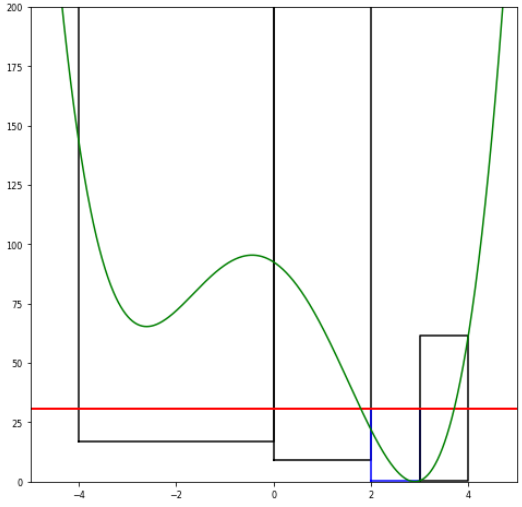
- Keep going ...



Branch & bound II

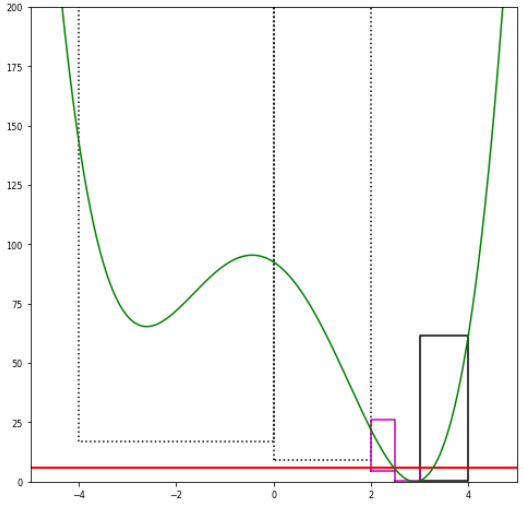
- Choose search range
- Subdivide
- Select region with best lower bound
- Subdivide. Red line is best upper bound

- Keep going . . .



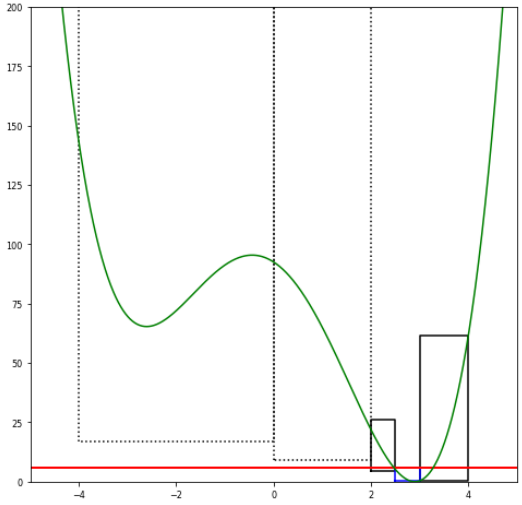
Branch & bound II

- Choose search range
- Subdivide
- Select region with best lower bound
- Subdivide. Red line is best upper bound
- Keep going . . .
- Delete any region with a lower bound above the best upper bound



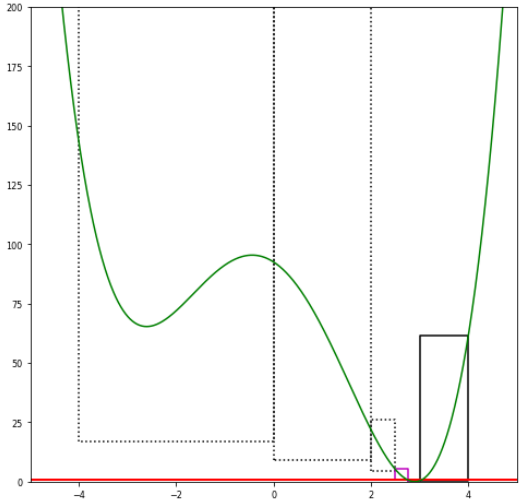
Branch & bound II

- Choose search range
- Subdivide
- Select region with best lower bound
- Subdivide. Red line is best upper bound
- Keep going ...
- Delete any region with a lower bound above the best upper bound



Branch & bound II

- Choose search range
- Subdivide
- Select region with best lower bound
- Subdivide. Red line is best upper bound
- Keep going . . .
- Delete any region with a lower bound above the best upper bound



UNIVERSITY OF BATH

Branch & bound III

- Gradient descent to refine/tighten upper bounds
- Generalises to many dimensions
- Generalises to non-numeric,
  e g. can apply to travelling salesperson
- Memory explosion from tree limits use
- Often slow

6. Stupid can be smart

UNIVERSITY OF BATH

Simple $\neq$ bad

Typically:

- You get 80% of the performance with simple algorithms
- Big data + simple algorithm > little data + smart algorithm
- Random selection does well at active learning
- Features designed using domain knowledge will win

UNIVERSITY OF BATH

Hyperparameter optimisation

- Grid search:
  1. Choose discrete set of values for each parameter
  2. Evaluate every combination
  3. Select best
- Random search:
  1. Put distribution on each parameter
  2. Draw n sets of values
  3. Evaluate each
  4. Select best
- Which is best?

UNIVERSITY OF BATH

Hyperparameter optimisation

- Grid search:
  1. Choose discrete set of values for each parameter
  2. Evaluate every combination
  3. Select best
- Random search:
  1. Put distribution on each parameter
  2. Draw n sets of values
  3. Evaluate each
  4. Select best
- Which is best?

- Most parameters don't matter
- Those that do often only matter sometimes
- Grid search wastes time with irrelevant combinations
- Random search *gets better results* for the same computation
- Random wins!

7. Random kitchen sinks

Random kitchen sinks

- **Random kitchen sinks**, also called
  - Random neural networks
  - Extreme learning machines
  - Reservoir computing (Includes doing this physically, e.g. shining lasers through crystals...)

(people keep reinventing this...)

Random kitchen sinks

- **Random kitchen sinks**, also called
  - Random neural networks
  - Extreme learning machines
  - Reservoir computing (Includes doing this physically, e.g. shining lasers through crystals...)

  (people keep reinventing this...)

- Algorithm:
  1. Select non-linearity, $\phi(x)$ (anything you would use in a neural network, e.g. $\sin(x)$)
  2. Given F features generate K new features:

     $$f_g(\vec{x}) = \phi(\vec{x} \cdot \vec{w}_k)$$

     where w_k is a (generated once) **random vector** (standard Gaussian often good choice)
  3. Train simple model on new features, e.g. linear regression (can include original features as well)

     $$y = \sum_{k \in K} \alpha_k \phi(\vec{x} \cdot \vec{w}_k)$$

Wat? I

Given

$$y = \sum_{k \in K} \alpha_k \phi(\vec{x} \cdot \vec{w}_k)$$

you would normally minimise some distance, e.g.

$$\text{argmin} \left(\sum_i \left(y_i - \sum_{k \in K} \alpha_k \phi(\vec{x}_i \cdot \vec{w}_k) \right)^2 \right)$$

by optimising $\alpha \in \mathbb{R}^K$ and $W \in \mathbb{R}^{K \times F}$

Wat? I

Given

$$y = \sum_{k \in K} \alpha_k \phi(\vec{x} \cdot \vec{w}_k)$$

you would normally minimise some distance, e.g.

$$\text{argmin} \left(\sum_i \left(y_i - \sum_{k \in K} \alpha_k \phi(\vec{x}_i \cdot \vec{w}_k) \right)^2 \right)$$

by optimising $\alpha \in \mathbb{R}^K$ and $W \in \mathbb{R}^{K \times F}$

Kitchen sinks observes that *randomisation* can replace (some) *optimisation* (randomise W, minimise α)

Wat? II

- Basis vector:
  - In N dimensional space N orthogonal vectors can describe all points uniquely

$$\forall x \cdot x = \sum_{i=1}^{N} \alpha_i \vec{b}_i$$

where α are weights, $\vec{b}_i$ the basis vectors

Wat? II

- Basis vector:
  - In N dimensional space N orthogonal vectors can describe all points uniquely

$$\forall x \cdot x = \sum_{i=1}^{N} \alpha_i \vec{b}_i$$

    where α are weights, $\vec{b}_i$ the basis vectors

  - More than N creates redundancy, representation not unique
  - Not orthogonal also means representation not unique
  - Uniqueness doesn't matter!

Wat? II

- Basis vector:
  - In N dimensional space N orthogonal vectors can describe all points uniquely

$$\forall x \cdot x = \sum_{i=1}^{N} \alpha_i \vec{b}_i$$

    where α are weights, $\vec{b}_i$ the basis vectors

  - More than N creates redundancy, representation not unique
  - Not orthogonal also means representation not unique
  - Uniqueness doesn't matter!

  - Lots of **random basis vectors**: Can represent every x, or at least get really close

Wat? II

- Basis vector:
  - In N dimensional space N orthogonal vectors can describe all points uniquely

$$\forall x \cdot x = \sum_{i=1}^{N} \alpha_i \vec{b}_i$$

where α are weights, $\vec{b}_i$ the basis vectors

  - More than N creates redundancy, representation not unique
  - Not orthogonal also means representation not unique
  - Uniqueness doesn't matter!

  - Lots of **random basis vectors**: Can represent every x, or at least get really close

- Random kitchen sinks = **random basis functions**:
  - $\phi(\vec{x} \cdot \vec{w}_k)$: Random basis function
  - α_k: Weights to find the closest function to the target function

Wat? III

- Little coding effort – saves you time!
- Potentially need lots of random features:
  - Optimising α can get expensive (inverting matrix often too much)
  - Model evaluation slow
  - Need lots of memory to store W (Can store RNG seed instead however)

UNIVERSITY OF BATH

Example

- Code to generate extra features:

```
w = numpy.random.standard_normal((extra, x.shape[1]))
new_x = numpy.sin(numpy.einsum('ef,gf->eg', x, w))
ext_x = numpy.append(x, new_x, axis=1)
```

UNIVERSITY OF BATH

Example

- Code to generate extra features:

```
w = numpy.random.standard_normal((extra, x.shape[1]))
new_x = numpy.sin(numpy.einsum('ef,gf->eg', x, w))
ext_x = numpy.append(x, new_x, axis=1)
```

- Run on white wine quality data set:
  - Original (11): Error = 0.755
  - 256 extra: Error = 0.717
  - 512 extra: Error = 0.686
  - 1024 extra: Error = 0.627
  - 2048 extra: Error = 0.500
  - 4096 extra: Error = 0.000

RMSE error on train set to show representation ability and ignore overfitting,
source: `https://archive.ics.uci.edu/ml/datasets/wine+quality`

Summary

- Always more to learn!
- Never forget . . .
  - system as a whole
  - simple approaches
  - limitations of ML

UNIVERSITY OF BATH

Further reading

- Why random search is better than grid search:
  "*Random Search for Hyper-Parameter Optimization*"
  by J. Bergstra & Y. Bengio
  `http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf`

- Random kitchen sinks:
  "*Weighted Sums of Random Kitchen Sinks: Replacing minimization with randomization in learning*"
  by A. Rahimi & B. Recht
  `https://people.eecs.berkeley.edu/~brecht/kitchensinks.html`