Uncertainty Quantification in Machine Learning

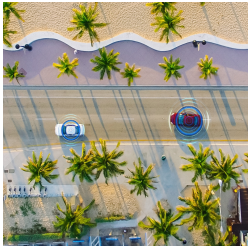
Trustworthy Machine Learning

Stephan Rabanser stephan@cs.toronto.edu

May 1, 2024

Motivation

Machine Learning systems are becoming ubiquitous.

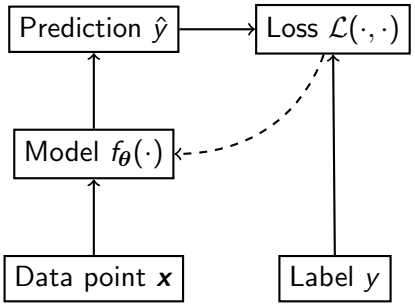



Especially in high-stakes decision-making, it is of vital importance to quantify our uncertainty in the predictions we make.

Image credit: `unsplash.com`

Supervised Learning Recap

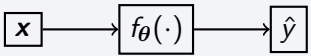
- Dataset $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ where $(\boldsymbol{x}, y) \sim p$ over $\mathcal{D} = \mathcal{X} \times \mathcal{Y}$ with $\boldsymbol{x} \in \mathcal{X}$, $y \in \mathcal{Y}$.
- Prediction function $f_{\boldsymbol{\theta}}(\boldsymbol{x}) : \mathcal{X} \to \mathcal{Y}$ producing labels $\hat{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x})$ with $f_{\boldsymbol{\theta}}(\cdot) \in \mathcal{H}$.
- Loss function $\mathcal{L}(\hat{y}, y)$ measuring prediction quality of $f_{\boldsymbol{\theta}}(\cdot)$.



How can we quantify uncertainty in the prediction $\hat{y}$?
We need to output a probability distribution $p_\theta(y|\boldsymbol{x})$, not just a single $\hat{y}$!

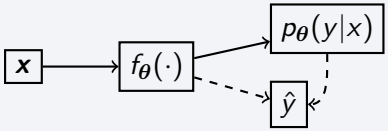
Labelling Models vs Probabilistic Models

Labelling Models



- Directly provides a label $\hat{y}$:
  - Regression: $\hat{y} \in \mathbb{R}$
  - Classification: $\hat{y} \in \{1, \ldots, C\}$
- Do not provide a measure of confidence, just a decision.
- Typically not suitable for uncertainty quantification.

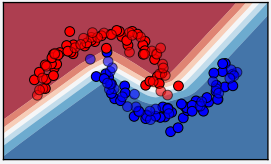
Probabilistic Models



- Provide a measure of confidence $p_\theta(y|x)$ alongside a decision $\hat{y}$.
- Enables uncertainty quantification.
- Can be turned into labelling models by hiding confidence score.

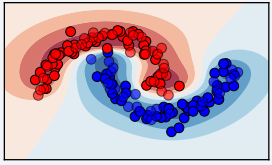
Discriminative vs Generative Models

Discriminative (Conditional) Models



- Aim at identifying and approximating the discriminatory boundary.
- Maximize conditional: $p_{\theta}(y|\boldsymbol{x})$.
- Better raw predictive performance.

Generative Models



- Model the data distribution to gain ability to generate faithful samples.
- Maximize joint: $p_{\theta}(\boldsymbol{x}, y)$.
- Better uncertainty quantification.

Aleatoric vs Epistemic Uncertainty

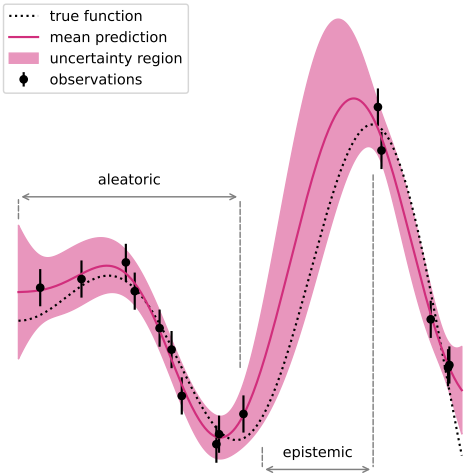
$$\text{Uncertainty} = \text{AU} + \text{EU}$$

Aleatoric Uncertainty (AU)

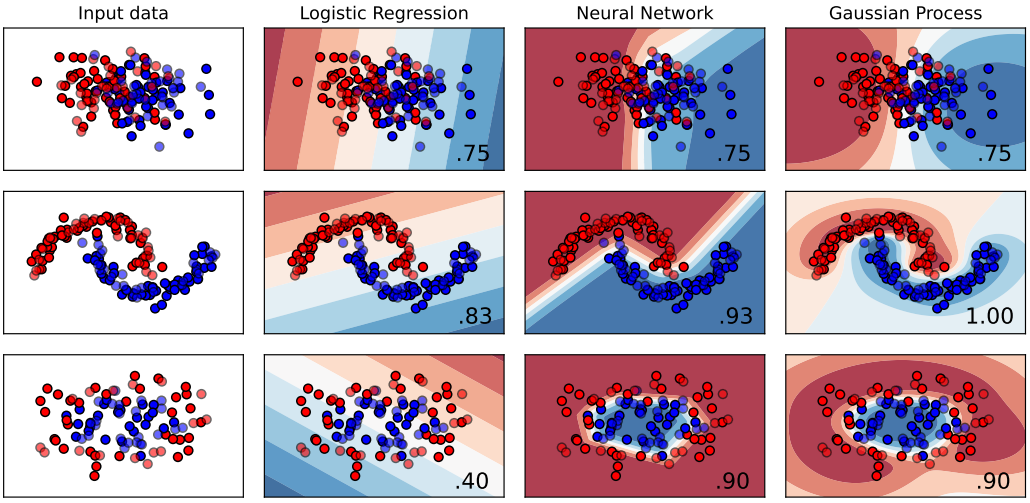
- Uncertainty within the data
- Irreducible with more data
- Analogy: Bayes error

Epistemic Uncertainty (EU)

- Uncertainty away from data
- Reducible with more data
- Analogy: approximation error



Bayes Error vs Approximation Error



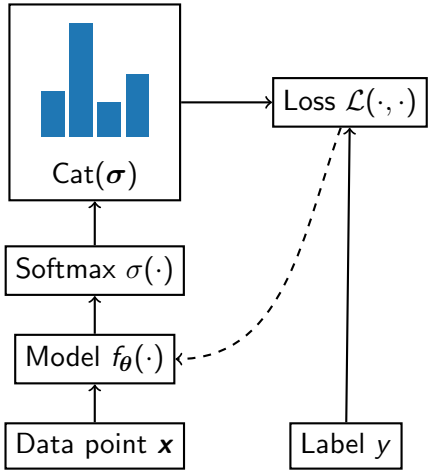
Uncertainty in Classification

Softmax cross-entropy models are already probabilistic models!

- Classification are often trained using the softmax cross-entropy (CE) loss.
- The model's logits are mapped through the softmax function:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \qquad \begin{array}{l} \sum_i \sigma(\mathbf{z})_i = 1 \\ 0 \leq \sigma(\mathbf{z})_i \leq 1 \end{array}$$

- The CE loss measures agreement of label and the categorical distribution.

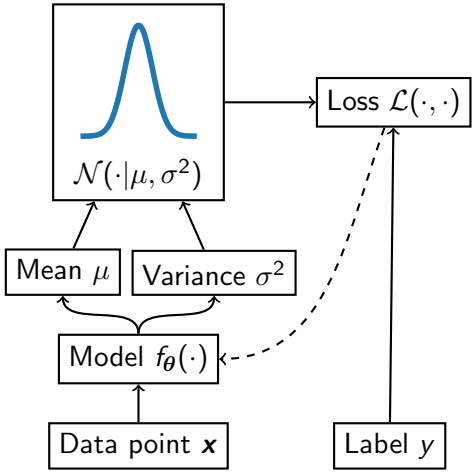


Uncertainty in Regression

Regression models need to be turned into probabilistic models!

- Regression models are often trained with squared error loss (not prob.).
- Instead model the output as a conditional Gaussian distribution parameterized by
  - Predictive mean μ,
  - Predictive variance σ^2.
- Define loss as negative log likelihood:

$$\mathcal{L} = -\log \mathcal{N}(y|\mu, \sigma^2)$$



Evaluation of Uncertainties

Negative Log Likelihood

- Commonly used to evaluate the quality of uncertainty on some test set.

$$\text{NLL} = -\log p(y|x)$$

- Can over-emphasize tail probabilities.
- Is a **proper scoring rule**: pred. probabilities match true probabilities exactly.

Brier Score

- Measures squared error of predicted probabilities and true one-hot labels.

$$\text{BS} = \frac{1}{|\mathcal{Y}|} \sum_{y' \in \mathcal{Y}} \left(p_\theta(y'|\boldsymbol{x}) - y_{\text{OH}}\right)^2$$

- Insensitive to predicted probabilities of infrequent events (class imbalance).
- Is a proper scoring rule.

Expected Calibration Err.

- Measures alignment between predicted probabilities and accuracy in distinct confidence buckets.

$$\text{ECE} = \sum_{s=1}^{S} \frac{|B_s|}{N} |\text{acc}(B_s) - \text{conf}(B_s)|$$

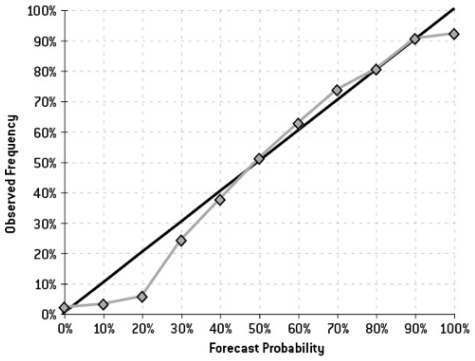
$$B_s = \{n \,|\, p_\theta(y|x) \in [\rho_s, \rho_s + 1]\}$$

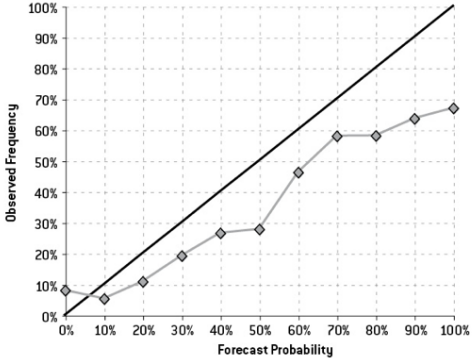
- Not a proper scoring rule, binning might cause non-monotonic increase.

Predictive Entropy

Measures prediction surprise; not a proper scoring rule. $H = -\sum_{y' \in \mathcal{Y}} p(y'|\boldsymbol{x}) \log p(y'|\boldsymbol{x})$

Calibration





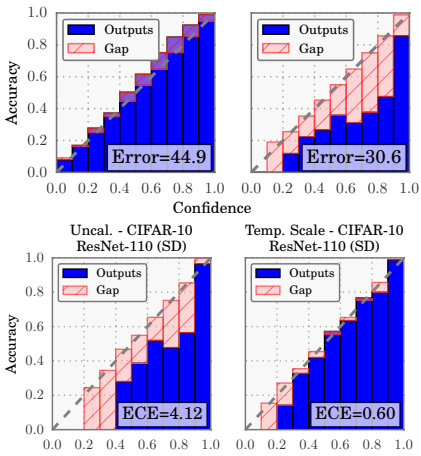
The frequency of predicted events should match the truly observed frequency of events.

Calibration: Temperature Scaling

- A classification network predicts $\sigma(\boldsymbol{z})$:

$$\sigma(\boldsymbol{z})_k = \frac{e^{z_k}}{\sum_{k'} e^{z_{k'}}}$$

- Replace $\sigma(\boldsymbol{z})$ with $\sigma(\boldsymbol{z}/T)$ where $T \in \mathbb{R}_+$ is called the temperature.
- T is tuned to minimize NLL (a proper scoring rule) on a validation set.
- As a result, the algorithm is incentivized to match the true probabilities as closely as possible.

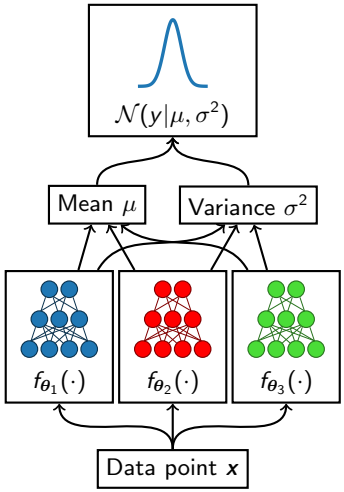


(Deep) Ensembling

- Ensembling is a classical technique from frequentist statistics to bootstrap predictive variance.
- Don't train a single model, train multiple!
- Randomness of data selection and hyperparameters.
- The predictions from these sub-models then yield:

$$\hat{y} = \mu = \frac{1}{M} \sum_{m=1}^{M} f_{\boldsymbol{\theta}_m}(\boldsymbol{x})$$

$$\text{Var}[\hat{y}] = \sigma^2 = \frac{1}{M-1} \sum_{m=1}^{M} (f_{\boldsymbol{\theta}_m}(\boldsymbol{x}) - \hat{y})^2$$

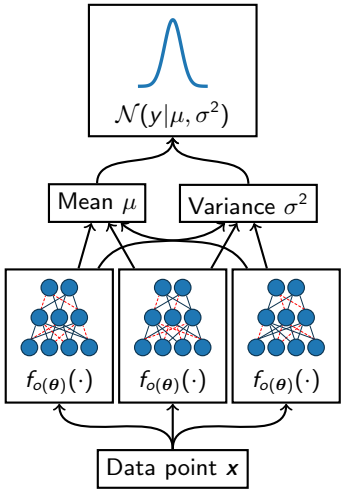


Monte Carlo (MC) Dropout

- Dropout is typically used as a regularization technique during training time.
- Apply the dropout function $o(\boldsymbol{\theta})$ at test time to yield a collection of M sparse sub-models.
- This is cheaper than training multiple models.
- The predictions from these sub-models then yield:

$$\hat{y} = \mu = \frac{1}{M} \sum_{m=1}^{M} f_{o(\boldsymbol{\theta})}(\boldsymbol{x})$$

$$\text{Var}[\hat{y}] = \sigma^2 = \frac{1}{M-1} \sum_{m=1}^{M} (f_{o(\boldsymbol{\theta})}(\boldsymbol{x}) - \hat{y})^2$$



Bayesian Parameter Estimation: Likelihood

- Motivating example: estimating the parameter of a biased coin
  - You flip a coin 100 times. It lands heads $N_H = 55$ and tails $N_T = 45$ times.
  - What is the probability it will come up heads if we flip again?
- Model: observations x_i are independent and identically distributed (i.i.d.) Bernoulli random variables with parameter θ.
- The likelihood function is the probability of the observed data (the entire sequence of H's and T's) as a function of θ:

$$\begin{aligned} L(\theta) = p(\mathcal{D}) &= \prod_{i=1}^{N} \theta^{x_i}(1-\theta)^{1-x_i} \\ &= \theta^{N_H}(1-\theta)^{N_T} \end{aligned}$$

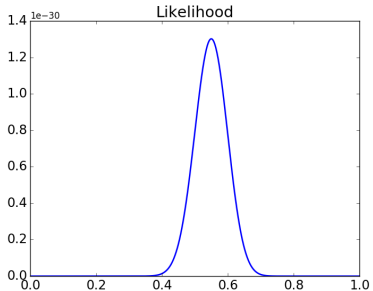
- N_H and N_T are sufficient statistics.

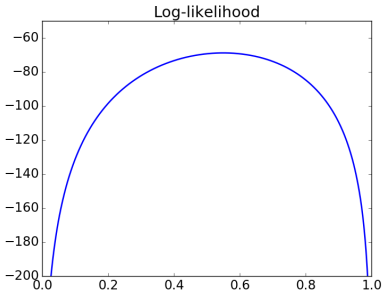
Bayesian Parameter Estimation: Likelihood (cont'd)

- The likelihood is generally very small, so it's often convenient to work with log-likelihoods.

$$L(\theta) = \theta^{N_H}(1-\theta)^{N_T} \approx 7.9 \times 10^{-31}$$

$$\ell(\theta) = \log L(\theta) = N_H \log \theta + N_T \log(1-\theta) \approx -69.31$$





Bayesian Parameter Estimation: Maximum Likelihood

- Good values of θ should assign high probability to the observed data. This motivates the maximum likelihood criterion.
- Solve by setting derivatives to zero:

$$\begin{aligned}\frac{\mathrm{d}\ell}{\mathrm{d}\theta} &= \frac{\mathrm{d}}{\mathrm{d}\theta}\left(N_H \log\theta + N_T \log(1-\theta)\right) \\ &= \frac{N_H}{\theta} - \frac{N_T}{1-\theta}\end{aligned}$$

- Setting this to zero gives the maximum likelihood estimate:

$$\hat{\theta}_{\mathrm{ML}} = \frac{N_H}{N_H + N_T},$$

- Normally there's no analytic solution, and we need to solve an optimization problem (e.g. using gradient descent).

Bayesian Parameter Estimation: Maximum Likelihood (cont'd)

- Maximum likelihood has a pitfall: if you have too little data, it can overfit.
- E.g., what if you flip the coin twice and get H both times?

$$\theta_{\mathrm{ML}} = \frac{N_H}{N_H + N_T} = \frac{2}{2+0} = 1$$

- But even a fair coin has 25% chance of showing this result.
- Because it never observed T, it assigns this outcome probability 0. This problem is known as data sparsity.
- If you observe a single T in the test set, the likelihood is $-\infty$.

Bayesian Parameter Estimation: Beyond Maximum Likelihood

- In maximum likelihood, the observations are treated as random variables, but the parameters are not.
- The Bayesian approach treats the parameters as random variables as well.
- To define a Bayesian model, we need to specify two distributions:
  - The prior distribution $p(\boldsymbol{\theta})$, which encodes our beliefs about the parameters *before* we observe the data
  - The likelihood $p(\mathcal{D} \mid \boldsymbol{\theta})$, same as in maximum likelihood
- When we update our beliefs based on the observations, we compute the posterior distribution using Bayes' Rule:

$$p(\boldsymbol{\theta} \mid \mathcal{D}) = \frac{p(\boldsymbol{\theta}) p(\mathcal{D} \mid \boldsymbol{\theta})}{\int p(\boldsymbol{\theta}') p(\mathcal{D} \mid \boldsymbol{\theta}') \, \mathrm{d}\boldsymbol{\theta}'}.$$

- We rarely ever compute the denominator explicitly due to intractability.

Bayesian Parameter Estimation: The Prior Distribution

- Let's revisit the coin example. We already know the likelihood:

$$L(\theta) = p(\mathcal{D}) = \theta^{N_H}(1-\theta)^{N_T}$$

- It remains to specify the prior $p(\theta)$.
  - We can choose an uninformative prior, which assumes as little as possible. A reasonable choice is the uniform prior.
  - But our experience tells us 0.5 is more likely than 0.99. One particularly useful prior that lets us specify this is the beta distribution:

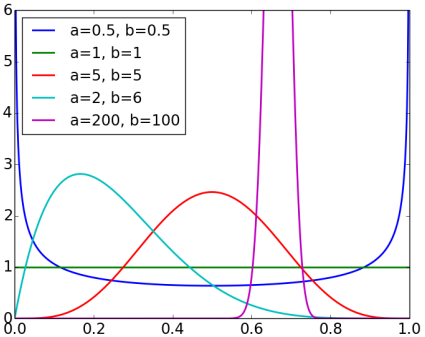
$$p(\theta; a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\theta^{a-1}(1-\theta)^{b-1}.$$

  - This notation for proportionality lets us ignore the normalization constant:

$$p(\theta; a, b) \propto \theta^{a-1}(1-\theta)^{b-1}.$$

Bayesian Parameter Estimation: The Prior Distribution (cont'd)

Beta distribution for various values of a, b:



- Some observations:
  - The expectation $\mathbb{E}[\theta] = a/(a+b)$.
  - The distribution gets more peaked when a and b are large.
  - The uniform distribution is the special case where $a = b = 1$.
- The main use-case for the beta distribution is as a prior for the Bernoulli distribution.

Bayesian Parameter Estimation: The Posterior Distribution

- Computing the posterior distribution:

$$\begin{aligned} p(\boldsymbol{\theta} \mid \mathcal{D}) &\propto p(\boldsymbol{\theta}) p(\mathcal{D} \mid \boldsymbol{\theta}) \\ &\propto \left[\theta^{a-1}(1-\theta)^{b-1}\right]\left[\theta^{N_H}(1-\theta)^{N_T}\right] \\ &= \theta^{a-1+N_H}(1-\theta)^{b-1+N_T}. \end{aligned}$$

- This is just a beta distribution with parameters $N_H + a$ and $N_T + b$.
- The posterior expectation of θ is:

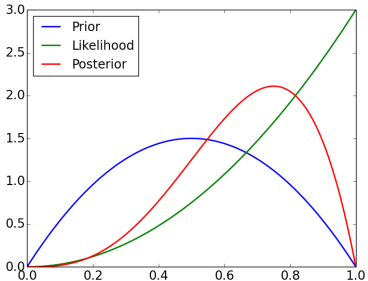
$$\mathbb{E}[\theta \mid \mathcal{D}] = \frac{N_H + a}{N_H + N_T + a + b}$$

- The parameters a and b of the prior can be thought of as pseudo-counts.
  - The reason this works is that the prior and likelihood have the same functional form. This phenomenon is known as conjugacy.

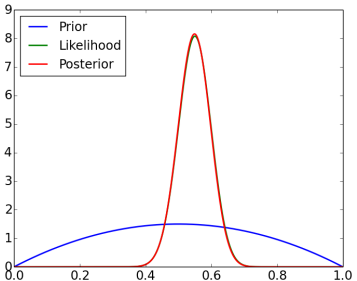
Bayesian Parameter Estimation: The Posterior Distribution (cont'd)

Bayesian inference for the coin flip example:

Small data setting
$N_H = 2,\ N_T = 0$



Large data setting
$N_H = 55,\ N_T = 45$



When you have enough observations, the data overwhelm the prior.

Bayesian Parameter Estimation: Maximum A-Posteriori

- What do we actually do with the posterior?
- Maximum a-posteriori (MAP) estimation: find the most likely parameter settings under the posterior
- This converts the Bayesian parameter estimation problem into a maximization problem

$$
\begin{aligned}
\hat{\boldsymbol{\theta}}_{\mathrm{MAP}} &= \arg\max_{\boldsymbol{\theta}} \; p(\boldsymbol{\theta} \mid \mathcal{D}) \\
&= \arg\max_{\boldsymbol{\theta}} \; p(\boldsymbol{\theta}, \mathcal{D}) \\
&= \arg\max_{\boldsymbol{\theta}} \; p(\boldsymbol{\theta})\, p(\mathcal{D} \mid \boldsymbol{\theta}) \\
&= \arg\max_{\boldsymbol{\theta}} \; \log p(\boldsymbol{\theta}) + \log p(\mathcal{D} \mid \boldsymbol{\theta})
\end{aligned}
$$

Bayesian Parameter Estimation: Maximum A-Posteriori (cont'd)

- Joint probability in the coin flip example:

$$\begin{aligned}\log p(\theta, \mathcal{D}) &= \log p(\theta) + \log p(\mathcal{D} \mid \theta) \\ &= \text{const} + (a-1)\log\theta + (b-1)\log(1-\theta) + N_H \log\theta + N_T \log(1-\theta) \\ &= \text{const} + (N_H + a - 1)\log\theta + (N_T + b - 1)\log(1-\theta)\end{aligned}$$

- Maximize by finding a critical point

$$0 = \frac{\mathrm{d}}{\mathrm{d}\theta} \log p(\theta, \mathcal{D}) = \frac{N_H + a - 1}{\theta} - \frac{N_T + b - 1}{1 - \theta}$$

- Solving for θ,

$$\hat{\theta}_{\mathrm{MAP}} = \frac{N_H + a - 1}{N_H + N_T + a + b - 2}$$

Bayesian Parameter Estimation: (Posterior) Predictive Distribution

- The posterior predictive distribution is the distribution over future observables given the past observations. We compute this by marginalizing out the parameter(s):

$$p(\mathcal{D}' \mid \mathcal{D}) = \int p(\boldsymbol{\theta} \mid \mathcal{D}) p(\mathcal{D}' \mid \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{\theta}.$$

- For the coin flip example:

$$\begin{aligned}
\theta_{\text{pred}} &= \Pr(\mathbf{x}' = H \mid \mathcal{D}) \\
&= \int p(\theta \mid \mathcal{D}) \Pr(\mathbf{x}' = H \mid \theta) \, \mathrm{d}\theta \\
&= \int \mathrm{Beta}(\theta; N_H + a, N_T + b) \cdot \theta \, \mathrm{d}\theta \\
&= \mathbb{E}_{\mathrm{Beta}(\theta; N_H + a, N_T + b)}[\theta] \\
&= \frac{N_H + a}{N_H + N_T + a + b},
\end{aligned}$$

Bayesian Parameter Estimation: Convergence Properties

Comparison of estimates in the coin flip example:

	Formula	$N_H = 2, N_T = 0$	$N_H = 55, N_T = 45$
$\hat{\theta}_{\mathrm{ML}}$	$\frac{N_H}{N_H+N_T}$	1	$\frac{55}{100} = 0.55$
$\hat{\theta}_{\mathrm{MAP}}$	$\frac{N_H+a-1}{N_H+N_T+a+b-2}$	$\frac{3}{4} = 0.75$	$\frac{56}{102} \approx 0.549$
θ_{pred}	$\frac{N_H+a}{N_H+N_T+a+b}$	$\frac{4}{6} \approx 0.67$	$\frac{57}{104} \approx 0.548$

How many samples do we need for $\hat{\theta}_{\mathrm{ML}}$ to be a good estimate of θ? Use Hoeffding's Inequality for sampling complexity bound

$$p(|\hat{\theta}_{\mathrm{ML}} - \theta| \geq \varepsilon) \leq 2e^{-2N\varepsilon^2}$$

where $N = N_H + N_T$.

From Bayesian Parameter Estimation to BLR/BNN and GPs

Maximum Likelihood Estimation (MLE)

We can pick the model that maximizes the data likelihood without restrictions.

$$\boldsymbol{\theta}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta})$$

Maximum A-Posteriori Estimation (MAP)

We can incorporate prior information and regularize the model's prediction by introducing a prior $p(\boldsymbol{\theta})$ and reason about the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ using Bayes' rule.

$$\boldsymbol{\theta}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}} \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$$

From Bayesian Parameter Estimation to BLR/BNN and GPs (cont'd)

Bayesian Model Averaging / Fully Bayesian Analysis

Use predictions of all potential models and weight each model's predictions by the posterior. This leads to Bayesian Linear Regression / Bayesian Neural Networks.

$$p(y|\boldsymbol{x}, \mathcal{D}) = \int_{\boldsymbol{\theta}} p(y|\boldsymbol{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta} = \int_{\boldsymbol{\theta}} p(y|\boldsymbol{x}, \boldsymbol{\theta}) \frac{p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta})}{\int_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta}} d\boldsymbol{\theta}$$

Gaussian Process (GP)

If both the prior $p(\boldsymbol{\theta})$ and the likelihood $p(\mathcal{D}|\boldsymbol{\theta})$ are Gaussian, then the posterior predictive distribution $p(y|\boldsymbol{x}, \mathcal{D})$ is Gaussian. Hence, we can model the predictive distribution directly without explicitly performing model averaging.

$$p(y|\boldsymbol{x}, \mathcal{D}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \qquad y = f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), \kappa(\boldsymbol{x}, \boldsymbol{x}'))$$

Bayesian Linear Regression: MLE Formulation

- We can give linear regression a probabilistic interpretation by assuming a Gaussian noise model:

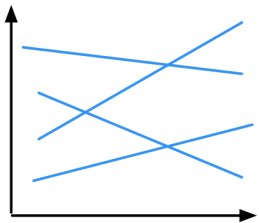
$$t \mid \mathbf{x} \sim \mathcal{N}(\mathbf{w}^\top \mathbf{x} + b,\ \sigma^2)$$

- Linear regression is just maximum likelihood under this model:

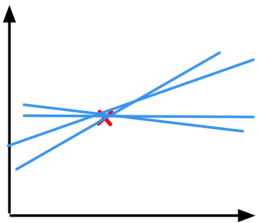
$$\begin{aligned}
\frac{1}{N}\sum_{i=1}^{N} \log p(t^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}, b) &= \frac{1}{N}\sum_{i=1}^{N} \log \mathcal{N}(t^{(i)}; \mathbf{w}^\top \mathbf{x} + b, \sigma^2) \\
&= \frac{1}{N}\sum_{i=1}^{N} \log \left[\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(t^{(i)} - \mathbf{w}^\top \mathbf{x} - b)^2}{2\sigma^2} \right) \right] \\
&= \text{const} - \frac{1}{2N\sigma^2} \sum_{i=1}^{N} (t^{(i)} - \mathbf{w}^\top \mathbf{x} - b)^2
\end{aligned}$$

Bayesian Linear Regression: Intuition

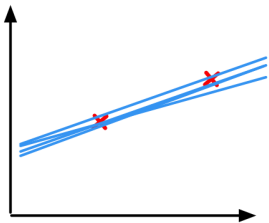
- Bayesian linear regression considers various plausible explanations for how the data points were generated.
- It makes predictions using all possible regression weights, weighted by their posterior probability.



no observations



one observation



two observations

Bayesian Linear Regression: Setup

- Leave out the bias for simplicity
- **Prior distribution:** a broad, spherical (multivariate) Gaussian centered at zero:

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \nu^2 \mathbf{I})$$

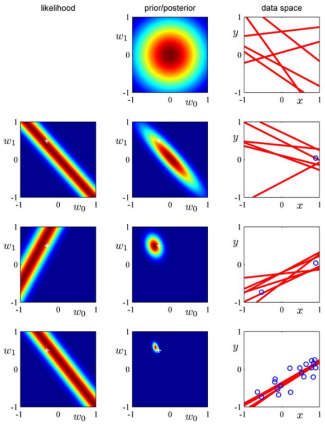
- **Likelihood:** same as in the maximum likelihood formulation:

$$t \,|\, \mathbf{x}, \mathbf{w} \sim \mathcal{N}(\mathbf{w}^\top \mathbf{x},\ \sigma^2)$$

- **Posterior:**

$$\mathbf{w} \,|\, \mathcal{D} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$\boldsymbol{\mu} = \sigma^{-2} \boldsymbol{\Sigma} \mathbf{X}^\top \mathbf{t} \qquad \boldsymbol{\Sigma}^{-1} = \nu^{-2} \mathbf{I} + \sigma^{-2} \mathbf{X}^\top \mathbf{X}$$



— Bishop, Pattern Recognition and Machine Learning

Bayesian Linear Regression: Predictive Distribution

Posterior predictive distribution:

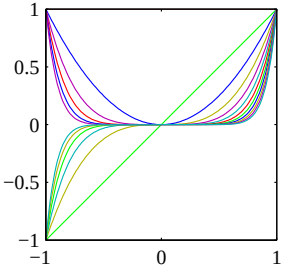
$$
\begin{aligned}
p(t \mid \mathbf{x}, \mathcal{D}) &= \int p(t \mid \mathbf{x}, \mathbf{w}) p(\mathbf{w} \mid \mathcal{D}) \, \mathrm{d}\mathbf{w} \\
&= \mathcal{N}(t \mid \boldsymbol{\mu}^\top \mathbf{x}, \sigma^2_{\text{pred}}(\mathbf{x})) \\
\sigma^2_{\text{pred}}(\mathbf{x}) &= \sigma^2 + \mathbf{x}^\top \boldsymbol{\Sigma} \mathbf{x},
\end{aligned}
$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the posterior mean and covariance of $\boldsymbol{\Sigma}$.

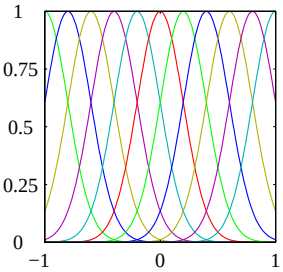
Bayesian Linear Regression: Non-Linearity via Basis Functions

- We can turn this into nonlinear regression using basis functions.

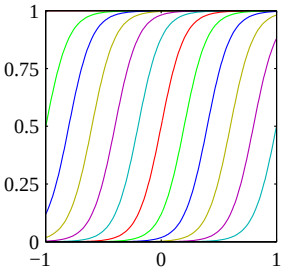
$$\phi_j(x) = x^j$$



$$\phi_j(x) = \exp\left(-\frac{(x-\mu_j)^2}{2s^2}\right)$$

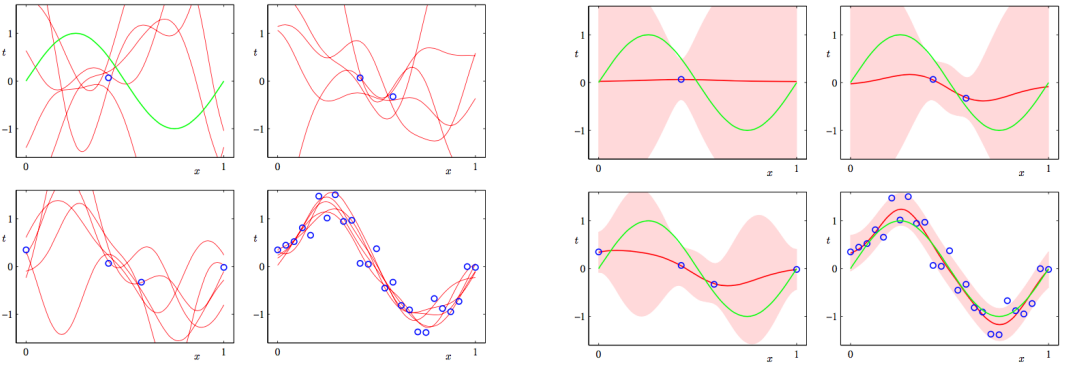


$$\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right)$$



— Bishop, Pattern Recognition and Machine Learning

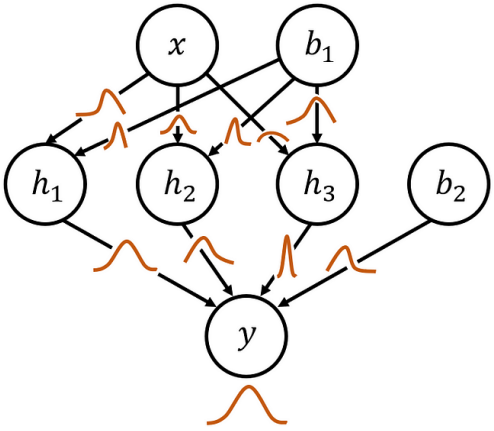
Bayesian Linear Regression: Predictive Uncertainty



— Bishop, Pattern Recognition and Machine Learning

Bayesian Neural Networks: Motivation

In addition to assuming a distribution on the output y, also assume a distribution on the parameters θ.



Bayesian Neural Networks: Computational Issues

- Computationally difficult integrals arise in Bayesian parameter estimation:
  - Marginal likelihood (needed for posterior): $p(D) = \int_{\boldsymbol{\theta}} p(D|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}$
  - Posterior predictive distribution: $p(y|\boldsymbol{x}, D) = \int_{\boldsymbol{\theta}} p(y|\boldsymbol{x}, \boldsymbol{\theta})p(\boldsymbol{\theta}|D)d\boldsymbol{\theta}$
- Approximately compute one or both of these objects!

Sampling Methods

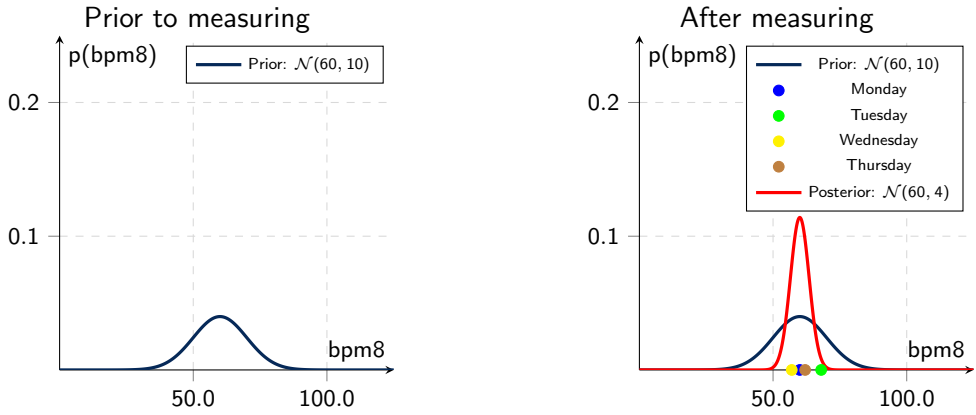
- Approximate $p(y|\boldsymbol{x}, D)$ by generating a finite parameter set $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_T\}$ whose empirical distribution matches $p(\boldsymbol{\theta}|D)$.
- Find good approx. with low T.
- Slow but asymptotically exactly recovers posterior.

Variational Inference

- Model posterior $p(\boldsymbol{\theta}|D)$ using a parameterized approximate posterior $q_\phi(\boldsymbol{\theta})$, often Gaussian.
- Iteratively improve approximation via optimization of ϕ.
- Fast but limited in functional form of $q_\phi(\boldsymbol{\theta})$.

Bayesian Parameter Estimation Example

Measure your heart rate at 8am

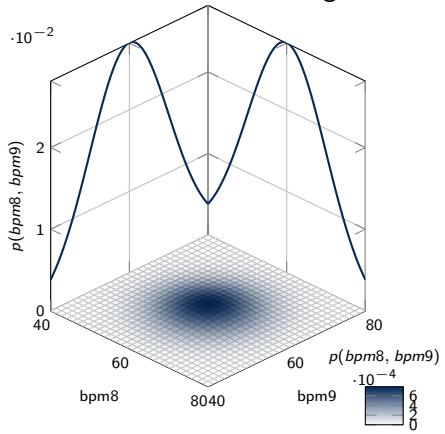


— Example from `http://videolectures.net/mlss2012_cunningham_gaussian_processes/`

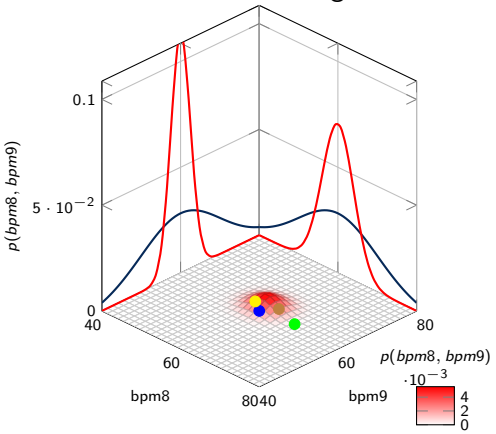
Bayesian Parameter Estimation Example

Measure your heart rate at 8am and 9am

Prior to measuring

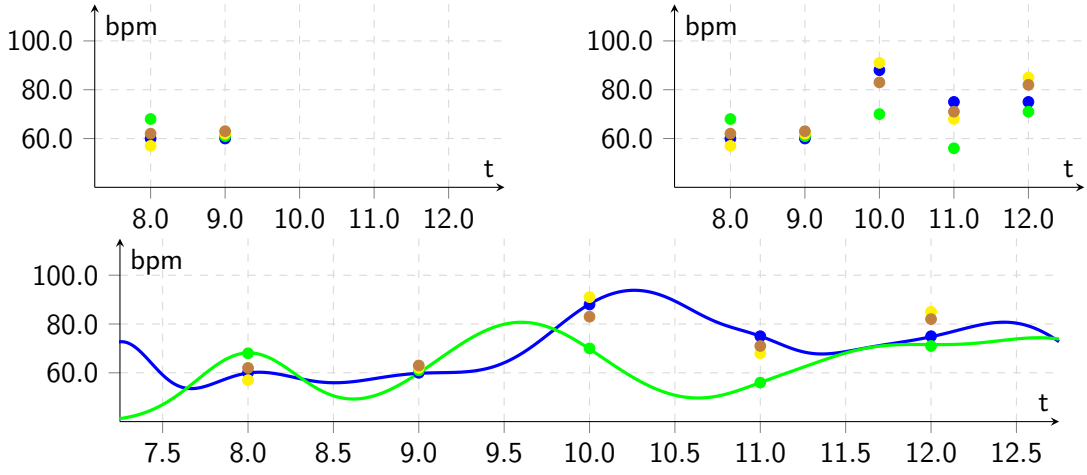


After measuring



Bayesian Parameter Estimation Example

Measuring your heart rate throughout the day



Gaussian Process: Definition

A Gaussian process describes a distribution over functions (infinitely long vectors).

- Notation: $f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), \kappa(\boldsymbol{x}, \boldsymbol{x}'))$
- Mean function: $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$
- Covariance function: $\kappa(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$

We have data points $\boldsymbol{X} = [\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_n^\top]^\top$ and are interested in their function values $f(\boldsymbol{X}) = (f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n))^\top$.

A Gaussian process is a collection of random variables, any finite number of which have joint Gaussian distribution.

$f(\boldsymbol{x})$ is one such subset and has (prior) joint Gaussian distribution.

Gaussian Process: Mean and Covariance

The mean function m

- The mean function $m(\cdot)$ encodes the a-priori expectation of the function.
- $m(\mathbf{x})$ will dominate the inference result in case we have not yet observed data similar to $\mathbf{x}$.
- Typical choice: zero-centering the data: $m(\mathbf{x}) = 0$

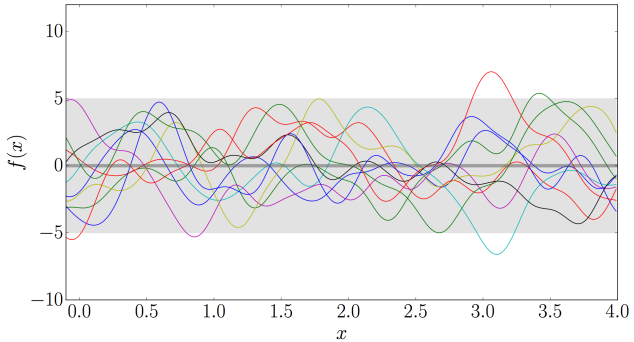
The covariance function κ

- $\kappa(\mathbf{x}, \mathbf{x}')$ measures similarity between $\mathbf{x}$ and $\mathbf{x}' \rightarrow$ similar data points have similar function values.
- κ is a Mercer kernel.
- Typical choice: squared exponential kernel: $\kappa(\mathbf{x}, \mathbf{x}') = \sigma^2 e^{-\frac{(\mathbf{x}-\mathbf{x}')^\top(\mathbf{x}-\mathbf{x}')}{2\ell^2}}$ where σ defines the height and ℓ the width of the kernel.

Gaussian Process: Sampling from Prior

Same procedure as for multivariate Gaussians:

- Generate $\boldsymbol{u} \in \mathbb{R}^D$ by drawing d samples from $\mathcal{N}(\mathbf{0}, \mathbf{I}_D)$.
- Perform Cholesky decomposition $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^\top$.
- Compute $\boldsymbol{y} = \boldsymbol{\mu} + \boldsymbol{L}\boldsymbol{u}$ where $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.



Gaussian Process: The Joint Distribution

We have training data $\boldsymbol{X} \in \mathbb{R}^{N \times D}$, corresponding observations $\boldsymbol{y} = f(\boldsymbol{X})$, and test data points $\boldsymbol{X}_* \in \mathbb{R}^{N_* \times D}$ for which we want to infer function values $\boldsymbol{y}_* = f(\boldsymbol{X}_*)$. The GP defines the following joint distribution

$$p(\boldsymbol{y}, \boldsymbol{y}_* | \boldsymbol{X}, \boldsymbol{X}_*) = \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{y}_* \end{pmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(\boldsymbol{X}) \\ m(\boldsymbol{X}_*) \end{bmatrix}, \begin{bmatrix} \boldsymbol{K} + \sigma_n^2 \boldsymbol{I} & \boldsymbol{K}_* \\ \boldsymbol{K}_*^\top & \boldsymbol{K}_{**} \end{bmatrix} \right)$$

where

$$\boldsymbol{K} = \kappa(\boldsymbol{X}, \boldsymbol{X}) \qquad \boldsymbol{K}_* = \kappa(\boldsymbol{X}, \boldsymbol{X}_*) \qquad \boldsymbol{K}_{**} = \kappa(\boldsymbol{X}_*, \boldsymbol{X}_*).$$

Typically, data points are corrupted by noise $\rightarrow$ our functions should not act as interpolators. We therefore assume

$$y_i = f(\boldsymbol{x}_i) + \epsilon \qquad \text{where} \qquad \epsilon \sim \mathcal{N}(0, \sigma_n^2).$$

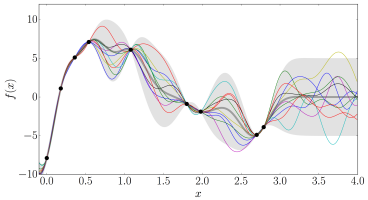
Gaussian Process: Inference

Inferring an unknown function value and its covariance follows from conditioning multivariate Gaussians:

$$p(\boldsymbol{y}_*|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{X}_*) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

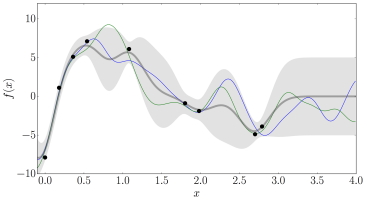
Non-noisy case

- $\boldsymbol{\mu} = m(\boldsymbol{X}_*) + \boldsymbol{K}_*^\top \boldsymbol{K}^{-1}(\boldsymbol{y} - m(\boldsymbol{X}))$
- $\boldsymbol{\Sigma} = \boldsymbol{K}_{**} - \boldsymbol{K}_*^\top \boldsymbol{K}^{-1} \boldsymbol{K}_*$



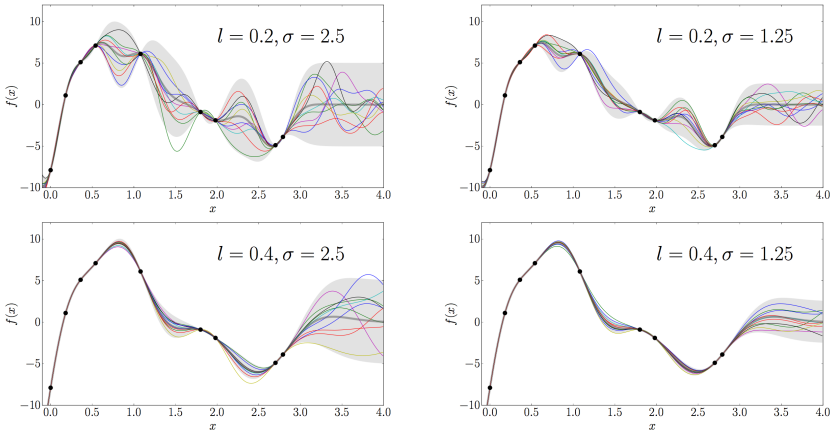
Noisy case

- $\boldsymbol{\mu} = m(\boldsymbol{X}_*) + \boldsymbol{K}_*^\top (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1}(\boldsymbol{y} - m(\boldsymbol{X}))$
- $\boldsymbol{\Sigma} = \boldsymbol{K}_{**} - \boldsymbol{K}_*^\top (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{K}_*$



Gaussian Process: Influence of Kernel Hyperparameters

$$\kappa(\mathbf{x}, \mathbf{x}') = \sigma^2 e^{-\frac{(\mathbf{x}-\mathbf{x}')^\top(\mathbf{x}-\mathbf{x}')}{2\ell^2}}$$



Conformal Prediction

- Can we construct confidence intervals for predictions? Yes!
- Want a function $\gamma(\cdot)$ that takes in a sample and maps it to a prediction set.
  - Classification: return a set of classes: $\gamma : \mathcal{X} \to \mathcal{C}$ with $\mathcal{C} \subseteq \mathcal{Y}$.
  - Regression: return a prediction range: $\gamma : \mathcal{X} \to [a, b]$ with $a, b \in \mathbb{R}, a \leq b$.
- We want to find $\gamma(\cdot)$ such that the prediction set contains the true label y with high probability (at significance level α):

$$p(y \in \gamma(\boldsymbol{x})) \geq 1 - \alpha$$

— https://arxiv.org/abs/2107.07511

Conformal Prediction: Adaptive Prediction Sets

- Desiderata for $\gamma(\cdot)$: small for easy samples, large for hard samples.
- Assume access to a validation set $D_{\text{val}} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N_{\text{val}}}$.

1) Get score of correct class

For each $(\boldsymbol{x}_i, y_i) \in D_{\text{val}}$ compute the consumed prob. mass until correct class.

$s_i = \sum_{j=1}^{K} f_{\boldsymbol{\theta}}(x_i)_j$

sorted softmax scores

2) Get $1-\alpha$ quantile

For $\{s_1, \ldots, s_{N_{\text{val}}}\}$, sort list and compute desired quantile $\hat{q}$ at index $\frac{\lceil (N_{\text{val}}+1)(1-\alpha) \rceil}{N_{\text{val}}}$.

3) Form prediction sets

For a new data point $\boldsymbol{x}$ get sorted softmax probabilities and add classes to prediction until $\hat{q}$ is reached.

$\gamma(\boldsymbol{x}) = \{c3, c7\}$ at α.

Selective Prediction

Selective prediction introduces a rejection class $\perp$ via gating mechanism.

Goal: Derive a selection function $g : \mathcal{X} \to \mathbb{R}$ which, given an acceptance threshold τ, determines whether a model $f : \mathcal{X} \to \mathcal{Y}$ should predict on a data point $\boldsymbol{x}$.

$$(f, g)(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}) & g(\boldsymbol{x}) \geq \tau \\ \perp & \text{otherwise.} \end{cases}$$

The performance of a selective classifier (f, g) on a dataset D is assessed based on

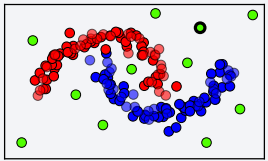
- the *coverage* of (f, g), i.e. what fraction of points we predict on; and
- the selective *utility* of (f, g) on the points it accepts.

$$\text{cov}(f, g) = \frac{|\{\boldsymbol{x} : g(\boldsymbol{x}) \geq \tau\}|}{|D|} \qquad \text{util}(f, g) = \sum_{\{(\boldsymbol{x}, y) : g(\boldsymbol{x}) \geq \tau\}} u(f(\boldsymbol{x}), y)$$

Anomaly / Out-of-Distribution Sample Detection

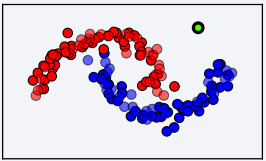
- Selective prediction identifies hard-to-classify examples within the distribution.
- But what about examples that are completely outside of the known distribution?

Supervised



Expose model to OOD points during training to regularize predictions.

Unsupervised

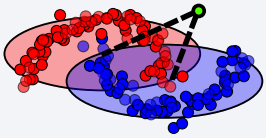


Determine OOD without knowledge of the nature of anomalous points.

Anomaly / Out-of-Distribution Sample Detection: Approaches

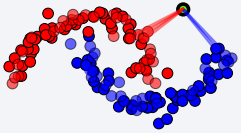
Mahalanobis Distance

Assume each class is a Gaussian in output representation and compute min distance.



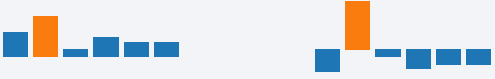
Nearest Neighbor Guiding

Check whether nearest neighbors in output representation are within a distance.



Max Softmax / Max Logit

Threshold either the maximum of the softmax or the pre-softmax / logit output.



Open Problems & Discussion

Scalability

- Bayesian models: computational infeasibility or approximations.
- Ensembles: need to train multiple models from scratch.

Distinguishing Types of Uncertainty

Correct error attribution is challenging in real world high-dimensional data; relevant for decision-making.

Model Misspecification

If the model assumption is violated, UQ methods can lure users into a false sense of security.

Evaluation & Regulation

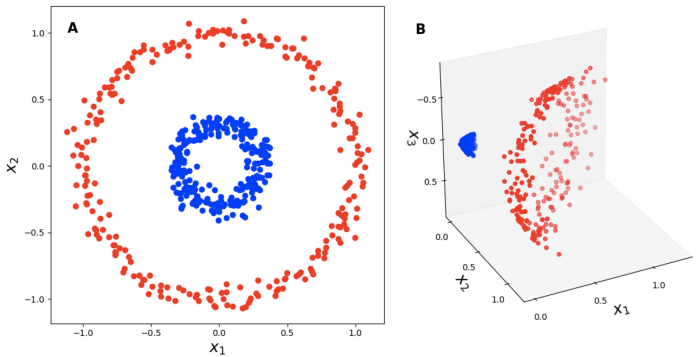
- Ambiguity of ground truth.
- Validation, certification, and ethical use of UQ methods for usage in highly critical applications.

Backup

Recap: Basis Functions

- How is this useful? We can use linear methods on non-linear features to yield non-linear decision boundaries and regression curves.

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{bmatrix}$$



— https://gregorygundersen.com/blog/2019/12/10/kernel-trick/

Kernels: Motivation

Generalized Linear Models (GLM)

- Fixed non-linear basis functions.
- Limited hypothesis space.
- Easy to optimize (convex).

Neural Network (NN)

- Adaptive non-linear basis functions.
- Rich hypothesis space.
- Hard to optimize (non-convex).

Towards Kernel Methods

- Feature space in GLM and NN needs to be explicitly constructed.
- Can we use a large (possibly infinite) set of fixed non-linear basis functions without explicitly constructing this space?
- Yes, by using kernel methods!

Kernel Methods

- Kernel methods are instance-based learners: they assign a weight θ_i to any training point $\mathbf{x}_i$.
- Predictions on new data points $\mathbf{x}'$ make use of a kernel function $\kappa(\cdot, \cdot)$ measuring the similarity of $\mathbf{x}'$ with all points $\mathbf{x}_i$ from the training set.
- Kernelized binary classification example:

$$\hat{y} = \operatorname{sgn} \sum_{i=1}^{n} \theta_i y_i \kappa(\mathbf{x}_i, \mathbf{x}')$$

where

- $y \in \{-1, +1\}$ is the label assigned to a data point $\mathbf{x}$.
- θ_i is the weight for training example $\mathbf{x}_i$.
- $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the kernel function measuring similarity between $\mathbf{x}, \mathbf{x}' \in \mathbb{R}$.

The Kernel Trick

- Let $\phi(\cdot)$ be a set of not further specified basis functions mappings.
- Explicitly constructing a high-dimensional feature space is expensive.
- By using the kernel trick, we can implicitly perform operations in a high-dimensional feature space.
- In many algorithms, this feature space only appears as a dot product $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle = \phi(\mathbf{x})^\top \phi(\mathbf{x}')$ of input pairs $\mathbf{x}$, $\mathbf{x}'$.
- We define these dot products as the kernel function

$$\kappa(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle = \phi(\mathbf{x})^\top \phi(\mathbf{x}')$$

  which can also be thought of as a similarity function between $\mathbf{x}$ and $\mathbf{x}'$.

Dual Representation

- Recall the regularized linear regression objective:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{n=1}^{N}(\boldsymbol{\theta}^\top \phi(\mathbf{x}_n) - y_n)^2 + \frac{\lambda}{2}\boldsymbol{\theta}^\top \boldsymbol{\theta}$$

- Finding optimal $\boldsymbol{\theta}$:

$$\nabla_\theta \mathcal{L}(\boldsymbol{\theta}) = \sum_{n=1}^{N}(\boldsymbol{\theta}^\top \phi(\mathbf{x}_n) - y_n)\phi(\mathbf{x}_n) + \lambda\boldsymbol{\theta} = 0$$

$$\boldsymbol{\theta} = -\frac{1}{\lambda}\sum_{n=1}^{N}\underbrace{(\boldsymbol{\theta}^\top \phi(\mathbf{x}_n) - y_n)}_{a_n}\phi(\mathbf{x}_n)$$

- The weights $\boldsymbol{\theta}$ can be written as a linear combination of the training examples:

$$\boldsymbol{\theta} = \sum_{n=1}^{N} a_n \phi(\mathbf{x}_n) \qquad \text{where } a = \left[a_1, \ldots, a_n\right] \text{ are called the dual parameters}$$

Dual Representation

- Substituting $\boldsymbol{\theta}$ back into linear regression $y(\mathbf{x}) = \boldsymbol{\theta}^\top \phi(\mathbf{x})$ yields:

$$\boldsymbol{\theta} = \sum_{n=1}^{N} a_n \phi(\mathbf{x}_n) \qquad y(\mathbf{x}) = \sum_{n=1}^{N} a_n \phi(\mathbf{x}_n)^\top \phi(\mathbf{x}) = \sum_{n=1}^{N} a_n \kappa(\mathbf{x}_n, \mathbf{x})$$

- The feature space only appears as a dot product.
- The kernel matrix, or gram matrix, $\mathbf{K} \in \mathbb{R}^{N \times N}$ collects kernel values in a symmetric positive semi-definite matrix for all data points (Mercer's theorem):

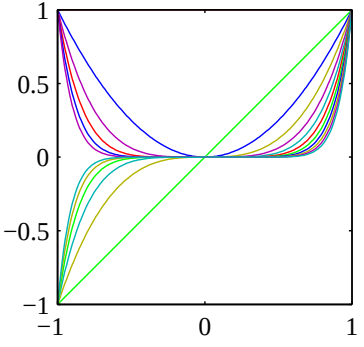
$$\mathbf{K}_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$$

- If a kernel defines such a kernel matrix, then the kernel is valid.

Popular Kernels

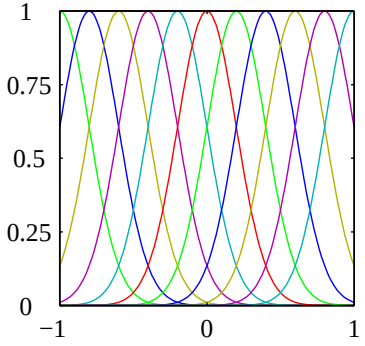
Polynomial Kernel

$$\kappa_{\mathrm{Pol}}(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + c)^d$$



Squared Exponential Kernel

$$\kappa_{\mathrm{SE}}(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{(\mathbf{x} - \mathbf{x}')^2}{2\ell^2}\right)$$



Kernel Composition Rules

Let $\kappa_1(\mathbf{x}, \mathbf{x}')$ and $\kappa_2(\mathbf{x}, \mathbf{x}')$ be valid kernels, then the following kernels are also valid:

- $\kappa(\mathbf{x}, \mathbf{x}') = c\kappa_1(\mathbf{x}, \mathbf{x}') \qquad \forall c > 0$
- $\kappa(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})\kappa_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}') \qquad \forall f$
- $\kappa(\mathbf{x}, \mathbf{x}') = g(\kappa_1(\mathbf{x}, \mathbf{x}')) \qquad g$ is polynomial with coefficients ≥ 0.
- $\kappa(\mathbf{x}, \mathbf{x}') = \exp(\kappa_1(\mathbf{x}, \mathbf{x}'))$
- $\kappa(\mathbf{x}, \mathbf{x}') = \kappa_1(\mathbf{x}, \mathbf{x}') + \kappa_2(\mathbf{x}, \mathbf{x}') \qquad$ kernel OR-ing
- $\kappa(\mathbf{x}, \mathbf{x}') = \kappa_1(\mathbf{x}, \mathbf{x}')\kappa_2(\mathbf{x}, \mathbf{x}') \qquad$ kernel AND-ing
- $\kappa(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{A}\mathbf{x}' \qquad \mathbf{A}$ symmetric and p.s.d.

Check out the Kernel Cookbook:
`https://www.cs.toronto.edu/~duvenaud/cookbook/`