CPSC 440: Machine Learning

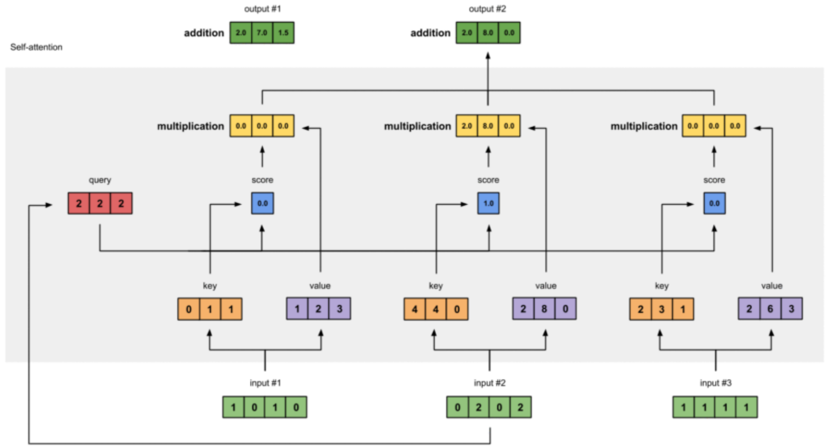
Gaussians

Winter 2022

Last Time: Attention and Transformers

- We discussed attention in RNNs:
  - Re-weight encoder states into context vector at each time.

- We discuss transformer networks:
  - Include "self-attention" layers.
    - Use attention mechanism between all input values.
  - Outperform CNNs/RNNs on many applications.
    - Many details/heuristics needed to make transformers work.
  - Basis of modern self-supervised language models like BERT and GPT-3.
    - Use BERT as pre-training for language models.



https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/
https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

OpenAI's GPT-3

- One of the most widely-used methods is GPT-3:
  - Recent "massive number of parameters" NLP model.
    - Full version has 175 billion parameters.
  - Often works well in new applications with little or no "fine-tuning" on the application (pre-training does almost everything).
  - Basis for many modern language applications.
  - See the paper for a starting point on where we are (and are not) in terms of language understanding.

Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*

Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry

Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan

Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter

Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray

Benjamin Chess Jack Clark Christopher Berner

Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

OpenAI

Abstract

Recent work has demonstrated substantial gains on many NLP tasks and benchmarks by pre-training on a large corpus of text followed by fine-tuning on a specific task. While typically task-agnostic in architecture, this method still requires task-specific fine-tuning datasets of thousands or tens of thousands of examples. By contrast, humans can generally perform a new language task from only a few examples or from simple instructions – something which current NLP systems still largely struggle to do. Here we show that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even reaching competitiveness with prior state-of-the-art fine-tuning approaches. Specifically, we train GPT-3, an autoregressive language model with 175 billion parameters, 10x more than any previous non-sparse language model, and test its performance in the few-shot setting. For all tasks, GPT-3 is applied without any gradient updates or fine-tuning, with tasks and few-shot demonstrations specified purely via text interaction with the model. GPT-3 achieves strong performance on many NLP datasets, including translation, question-answering, and cloze tasks, as well as several tasks that require on-the-fly reasoning or domain adaptation, such as unscrambling words, using a novel word in a sentence, or performing 3-digit arithmetic. At the same time, we also identify some datasets where GPT-3's few-shot learning still struggles, as well as some datasets where GPT-3 faces methodological issues related to training on large web corpora. Finally, we find that GPT-3 can generate samples of news articles which human evaluators have difficulty distinguishing from articles written by humans. We discuss broader societal impacts of this finding and of GPT-3 in general.

https://arxiv.org/pdf/2005.14165.pdf

More Applications

- Generating memes:
  - https://github.com/alpv95/Dank-Learning
- Generating Wikipedia articles:
  - https://arxiv.org/pdf/1801.10198.pdf
- Talking with historical figures:
  - https://www.besttechie.com/aiwriter-uses-openai-to-simulate-conversations-with-historical-figures/
- Generating music:
  - https://magenta.tensorflow.org/music-transformer
- Writing code:
  - https://copilot.github.com
- Generating video game content:
  - https://play.aidungeon.io/main/home

End of Part 2 (“Categorical Variables”): Key Concepts

- We discussed categorical density estimation.
  - Model the proportion of times different categories appear.
  - Categorical θ_c parameterization and unnormalized probabilities $\tilde{\theta}_c$.
  - Sampling using the cumulative distribution function (CDF).
- We discussed Monte Carlo for approximating expectations.
  - Generate samples from a model.
  - Compute the average function value on the samples.
- We discussed conjugate priors.
  - For a given likelihood, a prior that leads to posterior in “family” of prior.
  - Conjugate prior for categorical distribution is the Dirichlet distribution.
    - Dirichlet gives a “probability over discrete probabilities”.

End of Part 2 ("Categorical Variables"): Key Concepts

- We reviewed standard conditional independence assumptions:
  - Data is IID [given parameters].
  - Data is independent of hyper-parameters given parameters.
  - Discriminative models assume parameters are independent of features.
- We discussed Bayesian learning:
  - Instead of using a single parameter, sum/integrate over all parameters.
  - Prediction using the posterior predictive distribution.
    - And possibly a cost function for Bayesian decision theory.
  - Very-strong protection against overfitting.
- We discussed empirical Bayes:
  - Optimize hyper-parameters using the marginal likelihood.
  - Can optimize a large number of hyper-parameters, without a validation set.
- We discussed hierarchical Bayes:
  - Putting a prior on the prior, which we used to model non-IID grouped data.

End of Part 2 (“Categorical Variables”): Key Concepts

- We discussed multi-class classification.
  - Categorical generalization of sigmoid function is the softmax function.
- We discussed multi-class neural networks.
  - Put softmax on the last layer.
  - Other layers can stay the same, and the same tricks are used/needed.
- We discussed “what have we learned”.
  - Layers in CNNs seem to be doing something sensible.
  - But ML models are easily fooled in various ways.
  - And ML models can have harmful biases.

End of Part 2 (“Categorical Variables”): Key Concepts

- We discussed recurrent neural networks (RNNs).
  - Use tied parameters across time to model sequences of different lengths.
    - Makes vanishing/exploding gradient and “forgetting” problems worse.
  - Sequence-to-sequence handles output sequences of unknown lengths.
  - Multi-modal learning considers input and output of different formats.
- We discussed long short term memory (LSTM) models.
  - Include memory cells that are read/written/cleared with gates.
  - Allows modeling longer-range dependencies than standard RNNs.
- We discussed attention.
  - Allows decoder to access information from all encoding steps.
- We discussed transformers.
  - “Fully-connected” attention that forms basis for many modern methods.

Next Topic: Gaussian Density Estimation

Motivating Problem: Cell Phone Battery Life

- Consider modeling battery life between charges:
  - It makes sense to view this as a continuous quantity.
    - Rather than a fixed set of values, the battery life could be any real number.

- Reviews/advertisements will often advertise estimates:

If you want the longest battery life, the iPhone 13 Pro Max is the one to get. In our battery test, the iPhone 13 Pro Max streamed a continuous video at full screen brightness for a whopping **20 hours and 18 minutes**. Nov 11, 2021

https://www.businessinsider.com › ... › Tech › Smartphones

iPhone 13 Pro Max Review: Longest Battery Life and Biggest ...

- We want to find the full distribution over charging times.
  - So we can solve real-world problems like:
    - “If I have not charged for 18 hours, what is the probability I will make it to 21 hours?”

General Problem: Continuous Density Estimation

- We can view this as density estimation with a continuous variable:
  - Input: 'n' IID samples of continous values x^1, x^2, x^3,…, x^n from a population.
  - Output: model of probability density for any real number 'x'.
- Continuous density estimation as a picture:

X =

Battery Life
20h18m00s
16h53m42s
21h03m50s
17h33m13s
27h46m28s

p(x = 20h18m02s) = 0.8

- Watch out: we are estimating the density here, not the probability.
  - We could have $p(x) > 1$.
  - You would get probabilities for doing integrals of the density over intervals.

Other Applications

- Other applications where continuous density estimation is useful:
  - Modeling sizes (size of food grown in field, birthweight of babies).
  - Modeling times or control values in a manufacturing process.
  - Modeling stock variations or income distributions.
  - Modeling continuous medical measurements (blood pressure).
  - Modeling grades.

- Even with 1 variable there are many possible distributions.
  - More complicated than binary/categorical.

- We first consider the simple case were we assume data is Gaussian.
  - Also known as a "normal" distribution.

Univariate Gaussian

- The Gaussian probability density has the form:

$$p(x^i \mid \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x^i-\mu)^2}{2\sigma^2}\right)$$

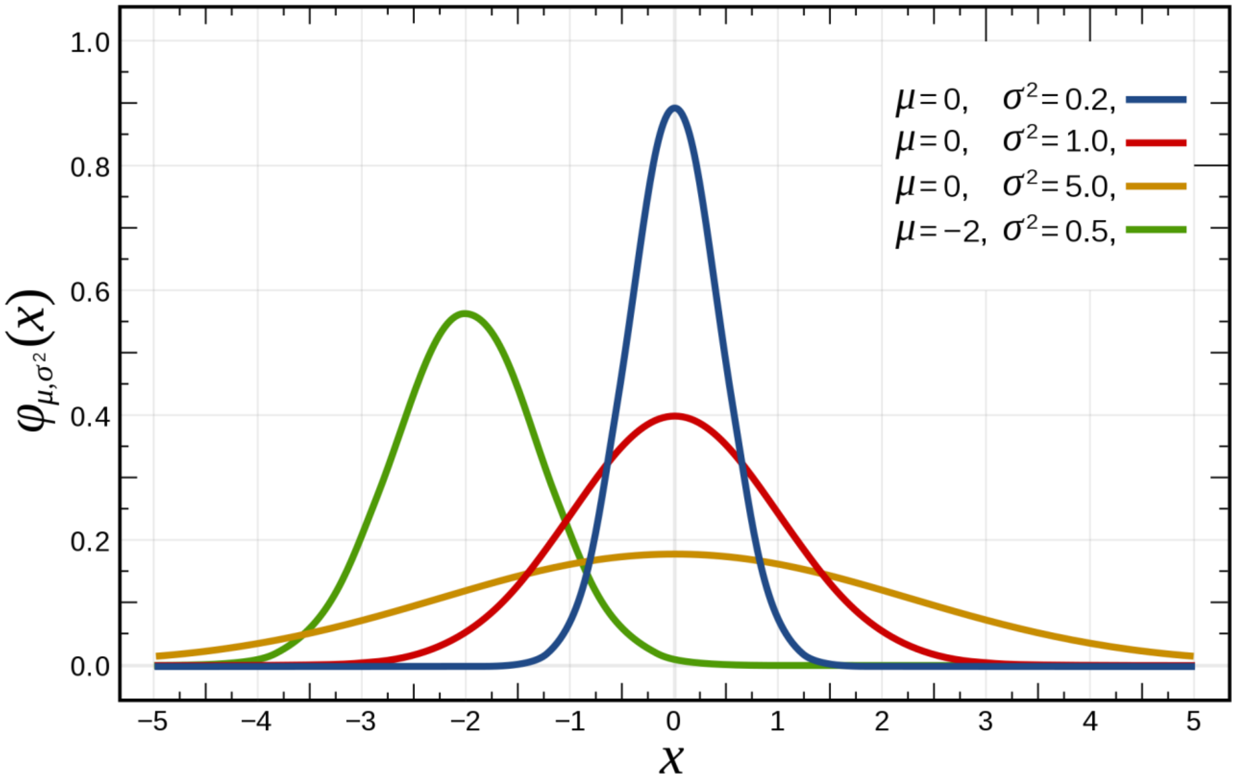
  - The mean parameter μ can be any real number.
  - The standard deviation σ can be any positive number.
    - We call σ^2 the variance.
    - Gaussians are also known as normal distributions.

- If we assume x^i follows a Gaussian distribution, we often write:

$$x^i \sim \mathcal{N}(\mu, \sigma^2)$$

"x^i is generated from a normal distribution with mean μ and variance σ^2"

Univariate Gaussian



- Mean parameter μ controls location of center of density.
- Variance parameter σ^2 controls how spread out density is.
  - As $\sigma \to 0$ you get a "spike" at the mean, as $\sigma \to \infty$ you get uniform.

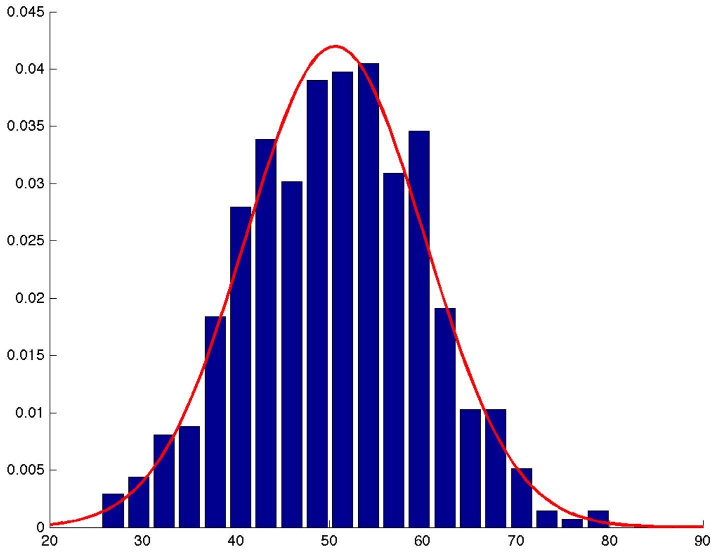
https://en.wikipedia.org/wiki/Normal_distribution

Motivation for Gaussian

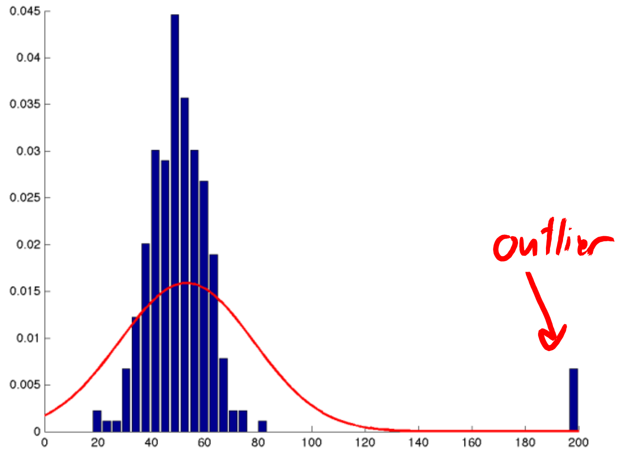
- Why use the Gaussian distribution?
  - Data might actually follow Gaussian.
    - Good justification if true, but usually false.
  - Central limit theorem: many sums of random variables converge* to Gaussian.
    - Usually a bad justification: does not imply data distribution converges to a Gaussian.
      - You would have to argue that your data comes from an asymptotic process where CLT applies.
  - Distribution with maximum entropy that fits mean and variance of data.
    - "Makes the least assumptions" while matching the mean and variance of data.
      - We will discuss this later when we discuss the "exponential family".
    - But for complicated problems, just matching means and variances is not enough.
  - Makes many computations and doing theory much easier.
    - The same reason we use a lot of the common distributions.
    - Sometimes Gaussians are "good enough to be useful".
    - Gaussians are common "building blocks" in more-advanced methods.

Motivations for not using Gaussians

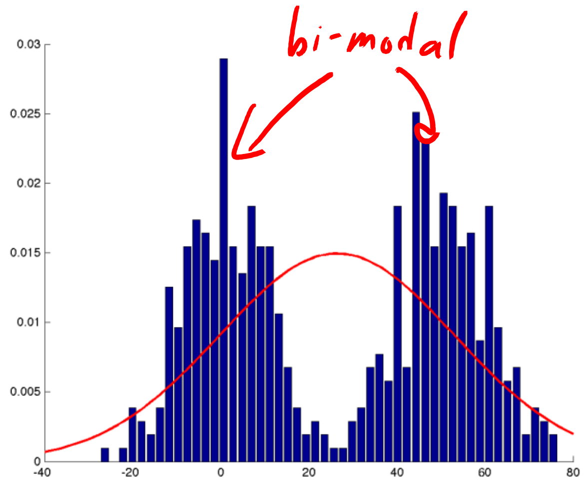
- Histogram of x^i values with red line being MLE Gaussian density:



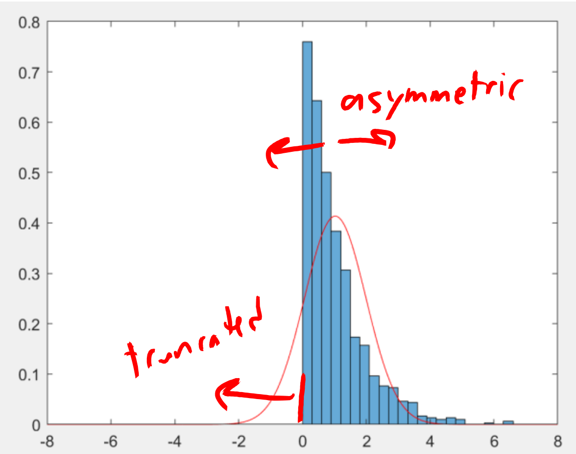
Symmetric around mean, untruncated, no outliers, uni-modal =>



Sensitive to outliers



Cannot model multiple modes



assumes symmetric and not truncated

- Grades usually have all these issues.

Next Topic: Gaussian Inference and Learning

Inference in Univariate Gaussians

- Decoding: find 'x' that maximizes the PDF p(x | μ, σ^2).
  - The decoding is given by the mean μ.
- Computing likelihood of an IID dataset:

$$p(X \mid \mu, \sigma^2) = \prod_{i=1}^{n} p(x^i \mid \mu, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x^i-\mu)^2}{2\sigma^2}\right) = \frac{1}{(\sigma\sqrt{2\pi})^n} \prod_{i=1}^{n} \exp\left(-\frac{(x^i-\mu)^2}{2\sigma^2}\right)$$

$$= \frac{1}{(\sigma\sqrt{2\pi})^n} \exp\left(-\frac{\sum_{i=1}^{n}(x^i-\mu)^2}{2\sigma^2}\right)$$

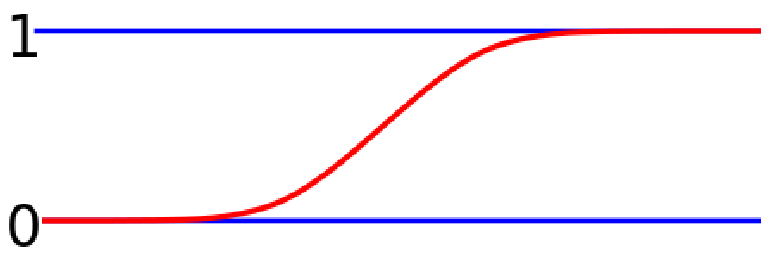
  - Not that the likelihood is a density and not a probability.
- Computing probability that an 'x' lies in an interval:

$$\text{prob}(a \leq x \leq b \mid \mu, \sigma^2) = \int_a^b p(x \mid \mu, \sigma^2)dx = \underbrace{\text{prob}(x \leq b \mid \mu, \sigma^2)}_{\text{CDFs}} - \underbrace{\text{prob}(x \leq a \mid \mu, \sigma^2)}_{\text{CDFs}}$$

  - If a=b this is zero, so any 'x' has probability zero.

Cumulative Distribution Function (CDF)

- We often use F(c) = prob(x ≤ c) = $\int_{-\infty}^{c} p(x)$ to denote the CDF.
  - F(c) is between 0 and 1, giving proportion of times ‘x’ is below ‘c’.
  - F(c) monotonically increases with ‘c’.



- The Gaussian CDF is given by: $F(c) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{c-\mu}{\sigma\sqrt{2}}\right)\right]$
  - Where the “error function” erf is computed numerically and given by:

$$\operatorname{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt$$

Sampling with the Inverse CDF ("Quantile") Function

- How can we sample from a continuous density?
- We want to write a function that takes a uniform sample and:
  - 50% of the time it returns a sample in the region where F(c)= 50%.
  - 25% of the time it returns a sample in the region where F(c) = 25%.
  - 75% of the time it returns a sample in the region where F(c) = 75%.
  - 10% of the time it returns a sample in the region where F(c) = 10%.
  - And so on, so the CDF F(c) divides up the interval [0,1].
- The function we want is the inverse of the CDF F^{-1} ("quantile" function):
  - $F^{-1}(u) = c$ for the unique 'c' where $F(c) = u$.
  - Allows sampling from Gaussians and using Monte Carlo with Gaussians.

Inverse Transform Method (Exact 1D Sampling)

- Inverse transform method for exact sampling of a continuous density in 1D:
  1. Sample 'u' uniformly between 0 and 1.
  2. Return $F^{-1}(u)$.

- For Gaussians, we have $F^{-1}(u) = \mu + \sigma\sqrt{2}\,\mathrm{erf}^{-1}(2u - 1)$.
  - Formula will convert uniform 'u' values into sample from a Gaussian.
  - To sample a N(0,1) distribution as in the "randn()" function, use "sqrt(2)*erfinv(2*rand()-1)".

- Showing that CDF of samples has CDF we want to sample from (for invertible 'F'):

$$
\begin{aligned}
\mathrm{prob}(\text{sample} \leq c) &= \mathrm{prob}(F^{-1}(u) \leq c) && \text{(sample is given by } F^{-1}(u)\text{)} \\
&= \mathrm{prob}(F(F^{-1}(u)) \leq F(c)) && \text{(apply strictly-monotonic 'F' to inequality)} \\
&= \mathrm{prob}(u \leq F(c)) && \text{(F and } F^{-1} \text{ are inverses)} \\
&= F(c) && \text{(}\mathrm{prob}(u \leq y) = y \text{ for uniform 'u')}
\end{aligned}
$$

  - So after the inverse transform, we have the CDF of the distribution we want.
- Video on pseudo-randomness and inverse-transform sampling.

MLE for Univariate Gaussian

- We showed that the likelihood for 'n' IID examples is given by:

$$p(X \mid \mu, \sigma^2) = \frac{1}{(\sigma\sqrt{2\pi})^n} \exp\left(-\frac{\sum_{i=1}^{n}(x^i-\mu)^2}{2\sigma^2}\right)$$

- To compute the MLE, minimize the NLL (which is convex):

$$-\log p(X \mid \mu, \sigma^2) = n \log \sigma + \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x^i-\mu)^2 + \text{constant}$$

- Setting derivative with respect to μ to 0 gives MLE of: $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x^i$
  - So MLE for the mean is the mean of the samples.
- Plugging in $\hat{\mu}$ and setting derivative with respect to σ to 0 gives: $\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x^i-\hat{\mu})^2$
  - So MLE for the variance is the variance of the samples.
    - Unless all x^i are equal (then NLL is not bounded below and MLE does not exist).

Conjugate Prior and Posterior for Mean

- For fixed variance, conjugate prior for mean is Gaussian.

If each $x^i \sim \mathcal{N}(\mu, \sigma^2)$ and $\mu \sim \mathcal{N}(m, v)$, then $\mu \mid x^1, x^2, \dots, x^n \sim \mathcal{N}(\tilde{m}, \tilde{v})$

where $\tilde{m} = \frac{vn}{vn + \sigma^2}\hat{\mu}_{MLE} + \frac{\sigma^2}{vn + \sigma^2} m$ and $\tilde{v} = \left(\frac{n}{\sigma^2} + \frac{1}{v}\right)^{-1}$

  - "Self conjugacy" is a **very special** property (a key to usefulness of Gaussians).
    - Derived by using '∝' and "completing the square" in exponent (see notes on webpage).
  - Formulas look a bit weird, but consider $\tilde{m}$ and $\tilde{v}$ change as 'n' grows:
    - As 'n' grows posterior mean $\tilde{m}$ converges from prior mean m towards MLE.
    - As 'n' grows posterior variance $\tilde{v}$ converges from prior variance v down to 0.
  - MAP estimate is given by $\tilde{m}$ (it has the highest PDF of the posterior).
  - Posterior predictive is also given by a Gaussian (not obvious, see notes linked on webpage).
    - With mean $\tilde{m}$ and variance $\tilde{v} + \sigma^2$.
    - For complicated Bayeisan inference tasks, can use Monte Carlo by sampling from Gaussian posterior.

- We will come back to MAP/Bayes estimation for variance later.

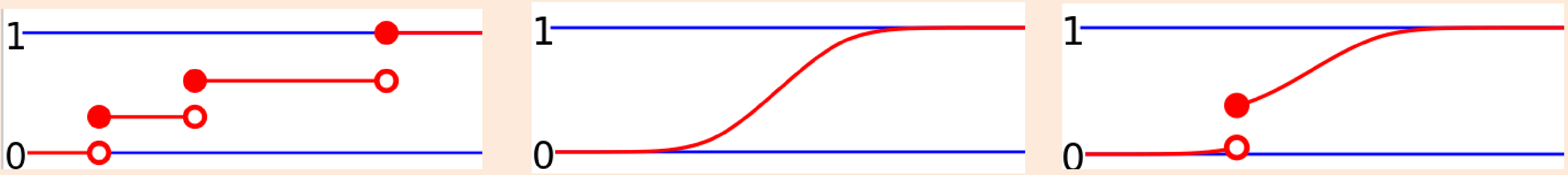
Summary

- Gaussian density estimation:
  - Modeling continuous variable samples, assuming it follows a Gaussian.
  - We use Gaussians because they have lots of nice properties.
  - But Gaussians assume symmetric, no outliers, no truncation, uni-modal.
- Mean and variance parameterization of Gaussians:
  - Mean specifies center of distribution.
  - Variance specifies spread of distribution.
- Inverse transform method for sampling:
  - Apply the “inverse” of the CDF to uniform samples to generate samples.
- MLE and MAP for Gaussians:
  - MLE is given by mean and variance of samples.
  - Conjugate prior for mean is another Gaussian.
    - MAP moves between mean of samples and prior mean.
    - Posterior predictive is also Gaussian in this case.

- Next time: more about Gaussians than you ever wanted to know.

Cumulative Distribution Function (CDF)

- CDF can be used for discrete and continuous variables (and mixed).



- We can generalize the quantile function to non-invertible case.

https://en.wikipedia.org/wiki/Cumulative_distribution_function

Quantile Function – Non-Invertible Case

- If the CDF 'F' is not invertible, we define the quantile F^{-1} as:

$$F^{-1}(u) = \inf\{c \mid F(c) \geq u\}$$

- "Smallest value 'c' such that F(c) is bigger than u."
  - See notes on max and argmax if you have not seen 'inf' before.
    - It's a variant on 'min' that is defined in more cases.

- If 'F' is invertible at this 'c', this gives the usual inverse.
  - But this more-general definition handles non-invertible points.
    - For example, the CDF is not invertible for categorical variables at the "jumps" in CDF.
      - Many values of 'u' are mapped to by the same 'c'.