Classification

Advanced Statistical Inference

Simone Rossi

Where are we?

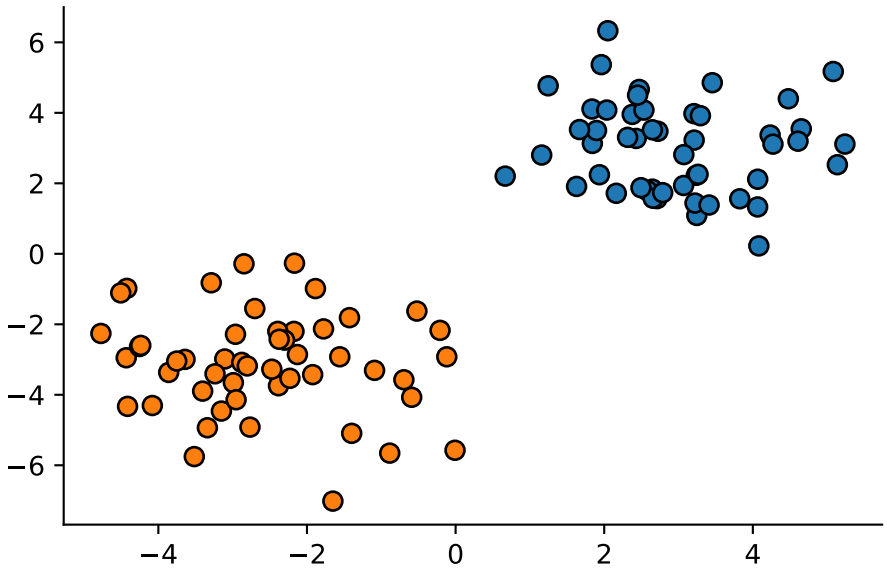
1. We have seen linear regression and how to do inference with it.
2. We have seen that exact inference is only possible for few cases.
3. We have seen that we can use approximate inference to do Bayesian inference in more complex models.
    - Laplace approximation, variational inference, MCMC methods.

Now we will implement what we have learned for a more complex model: **Classification**

Classification

A classification problem is a problem where we want to assign a label to an input $\boldsymbol{x} \in \mathbb{R}^D$.

- **Binary classification**: $y \in \{0, 1\}$.
- **Multiclass classification**: $y \in \{0, 1, \dots, K\}$.



Probabilistic vs Non-probabilistic classifiers

Given a training set $\{(x_n, y_n)\}_{n=1}^N$, we want to predict the label $y_\star$ of a new input $x_\star$.

- **Probabilistic classifiers**: output a probability distribution over the labels $P(y_\star = k \mid x_\star)$.
  - For binary classification, $P(y_\star = 1 \mid x_\star)$ and $P(y_\star = 0 \mid x_\star)$.
- **Non-probabilistic classifiers**: produce a hard assignment $y_\star = k$.

Examples: Probabilistic classifier

- Logistic regression
- Naive Bayes

Examples: Non-probabilistic classifier

- Support Vector Machines
- Decision Trees
- K-Nearest Neighbors

Probabilistic classifiers

Probabilistic classifiers are more informative than non-probabilistic classifiers.

- $P(y_\star = 1 \mid \boldsymbol{x}_\star) = 0.7$ is more informative than $y_\star = 1$.
- $P(y_\star = 1 \mid \boldsymbol{x}_\star)$ gives us a measure of confidence in the prediction.
- Particularly useful in applications where the cost of misclassification is high.

Logistic Regression

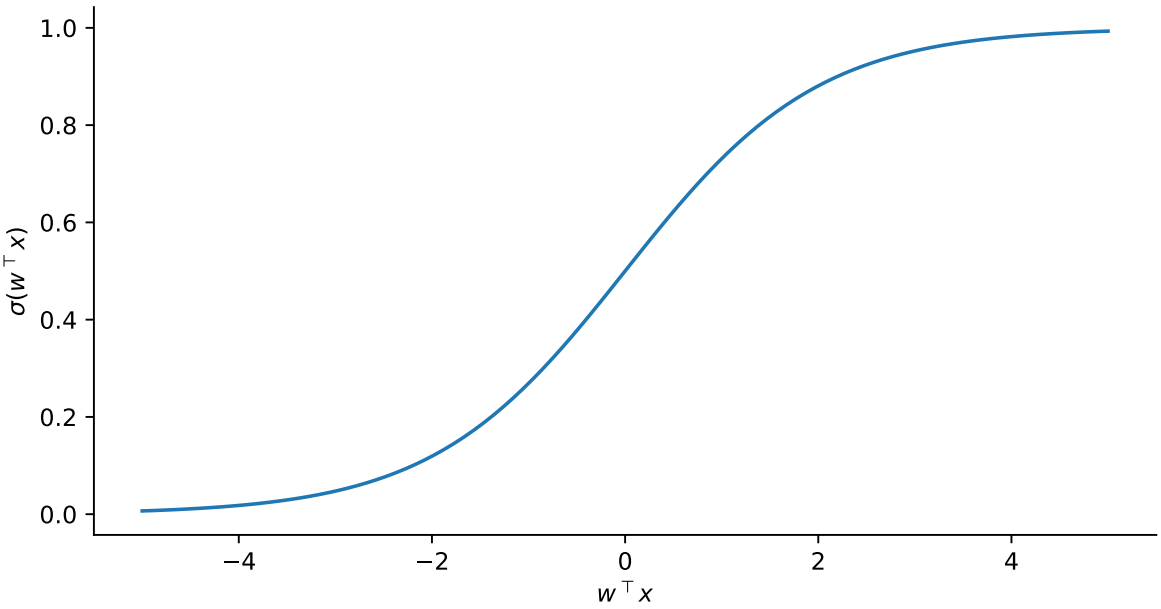
- Logistic regression is a probabilistic classifier that models the probability of the output y given the input $\boldsymbol{x}$.
- We model $P(y = 1 \mid \boldsymbol{x})$ through some function $f(\boldsymbol{w}, \boldsymbol{x})$.
- In linear regression, $f(\boldsymbol{w}, \boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$. Can we use this for classification?
  - No: the output of linear regression is unbounded and it can't be interpreted as a probability.
  - But, we can use a function $h(\cdot)$ to map the output of linear regression to the interval $[0, 1]$.

Logistic function

For logistic regression, we use the sigmoid function

$$P(y = 1 \mid \boldsymbol{x}) = \sigma(\boldsymbol{w}^\top \boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{w}^\top \boldsymbol{x})}$$

where $\sigma(\cdot)$ is the sigmoid function.



Bayesian Logistic Regression

- Same approach as for linear regression.
- We place a prior distribution over the parameters $\boldsymbol{w}$ and we define a likelihood to obtain the posterior distribution.

$$p(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{X})p(\boldsymbol{w})}{p(\boldsymbol{y} \mid \boldsymbol{X})}$$

- We can make predictions by integrating over the posterior distribution.

$$p(y_\star \mid \boldsymbol{x}_\star, \boldsymbol{y}, \boldsymbol{X}) = \int p(y_\star \mid \boldsymbol{w}, \boldsymbol{x}_\star)p(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X}) \, \mathrm{d}\boldsymbol{w}$$

Bayesian Logistic Regression: Likelihood

- First, we assume independence between the data points.

$$p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{X}) = \prod_{n=1}^{N} p(y_n \mid \boldsymbol{w}, \boldsymbol{x}_n)$$

We already know that:

- $P(y_n = 1 \mid \boldsymbol{w}, \boldsymbol{x}_n) = \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)$
- $P(y_n = 0 \mid \boldsymbol{w}, \boldsymbol{x}_n) = 1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)$.

What distribution does this correspond to?

Bayesian Logistic Regression: Likelihood

- The likelihood of a single data point is a Bernoulli distribution.

$$p(y_n \mid \boldsymbol{w}, \boldsymbol{x}_n) = \text{Bern}(y_n \mid \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n))$$

where

$$p(y_n \mid \boldsymbol{w}, \boldsymbol{x}_n) = \begin{cases} \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) & \text{if } y_n = 1 \\ 1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) & \text{if } y_n = 0 \end{cases}$$

Sometimes, we write this as

$$p(y_n \mid \boldsymbol{w}, \boldsymbol{x}_n) = \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)^{y_n} (1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n))^{1-y_n}$$

Bayesian Logistic Regression: Prior

- For logistic regression, we can use a Gaussian prior over the parameters.

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{0}, \sigma_w^2 \boldsymbol{I})$$

- Previously, we used a Gaussian prior because it makes the math easier.
- For logistic regression, no prior makes the math easier.
- We can use any prior that we want, but the Gaussian prior is a common choice.

Bayesian Logistic Regression: Posterior

- The posterior distribution is given by

$$p(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{X})p(\boldsymbol{w})}{p(\boldsymbol{y} \mid \boldsymbol{X})}$$

Now things get a bit more complicated.

- We can't compute the posterior distribution in closed form:
  - Prior is not conjugate to the likelihood. No prior is!
  - We don't know the form of $p(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X})$.
  - We can't compute the normalization constant $p(\boldsymbol{y} \mid \boldsymbol{X})$.

$$p(\boldsymbol{y} \mid \boldsymbol{X}) = \int p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{X})p(\boldsymbol{w}) \, \mathrm{d}\boldsymbol{w}$$

Bayesian Logistic Regression: Approximate Inference

Bayesian logistic regression is the first example where we will use approximate inference.

1. **Maximum a posteriori (MAP) estimation**: find the mode of the posterior distribution and use it as the point estimate of the parameters.
2. **Laplace approximation**: approximate the posterior distribution with a Gaussian distribution centered at the mode of the posterior.
3. **Variational inference**: approximate the posterior distribution with a simpler distribution that is easier to work with.
4. **MCMC methods**: sample from the posterior distribution using Markov Chain Monte Carlo methods.

Approximate Inference for Bayesian Logistic Regression

Maximum a Posteriori (MAP) Estimation

- We can find the mode of the posterior distribution by maximizing the (log) posterior distribution.

$$\boldsymbol{w}_{\text{MAP}} = \arg\max_{\boldsymbol{w}} \log p(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X}) = \arg\min \underbrace{-\log p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{X}) - \log p(\boldsymbol{w})}_{\mathcal{U}(\boldsymbol{w})}$$

- For linear regression, we could find $\boldsymbol{w}_{\text{MAP}}$ in closed form.
- For logistic regression, we can't do this in closed form.

We can use numerical optimization methods to find $\boldsymbol{w}_{\text{MAP}}$:

- Gradient ascent (or descent)
- Newton-Raphson's method

Gradient descent

- We can use gradient descent to find the mode of the posterior distribution.
- We need to compute the gradient $\boldsymbol{\nabla}_{\boldsymbol{w}} \mathcal{U}(\boldsymbol{w})$ and iterate until convergence.

$$\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} + \eta \boldsymbol{\nabla}_{\boldsymbol{w}} \mathcal{U}(\boldsymbol{w}^{(t)})$$

where η is the learning rate.

> **Note**
>
> In practice, we use **automatic differentiation** to compute the gradient.
>
> - JAX, PyTorch, TensorFlow, etc. have built-in functions to compute gradients.
> - In the labs, we will mostly use JAX, because it is fast and easy to use.

Laplace Approximation

Recall Laplace approximation:

- Approximate the posterior distribution with a Gaussian distribution $q(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$.
- Find the mode of the posterior distribution $\boldsymbol{w}_{\text{MAP}}$ (we have this already).
- Compute the Hessian of the log posterior at $\boldsymbol{w}_{\text{MAP}}$.

$$\boldsymbol{H} = -\boldsymbol{\nabla}^2 \log p(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X})\big|_{\boldsymbol{w}=\boldsymbol{w}_{\text{MAP}}}$$

- The covariance of the Gaussian approximation is given by the inverse of the Hessian.

Making predictions

- We can make predictions by integrating over the posterior distribution.

$$p(y_\star \mid \boldsymbol{x}_\star, \boldsymbol{y}, \boldsymbol{X}) = \int p(y_\star \mid \boldsymbol{w}, \boldsymbol{x}_\star) q(\boldsymbol{w}) \, \mathrm{d}\boldsymbol{w}$$

- Even though we have a Gaussian approximation of the posterior, this integral is intractable.

Two common approximations:

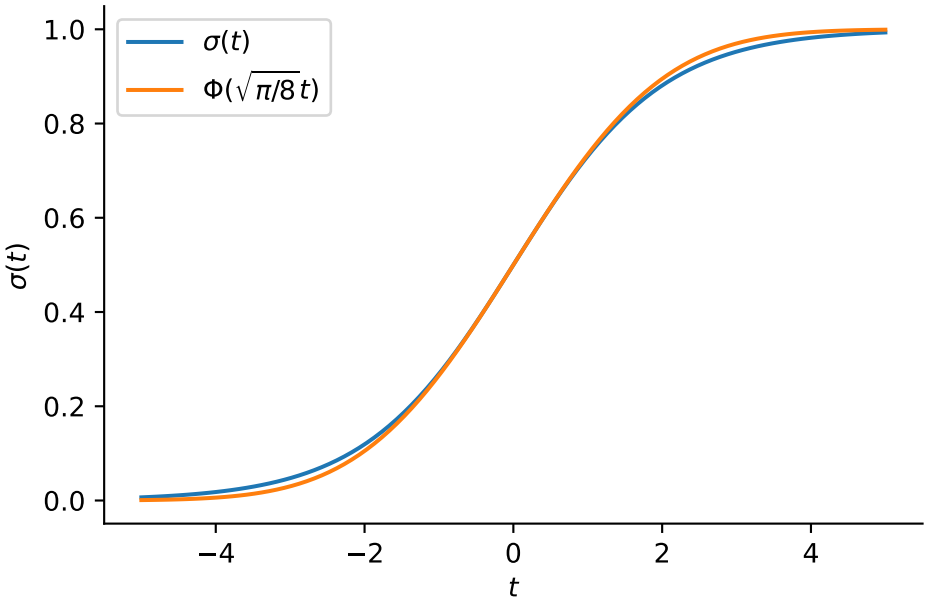
1. **Monte Carlo integration**: sample from the posterior distribution and average the predictions.

$$p(y_\star \mid \boldsymbol{x}_\star, \boldsymbol{y}, \boldsymbol{X}) \approx \frac{1}{S} \sum_{s=1}^{S} p(y_\star \mid \boldsymbol{w}^{(s)}, \boldsymbol{x}_\star) \quad \text{where } \boldsymbol{w}^{(s)} \sim q(\boldsymbol{w})$$

2. **Probit approximation**: use deterministic approximation of the sigmoid function.

Making predictions with probit approximation

Logistic function $\sigma(t)$ is approximated by the cumulative distribution function of a Gaussian $\Phi(\lambda t)$, where $\lambda = \sqrt{\pi/8}$.



Making predictions with probit approximation

- We can compute the distribution of $f_\star = f(\boldsymbol{w}, \boldsymbol{x}_\star)$ where $f(\boldsymbol{w}, \boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$ before applying the sigmoid function.
- The distribution of $f_\star$ is a Gaussian distribution.

$$q(f_\star \mid \boldsymbol{x}_\star, \boldsymbol{y}, \boldsymbol{X}) = \int p(f_\star \mid \boldsymbol{w}, \boldsymbol{x}_\star) \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \, \mathrm{d}\boldsymbol{w} = \mathcal{N}(f_\star \mid \boldsymbol{\mu}^\top \boldsymbol{x}_\star, \boldsymbol{x}_\star^\top \boldsymbol{\Sigma} \boldsymbol{x}_\star)$$

- Apply the probit approximation

$$p(y_\star \mid \boldsymbol{x}_\star, \boldsymbol{y}, \boldsymbol{X}) \approx \int \Phi(f_\star) q(f_\star \mid \boldsymbol{x}_\star, \boldsymbol{y}, \boldsymbol{X}) \, \mathrm{d}f_\star = \sigma\left(\frac{\boldsymbol{\mu}^\top \boldsymbol{x}_\star}{\sqrt{1 + \frac{\pi}{8} \boldsymbol{x}_\star^\top \boldsymbol{\Sigma} \boldsymbol{x}_\star}} \right)$$

- By marginalizing over the posterior distribution, we don't consider a single decision boundary.
- We consider all possible decision boundaries weighted by the uncertainty in the parameters.

Variational Inference

Recall variational inference:

1. Define a family of distributions $q(\boldsymbol{w}; \boldsymbol{\nu})$, where $\boldsymbol{\nu}$ are the variational parameters.
2. Minimize the KL divergence between the true posterior and the variational distribution

$$\boldsymbol{\nu}^* = \arg\min_{\boldsymbol{\nu}} \mathrm{KL}\left(q(\boldsymbol{w}; \boldsymbol{\nu}) \| p(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X})\right)$$

Remember: this is intracable!

3. Use the evidence lower bound (ELBO) to optimize the variational parameters.

$$\mathcal{L}_{\mathrm{ELBO}}(\boldsymbol{\nu}) = \mathbb{E}_{q(\boldsymbol{w}; \boldsymbol{\nu})}[\log p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{X})] - \mathrm{KL}\left(q(\boldsymbol{w}; \boldsymbol{\nu}) \| p(\boldsymbol{w})\right)$$

Variational Inference

Sampling from the posterior with MCMC

Recall MCMC methods:

- Even though we can't compute the posterior distribution, we can sample from it.
- Markov Chain Monte Carlo (MCMC) methods are a class of algorithms that allow us to sample from complex distributions.
- MCMC methods are based on the idea of proposing new samples and accepting them with a certain probability:
    1. Random walk: propose a new sample by adding Gaussian noise to the current sample.
    2. Hamiltonian Monte Carlo: propose a new sample by simulating the dynamics of a physical system.

Which method to use?

No free lunch theorem: no method is the best for all problems.

- **MAP estimation**: fast, but doesn't provide uncertainty estimates.
- **Laplace approximation**: needs to compute the MAP estimate and the Hessian inverse (can be very expensive for large models).
- **Variational inference**: flexible, scalable to large models and datasets, but rough approximation (unless you use more complex distributions).
- **MCMC**: most accurate, but computationally expensive
    - **Random walk Metropolis**: simple, but slow convergence.
    - **Hamiltonian Monte Carlo**: faster convergence, but requires gradient computation.

Uncertainty estimates

Uncertainty on class predictions

One advantage of Bayesian methods is that they provide uncertainty estimates.

- We have seen how to average all the decision boundaries for $P(y_\star = 1 \mid \boldsymbol{x}_\star, \boldsymbol{y}, \boldsymbol{X})$, by weighting them by the posterior distribution.
- But we can also compute the variance of the predictions, which gives us an idea of the uncertainty on the class probabilities, e.g. $P(y_\star = 1 \mid \boldsymbol{x}_\star, \boldsymbol{y}, \boldsymbol{X}) = 0.7 \pm 0.1$.

Uncertainty on class predictions

Thanks to Bayesian inference, we don't just get a point estimate of the class probabilities, but a full distribution over them.

Think about it:

- What does it mean if $P(y_\star = 1 \mid \boldsymbol{x}_\star, \boldsymbol{y}, \boldsymbol{X}) = 0.5$?
  - Option 1: the model is uncertain because it has seen conflicting data points (*aleatoric uncertainty*).
  - Option 2: the model is uncertain because it hasn't seen enough data points in that region of the input space (*epistemic uncertainty*).
- With Bayesian inference, we can distinguish between these two types of uncertainty by looking at the variance (or the entropy) of the predictions.

Performance evaluation

Evaluating the performance of a classifier

Accuracy: proportion of correctly classified examples.

Error rate (or **0/1 loss**): proportion of misclassified examples.

- Consider a set of predictions $\widehat{y}_i$ and true labels y_i for $i = 1, \ldots, N$.

$$\ell(\widehat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(\widehat{y}_i \neq y_i)$$

where $\mathbb{I}(\cdot)$ is the indicator function (1 if the condition is true, 0 otherwise).

$$\widehat{\boldsymbol{y}} = \arg\max_k p(y_i = k \mid \boldsymbol{x}_i)$$

Limitations of accuracy

Accuracy has some advantages:

- Easy to interpret.
- Can be used for binary and multiclass classification.
- Single number that we can use to compare different models.

But it has some limitations:

- Not suitable for imbalanced datasets.

Example

- We are building a classifier to detect fraudulent transactions; only 1% of the transactions are fraudulent.
- Class labels: $y = 0$ (not fraudulent); $y = 1$ (fraudulent).
- Build a classifier that always predicts $y = 0$.
- What is the accuracy of the classifier? Is it a good classifier?

Precision and recall

Need to define 4 quantities:

- **True positive (TP)**: number of positive examples correctly classified ($y = 1$, $\widehat{y} = 1$).
- **True negative (TN)**: number of negative examples correctly classified ($y = 0$, $\widehat{y} = 0$).
- **False positive (FP)**: number of negative examples incorrectly classified ($y = 0$, $\widehat{y} = 1$).
- **False negative (FN)**: number of positive examples incorrectly classified ($y = 1$, $\widehat{y} = 0$).

Precision and recall

- **Precision**: proportion of correctly classified positive examples among all examples classified as positive.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = 1 - \frac{\text{FP}}{\text{TP} + \text{FP}}$$

- **Recall** (or **sensitivity**): proportion of correctly classified positive examples among all positive examples.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = 1 - \frac{\text{FN}}{\text{TP} + \text{FN}}$$

Precision + Recall = F1 score

Single metric that combines precision and recall: **F1 score**.

$$\text{F1 score} = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

In our example of fraudulent transactions:

- Precision = 0 (no fraudulent transactions are detected).
- Recall = 1 (all non-fraudulent transactions are correctly classified).
- F1 score = 0.

Predictive test log-likelihood

- So far, we have evaluated the performance of the classifier based on the class predictions not the probabilities.
- **Predictive log-likelihood**: test the ability of the model to correctly predict the class probabilities.

Recall the predictive distribution $p(y_\star \mid \boldsymbol{x}_\star, \boldsymbol{y}, \boldsymbol{X})$ after marginalizing over the parameters with the posterior distribution (or any approximation).

$$p(y_\star \mid \boldsymbol{x}_\star, \boldsymbol{y}, \boldsymbol{X}) = \int p(y_\star \mid \boldsymbol{w}, \boldsymbol{x}_\star) p(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X}) \, \mathrm{d}\boldsymbol{w}$$

Predictive test log-likelihood

If we have samples from the posterior distribution:

$$\begin{aligned}\log p(y_\star \mid \boldsymbol{x}_\star, \boldsymbol{y}, \boldsymbol{X}) &\approx \log \frac{1}{S} \sum_{s=1}^{S} p(y_\star \mid \boldsymbol{w}^{(s)}, \boldsymbol{x}_\star) \\ &\approx \log \sum_{s=1}^{S} p(y_\star \mid \boldsymbol{w}^{(s)}, \boldsymbol{x}_\star) - \log S\end{aligned}$$

where $\boldsymbol{w}^{(s)} \sim p(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X})$.

Problem: we cannot swap log and sum, and we can have numerical issues when computing the the likelihood for each sample $p(y_\star \mid \boldsymbol{w}^{(s)}, \boldsymbol{x}_\star)$.

Log-sum-exp trick

- We can use the log-sum-exp trick to avoid numerical issues.

$$\begin{aligned}\log \sum_{s=1}^{S} p(y_\star \mid \boldsymbol{w}^{(s)}, \boldsymbol{x}_\star) &= \log \sum_{s=1}^{S} \exp\left(\log p(y_\star \mid \boldsymbol{w}^{(s)}, \boldsymbol{x}_\star)\right) \\ &= \text{logsumexp}\left(\log p(y_\star \mid \boldsymbol{w}^{(1)}, \boldsymbol{x}_\star), \ldots, \log p(y_\star \mid \boldsymbol{w}^{(S)}, \boldsymbol{x}_\star)\right)\end{aligned}$$

- Many libraries (Numpy, Pytorch, JAX, etc.) provide a numerical stable implementation of the log-sum-exp function
- The trick is based on the following identity:

$$\log(\exp(a) + \exp(b)) = a + \log(1 + \exp(b - a))$$

with $a \geq b$.

Calibration

Calibration: the predicted probabilities should match the true probabilities.

- When we predict a class probability of 0.9 for a class, we expect that 90% of the examples with this predicted probability belong to this class.

Tools to evaluate calibration:

- **Expected calibration error (ECE)**
- **Reliability diagram**

Accuracy and confidence

To compute calibration metrics, we need to group the examples based on the predicted probabilities.

Define:

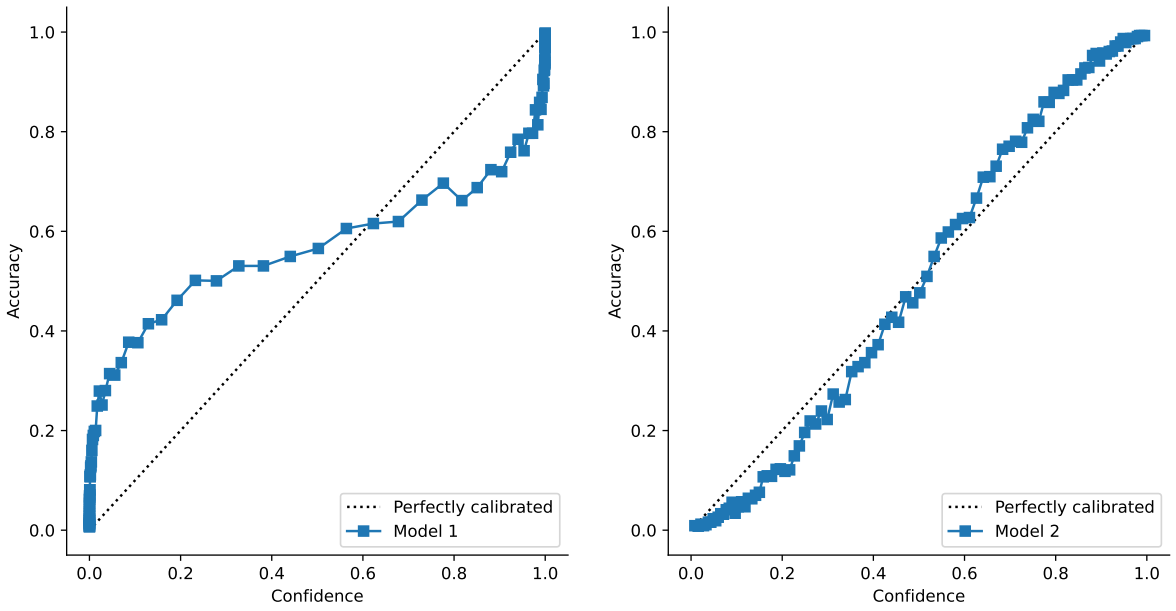
- $y_i = \arg\max_k p(y_i = k \mid x_i)$
- $\widehat{p}_i = \max_k p(y_i = k \mid x_i)$

Now:

1. Divide the range of predicted probabilities into B buckets or bins.
2. Let $\mathcal{B}_b$ be the set of examples with predicted probabilities in $\left(\frac{b-1}{B}, \frac{b}{B}\right]$.
3. For each bin $\mathcal{B}_b$, compute:
    - **Accuracy**: $\text{acc}(\mathcal{B}_b) = \frac{1}{|\mathcal{B}_b|} \sum_{i \in \mathcal{B}_b} \mathbb{I}(y_i = \widehat{y}_i)$
    - **Confidence**: $\text{conf}(\mathcal{B}_b) = \frac{1}{|\mathcal{B}_b|} \sum_{i \in \mathcal{B}_b} \widehat{p}_i$

Reliability diagram

Plot the accuracy as a function of the confidence.



Expected calibration error (ECE)

Expected calibration error (ECE) measures "how much" the predicted probabilities deviate from the true probabilities.

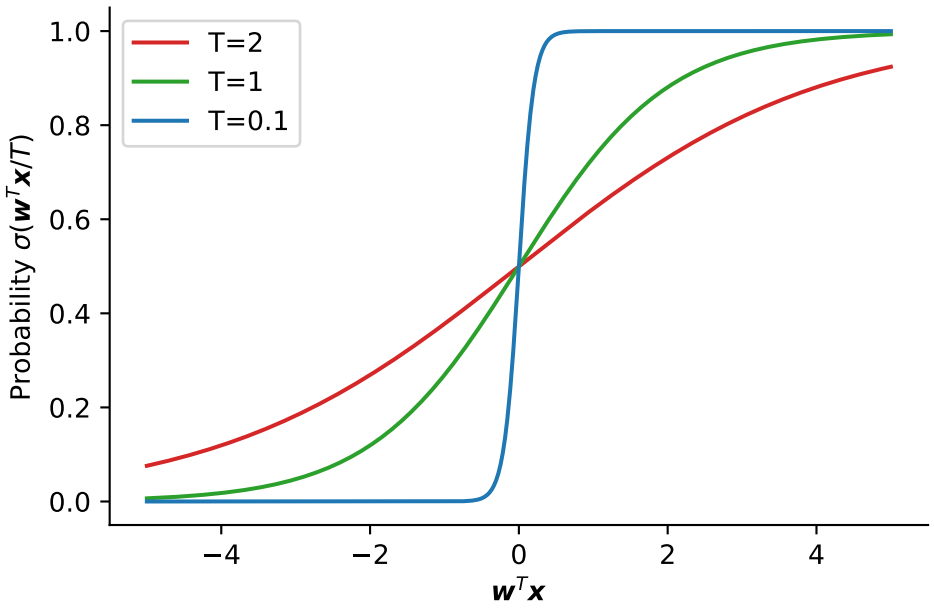
$$\text{ECE} = \sum_{b=1}^{B} \frac{|\mathcal{B}_b|}{B} |\text{acc}(\mathcal{B}_b) - \text{conf}(\mathcal{B}_b)|$$

- **ECE = 0**: perfect calibration.
- **ECE = 1**: worst calibration (maximum error).

Improving calibration

- **Temperature scaling**: scale the logits before applying the sigmoid function.

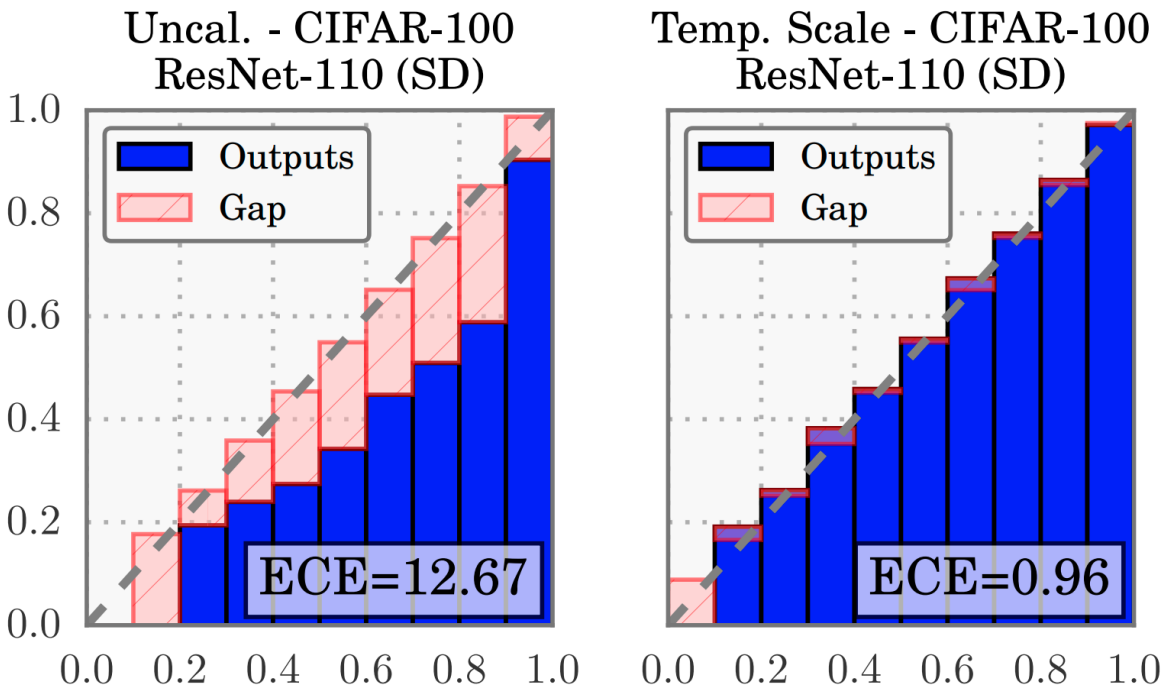
$$p(y_i \mid \boldsymbol{w}, \boldsymbol{x}_i) = \text{Bern}(\sigma(\boldsymbol{w}^T \boldsymbol{x}_i / T))$$



Improving calibration: a recipe for temperature scaling

1. **Train the model**.
2. **Compute the ECE** on the validation set.
3. **Choose the temperature** that minimizes the ECE on the validation set.

4. **Re-calibrate the model** using the chosen temperature.



Note: temperature scaling is not affecting model performance (accuracy, F1 score, etc.), but only the calibration.