Probabilistic Supervised Learning

Debaditya Roy

Probabilistic Supervised Learning

Goal: Learn the **conditional probability** of output y given input $\boldsymbol{x}$

$$p(y|\boldsymbol{x})$$

Choose distribution based on the type of output:

- Real → Gaussian
- Binary → Bernoulli
- Multiclass → Multinoulli
- Count → Poisson
- Positive real → Gamma

Discriminative vs. Generative Learning

Discriminative Models

- Learn **directly**: $p(y|\boldsymbol{x})$
- Focus on **prediction**
- Don't model input $\boldsymbol{x}$

Real output (Gaussian):

$$p(y|\boldsymbol{x}) = N(\boldsymbol{w}^\top \boldsymbol{x}, \beta^{-1})$$

Binary output (Bernoulli):

$$p(y|\boldsymbol{x}) = \text{Bernoulli}(\sigma(\boldsymbol{w}^\top \boldsymbol{x}))$$

Generative Models

- Learn joint: $p(y|\boldsymbol{x}) = \frac{p(\boldsymbol{x},y)}{p(\boldsymbol{x})}$
- Can be used for prediction and data generation
- Require modeling both $\boldsymbol{x}$ and y

Multiclass output (Multinoulli):

$$p(\boldsymbol{x}|y = k) = \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$
$$p(y) = \text{Categorical}$$

Count output(Poisson):

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = Poisson(\exp(\boldsymbol{w}^\top \boldsymbol{x}))$$

Note: Generative approach can also be used for other settings too, such as unsupervised learning and semi-supervised learning

Probabilistic Linear Regression

Goal: We **model uncertainty** – instead of just finding one best line, we define a *probabilistic model* over the data.

Given:

- Training data $\{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$, where $\mathbf{x}_n \in \mathbb{R}^D$, $y_n \in \mathbb{R}$
- Assume:

$$y_n = \mathbf{w}^\top \mathbf{x}_n + \epsilon_n, \quad \epsilon_n \sim \mathcal{N}(0, \beta^{-1})$$

Likelihood:

$$p(y_n \mid \mathbf{x}_n, \mathbf{w}, \beta) = \mathcal{N}(y_n \mid \mathbf{w}^\top \mathbf{x}_n, \beta^{-1})$$

Prior on Weights

Prior Assumption:

$$p(\boldsymbol{w}) = \mathcal{N}(0, \lambda^{-1}\boldsymbol{I})$$

Interpretation:

- Zero-mean Gaussian prior over weights
- Equivalent to L2 regularization
- λ controls strength of regularization

Remarks:

- Non-zero mean or structured covariance priors can be used if prior knowledge is available

Posterior over Weights

Bayes' Rule:

$$p(\boldsymbol{w} \mid \mathbf{X}, \boldsymbol{y}) \propto p(\boldsymbol{y} \mid \mathbf{X}, \boldsymbol{w}) \cdot p(\boldsymbol{w})$$

Closed-form posterior (Gaussian):

$$p(\boldsymbol{w} \mid \mathbf{X}, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N)$$

$$\boldsymbol{\Sigma}_N = \left(\lambda \boldsymbol{I} + \beta \mathbf{X}^\top \mathbf{X}\right)^{-1}$$

$$\boldsymbol{\mu}_N = \beta \boldsymbol{\Sigma}_N \mathbf{X}^\top \boldsymbol{y}$$

Posterior is Gaussian due to conjugacy.

Derivation of posterior

$$p(y_n \mid \mathbf{x}_n, \mathbf{w}, \beta) = \mathcal{N}(y_n \mid \mathbf{w}^\top \mathbf{x}_n, \beta^{-1})$$

$$p(\mathbf{w}) = \mathcal{N}(0, \lambda^{-1} \mathbf{I})$$

$$\log p(\mathbf{w} \mid \mathbf{X}, \mathbf{y}) \propto \log p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) + \log p(\mathbf{w})$$

$$\log p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) = -\frac{\beta}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) \qquad \log p(\mathbf{w}) = -\frac{\lambda}{2}\mathbf{w}^\top \mathbf{w}$$

$$(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{y}^\top \mathbf{y} - 2\mathbf{y}^\top \mathbf{X}\mathbf{w} + \mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w}$$

$$\log p(\mathbf{w} \mid \mathbf{X}, \mathbf{y}) \propto -\frac{1}{2}\left[\mathbf{w}^\top(\beta\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})\mathbf{w} - 2\beta\mathbf{y}^\top\mathbf{X}\mathbf{w}\right]$$

$$\mathbf{w}^\top A\mathbf{w} - 2\mathbf{b}^\top\mathbf{w} = (\mathbf{w} - \mathbf{w}_0)^\top A(\mathbf{w} - \mathbf{w}_0) - \mathbf{w}_0^\top A\mathbf{w}_0$$

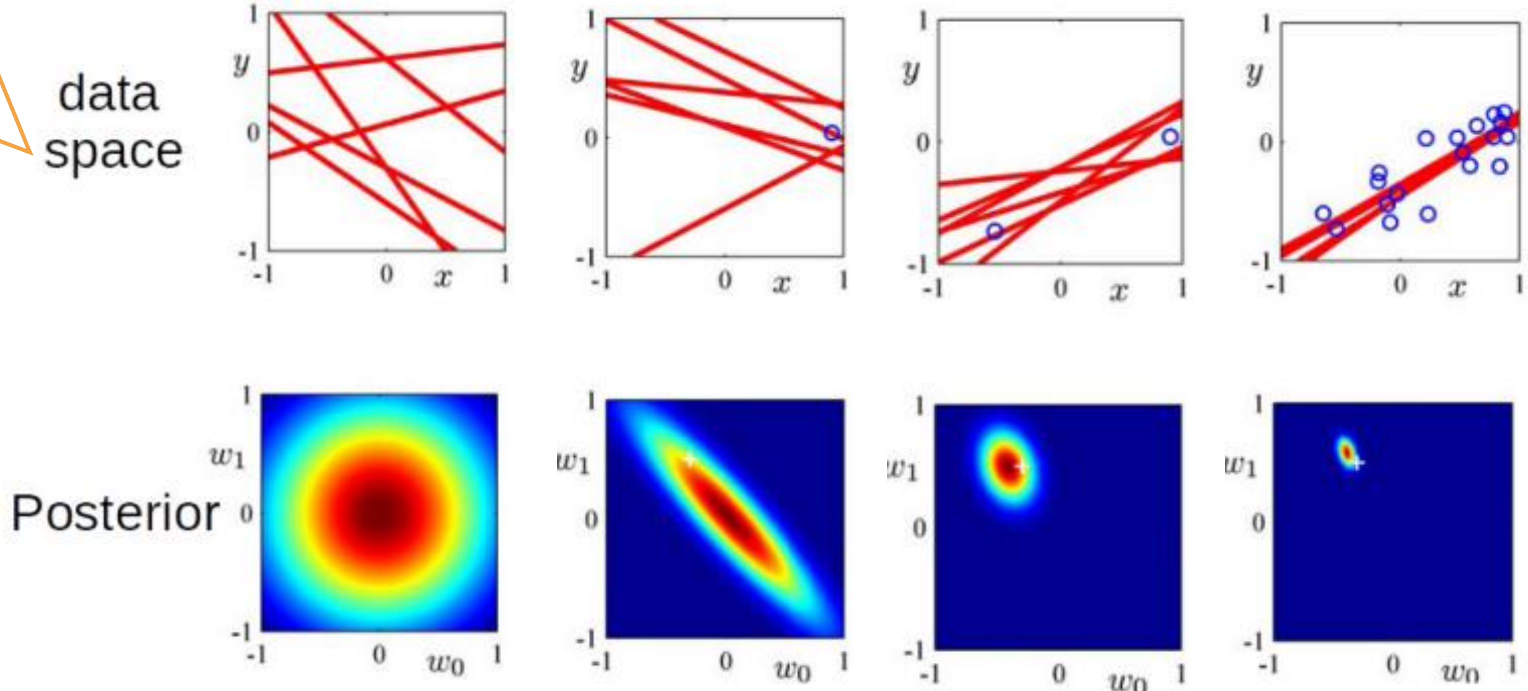
$$\mathbf{w}_0 = A^{-1}\mathbf{b} = (\beta\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}(\beta\mathbf{X}^\top\mathbf{y})$$

$$\mu_N = A^{-1}\mathbf{b} = \beta\Sigma_N\mathbf{X}^\top\mathbf{y} \qquad \mathbf{\Sigma}_N = (\lambda\mathbf{I} + \beta\mathbf{X}^\top\mathbf{X})^{-1}$$

Posterior: A Visualization

- Assume a lin. reg. problem with true $\boldsymbol{w} = [w_0, w_1], w_0 = -0.3, w_1 = 0.5$
- Assume data generated by a linear regression model $y = w_0 + w_1 x +$ "noise"
  - Note: It's actually 1-D regression (w_0 is just a bias term), or 2-D reg. with feature $[1, x]$
- Figures below show the "data space" and posterior of $\boldsymbol{w}$ for different number of observations (note: with no observations, the posterior = prior)

Each red line represents the "data" generated for a randomly drawn $\boldsymbol{w}$ from the current posterior



Posterior Predictive Distribution

For a new input x^*:

$$p(y^* \mid \mathbf{x}^*, \mathcal{D}) = \int p(y^* \mid \mathbf{x}^*, \mathbf{w})\, p(\mathbf{w} \mid \mathcal{D})\, d\mathbf{w}$$

Result: Gaussian distribution

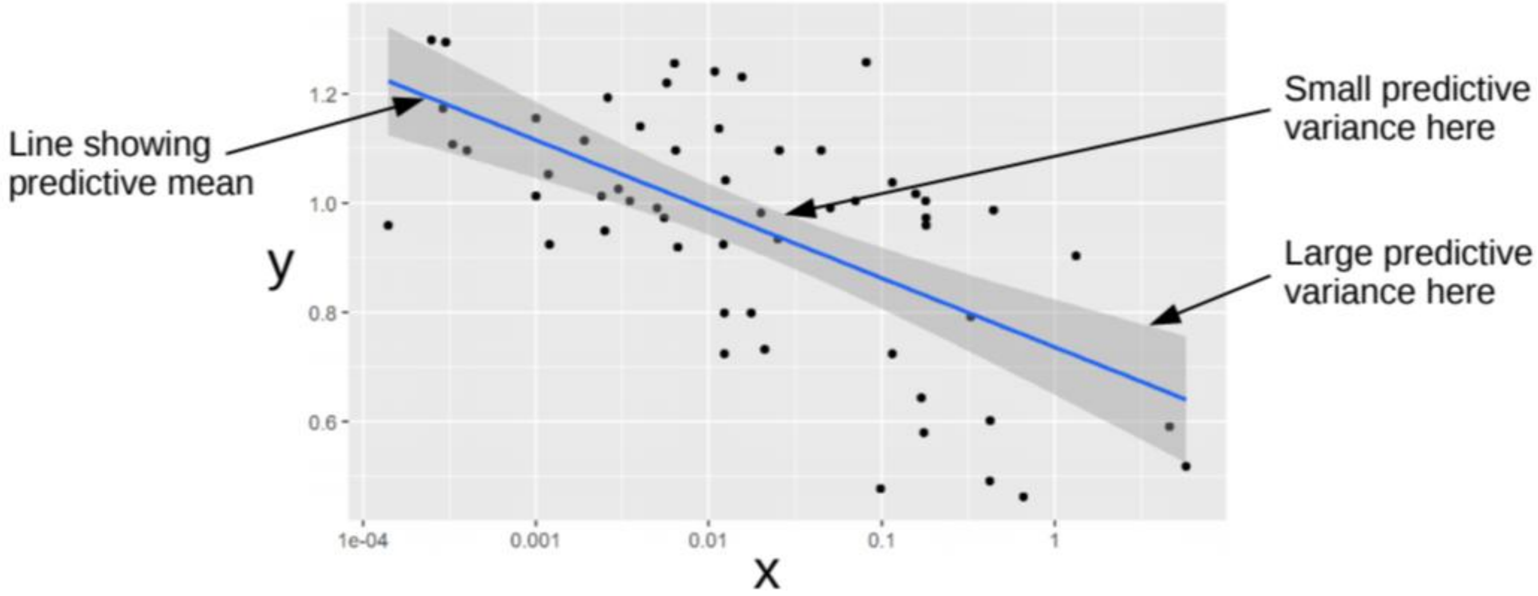
$$p(y^* \mid \mathbf{x}^*, \mathcal{D}) = \mathcal{N}(\mu_N^\top \mathbf{x}^*,\ \beta^{-1} + \mathbf{x}^{*\top} \Sigma_N \mathbf{x}^*)$$

Notes:

- Mean: $\mu_N^\top \mathbf{x}^*$
- Variance includes noise + uncertainty in $\mathbf{w}$
- More informative than MLE/MAP point estimates

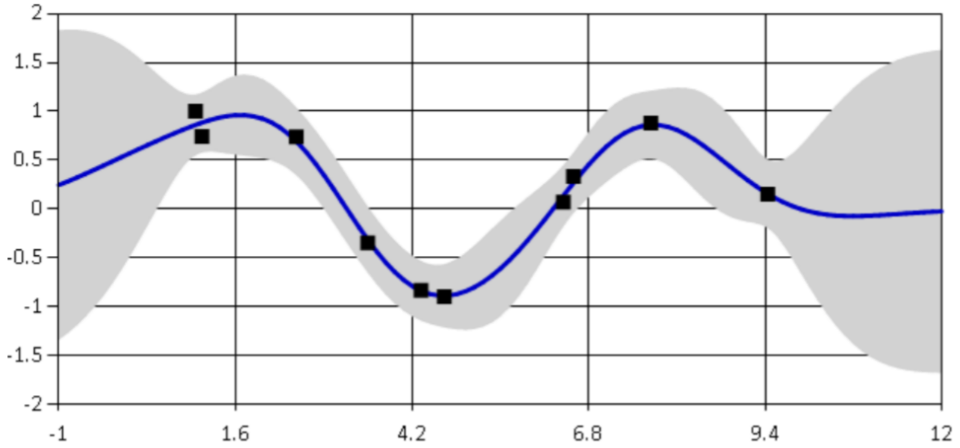
Posterior Predictive Distribution: An Illustration

- Black dots are training examples



- Width of the shaded region at any x denotes the predictive uncertainty at that x (+/- one std-dev)
- Regions with more training examples have smaller predictive variance

Nonlinear Regression



- Can extend the linear regression model to handle nonlinear regression problems

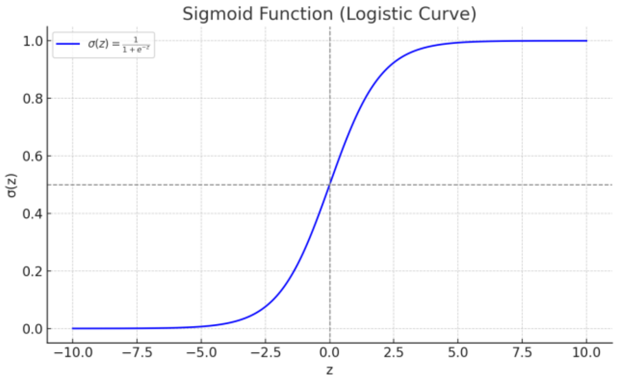
- One way is to replace the feature vectors $\boldsymbol{x}$ by a nonlinear mapping $\phi(\boldsymbol{x})$

$$p(y|\mathbf{x},\mathbf{w}) = \mathcal{N}(\mathbf{w}^\top \phi(\mathbf{x}), \beta^{-1})$$

Can be pre-defined (e.g., replace a scalar x by polynomial mapping $[1, x, x^2]$) or extracted by a pretrained deep neural net

- Alternatively, a kernel function can be used to implicitly define the nonlinear mapping
- More on nonlinear regression when we discuss Gaussian Processes

Probabilistic Binary Logistic Regression



For $y \in \{0, 1\}$, model:

$$p(y = 1 \mid \boldsymbol{x}, \boldsymbol{w}) = \sigma(\boldsymbol{w}^\top \boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{w}^\top \boldsymbol{x}}}$$

$$p(y \mid \mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^\top \mathbf{x})^y \cdot (1 - \sigma(\mathbf{w}^\top \mathbf{x}))^{1-y}$$

Decision rule: Predict 1 if probability > 0.5

Likelihood and MAP

Likelihood over all data:

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

Negative log-likelihood (Binary Cross-Entropy):

$$\mathcal{L}(\boldsymbol{w}) = -\sum_{n=1}^{N} [y_n \log \mu_n + (1 - y_n) \log(1 - \mu_n)]$$

With prior:

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I}) \propto \exp\left(-\frac{\lambda}{2}\|\mathbf{w}\|^2\right)$$

Encourages smaller weights (regularization)

$$\boldsymbol{w}_{\mathrm{MAP}} = \arg\min_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$$

Posterior Distribution

Posterior is intractable:

$$p(\boldsymbol{w} \mid \mathcal{D}) \propto p(\boldsymbol{w}) \cdot \prod_n \sigma(\boldsymbol{w}^\top \mathbf{x}_n)^{y_n}(1 - \sigma(\boldsymbol{w}^\top \mathbf{x}_n))^{1-y_n}$$

- Sigmoid likelihood is not conjugate to the Gaussian prior.
- Posterior does not have a closed-form solution — you can't simplify the product analytically.

Use Laplace Approximation:

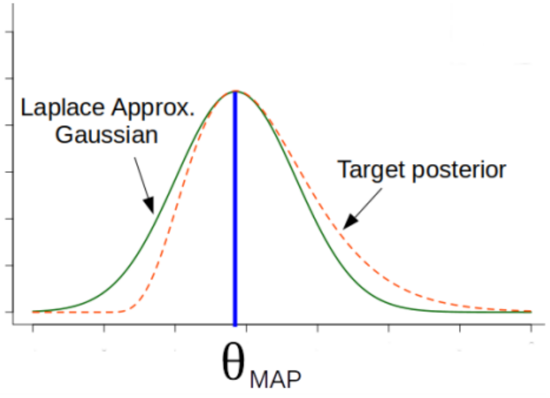
$$p(\boldsymbol{w} \mid \mathcal{D}) \approx \mathcal{N}(\boldsymbol{w}_{\mathrm{MAP}}, \Lambda^{-1})$$

Laplace's (or Gaussian) Approximation

- Consider a posterior distribution that is intractable to compute

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D},\theta)}{p(\mathcal{D})} = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}$$

- Laplace approximation approximates the above using a **Gaussian** distribution



$$p(\theta|\mathcal{D}) \approx \mathcal{N}(\theta|\theta_{MAP}, \boldsymbol{\Lambda}^{-1})$$

$$\theta_{MAP} = \operatorname{argmax}_\theta \log p(\theta|\mathcal{D})$$

$$\boldsymbol{\Lambda} = -\nabla_\theta^2 \log p(\theta|\mathcal{D})\Big|_{\theta=\theta_{MAP}} = -\nabla_\theta^2 \log p(\mathcal{D},\theta)\Big|_{\theta=\theta_{MAP}}$$

Tells us about the space (curvature) of the true posterior around θ_{MAP}

Negative of the Hessian, i.e., the second derivative of the log joint, at θ_{MAP}

- Laplace's approx. is based on a second-order Taylor approx. of the posterior

Derivation of the Laplace's Approximation

$$\boxed{p(\mathcal{D}) \approx \exp\big(\log p(\mathcal{D}, \theta_{MAP})\big) \times (2\pi)^{D/2} \det(\mathbf{\Lambda})^{1/2}}$$

We also get a Laplace approximation of the marginal likelihood (for free!)

Note: Sometimes marginal likelihood is also called model evidence

- Let's write the Bayes rule as

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}, \theta)}{p(\mathcal{D})} = \frac{p(\mathcal{D}, \theta)}{\int p(\mathcal{D}, \theta) d\theta} = \frac{\exp[\log p(\mathcal{D}, \theta)]}{\int \exp[\log p(\mathcal{D}, \theta)] d\theta}$$

- Consider second-order Taylor approximation of a function $f(\theta)$ around some θ_0

$$f(\theta) \approx f(\theta_0) + (\theta - \theta_0)^\top \nabla_\theta f(\theta_0) + \frac{1}{2}(\theta - \theta_0)^\top \nabla_\theta^2 f(\theta_0)(\theta - \theta_0)$$

- Assuming $f(\theta) = \log p(\mathcal{D}, \theta)$ and $\theta_0 = \theta_{MAP}$

Constant w.r.t. θ

Same as $\nabla^2 \log p(\theta_{MAP}|\mathcal{D})$

$$\log p(\mathcal{D}, \theta) \approx \log p(\mathcal{D}, \theta_{MAP}) + \frac{1}{2}(\theta - \theta_{MAP})^\top \nabla_\theta^2 \log p(\mathcal{D}, \theta_{MAP})(\theta - \theta_{MAP})$$

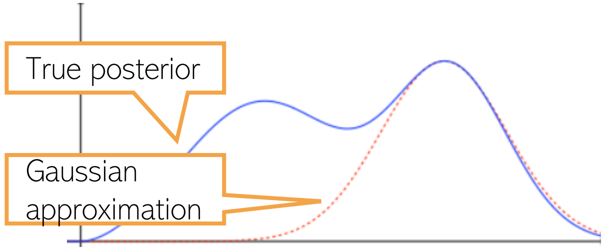
$$\begin{aligned} p(\theta|\mathcal{D}) &\propto \exp\left[-\frac{1}{2}(\theta - \theta_{MAP})^\top(-\nabla_\theta^2 \log p(\mathcal{D}, \theta_{MAP}))(\theta - \theta_{MAP})\right] \\ &= \mathcal{N}(\theta|\theta_{MAP}, \mathbf{\Lambda}^{-1}) \quad \text{(where } \mathbf{\Lambda} = -\nabla_\theta^2 \log p(\mathcal{D}, \theta_{MAP}) = -\mathbf{H}\text{)} \end{aligned}$$

Properties of Laplace's Approximation

- Straightforward if posterior's derivatives (first/second) can be computed easily
- Expensive if parameter θ is very high dimensional

  E.g., a deep neural network, or even in simpler models (e.g., logistic reg with a very large number of features

  - Reason: We need to compute and invert Hessian of size $D \times D$ (D is the # of params)
- Can do badly if the (true) posterior is multimodal



For multimodal posteriors, can use a mixture of Laplace approximations*

Useful for deep learning models

If K local modes, then define the approx. posterior as a mixture of K Gaussians

$$p(\theta|D) \approx \sum_{k=1}^{K} \pi^{(k)} \mathcal{N}(\theta|\theta_{MAP}^{(k)}, H^{(k)^{-1}})$$

(see paper cited below for details)

- Used only when θ is a real-valued vector (because of Gaussian approximation)
- Note: Even if we have a <u>non-probabilistic</u> model (loss function + regularization), we can obtain an approx. "posterior" for that model using the Laplace's approximation
  - Optima of the regularized loss function will be Gaussian's mean
  - Inverse of the second derivative of the regularized loss function will be covariance matrix

*Mixtures of Laplace Approximations for Improved Post-Hoc Uncertainty in Deep Learning (Eschenhagen et al, 2021)

Posterior Predictive Distribution

- The posterior predictive distribution can be computed as

$$p(y_* = 1|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \int p(y_* = 1|\boldsymbol{w}, \boldsymbol{x}_*) p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}) d\boldsymbol{w}$$

Integral not tractable and must be approximated

sigmoid

Gaussian (if using Laplace approx.)

- Monte-Carlo approximation of this integral is one possible way
  - Draw M samples $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_M$, from the approx. of posterior
  - Approximate the PPD as follows

$$p(y_* = 1|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) \approx \frac{1}{M}\sum_{m=1}^{M} p(y_* = 1|\boldsymbol{w}_m, \boldsymbol{x}_*) = \frac{1}{M}\sum_{m=1}^{M} \sigma(\boldsymbol{w}_m^\top \boldsymbol{x}_n)$$

- In contrast, when using MLE/MAP solution $\hat{\boldsymbol{w}}_{opt}$, the plug-in pred. distribution

$$\begin{aligned} p(y_* = 1|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) &= \int p(y_* = 1|\boldsymbol{w}, \boldsymbol{x}_*) p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}) d\boldsymbol{w} \\ &\approx p\left(y_* = 1\middle|\hat{\boldsymbol{w}}_{opt}, \boldsymbol{x}_*\right) = \sigma(\hat{\boldsymbol{w}}_{opt}{}^\top \boldsymbol{x}_n) \end{aligned}$$

Multiclass Logistic Regression

Now, let's **extend this to multiple classes** $y \in \{1, 2, \ldots, K\}$. We model the **probability of each class** using the **softmax function**:

$$P(y = k \mid \mathbf{x}, \mathbf{W}) = \frac{\exp(\mathbf{w}_k^\top \mathbf{x})}{\sum_{j=1}^{K} \exp(\mathbf{w}_j^\top \mathbf{x})}$$

Where:

- $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_K]$ are the weight vectors for each class.
- The model outputs a **categorical distribution** over the classes.

MLR: Prior, Likelihood, and Posterior

1. Prior over weights:

$$p(\mathbf{W}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{w}_k \mid \mathbf{0}, \tau^{-1}\mathbf{I})$$

2. Likelihood:

Given data $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}$, the likelihood is:

$$p(\mathcal{D} \mid \mathbf{W}) = \prod_{n=1}^{N} \prod_{k=1}^{K} P(y_n = k \mid \mathbf{x}_n, \mathbf{W})^{\mathbb{I}(y_n = k)}$$

3. Posterior:

$$p(\mathbf{W} \mid \mathcal{D}) \propto p(\mathbf{W}) \cdot p(\mathcal{D} \mid \mathbf{W})$$

As with binary logistic regression, the posterior is **intractable** due to the softmax in the likelihood. So we approximate it.

MLR: Posterior Distribution

We can approximate $p(\mathbf{W} \mid \mathcal{D})$ using:

- **Laplace Approximation:**
  - Approximate the posterior as Gaussian around the MAP estimate:

$$p(\mathbf{W} \mid \mathcal{D}) \approx \mathcal{N}(\mathbf{W}_{\text{MAP}}, \Lambda^{-1})$$

  - Λ is the **Hessian** of the negative log posterior at the MAP.
- **MCMC Methods:**
  - Sample from the posterior using methods like **Hamiltonian Monte Carlo**, **Gibbs sampling**, or **Metropolis-Hastings**.

MLR: Posterior Predictive Distribution

To make predictions for a new point $\mathbf{x}_*$, we **marginalize over the posterior:**

$$p(y_* \mid \mathbf{x}_*, \mathcal{D}) = \int p(y_* \mid \mathbf{x}_*, \mathbf{W}) p(\mathbf{W} \mid \mathcal{D}) \, d\mathbf{W}$$

Since this integral is intractable, we use:

- **Monte Carlo approximation:**
  - Draw samples $\mathbf{W}^{(s)} \sim p(\mathbf{W} \mid \mathcal{D})$
  - Compute:

$$p(y_* \mid \mathbf{x}_*, \mathcal{D}) \approx \frac{1}{S} \sum_{s=1}^{S} p(y_* \mid \mathbf{x}_*, \mathbf{W}^{(s)})$$

Generative Supervised Learning

- The conditional distribution $p(y|x)$ can also be defined as

$$p(y|x) = \frac{p(x,y)}{p(x)}$$

- Generative sup. learning is usually more work because $p(x,y)$ has to be estimated
- However, there are some benefits as well. For example, for classification

$p(y)$ is called the "class-prior" or "class-marginal" distribution

Can incorporate knowledge of frequency ("size") of each class in training data

Can incorporate knowledge of the distribution ("shape") of each class in training data

$$p(y|x) = \frac{p(x,y)}{p(x)} = \frac{p(y)p(x|y)}{p(x)}$$

Can assume simple/sophisticated types of distributions for the "class-conditional" distribution $p(x|y)$ and learned them using the training data of each class

Estimating Class Marginals

If only two classes, assume Bernoulli

- Estimating class marginals $p(y=k)$ is usually straightforward
- Since labels are discrete, we assume class marginal $p(y)$ to be a multinoulli

$\pi_k = p(y=k)$

These probabilities sum to 1: $\sum_{k=1}^{K} \pi_k = 1$

$$p(y|\boldsymbol{\pi}) = \text{multinoulli}(y|\pi_1, \pi_2, \ldots, \pi_K) = \prod_{k=1}^{K} \pi_k^{\mathbb{I}[y=k]}$$

- Given N i.i.d. labelled examples $\{(x_n, y_n)\}_{n=1}^{N}$, $y_n \in \{1,2,\ldots,K\}$ the MLE soln.

$$\boldsymbol{\pi}_{MLE} = \underset{\boldsymbol{\pi}}{\operatorname{argmax}} \sum_{n=1}^{N} \log p(y_n|\boldsymbol{\pi})$$

Subject to constraint $\sum_{k=1}^{K} \pi_k = 1$

- MLE solution is $p(y=k) = \pi_k = N_k/N$ where $N_k = \sum_{n=1}^{N} \mathbb{I}[y=k]$
  - Thus $p(y=k) = \pi_k$ is simply the fraction of inputs from class k
- Can also compute MAP estimate or full posterior of $\boldsymbol{\pi}$ using a Dirichlet prior

Estimating Class-Conditionals

To be estimated using the N_k training inputs $\{\boldsymbol{x}_n : y_n = k\}$ from class k

- Can assume a distribution $p(\boldsymbol{x}|y=k) = p(\boldsymbol{x}|\theta_k)$ for inputs of each class k
- If $\boldsymbol{x}$ is D-dimensional, $p(\boldsymbol{x}|\theta_k)$ will be a D-dimensional distribution
- Can compute MLE/MAP estimate or full posterior of θ_k
  - This essentially is a density estimation problem for the class-cond.
  - In principle, can use any density estimation method

E.g., if $p(\boldsymbol{x}|\theta_k)$ is multivariate Gaussian then assume it to have a diagonal covariance matrix instead of full covariance matrix

Such assumptions greatly reduce the number of parameters to be estimated

- Choice of the form of $p(\boldsymbol{x}|\theta_k)$ depends on various factors
  - Nature of input features, e.g.,
    - If $\boldsymbol{x} \in \mathbb{R}^D$, can use a D-dim Gaussian $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
    - If $\boldsymbol{x} \in \{0,1\}^D$, can use D Bernoullis (one for each feature)
  - Amount of training data available (important)
    - If D large and N_k small, it will be difficult to get a good estimate θ_k

In such cases, we may need to regularize θ_k or make some simplifying assumptions on $p(\boldsymbol{x}|\theta_k)$, such as features being conditionally independent given class e.g., $p(\boldsymbol{x}|\theta_k) = \prod_{d=1}^{D} p(x_d|\theta_{kd})$ - naïve Bayes

Especially if the number of features (D) is very large because large value of D means k consists of a large number of parameters (e.g., in the Gaussian case, $\theta_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), D$ params for $\boldsymbol{\mu}_k$ and $O(D^2)$ params for $\boldsymbol{\Sigma}_k$. Can overfit

Generative Classification: At Test Time

- Recall the form of the conditional distribution of the label

Class-marginal accounts for the frequency of class k labels in the training data

Class-conditional distribution of inputs accounts for the shape/spread of class k

$$p(y_* = k|\boldsymbol{x}_*) = \frac{p(y_* = k) \times p(\boldsymbol{x}_*|y_* = k)}{p(\boldsymbol{x}_*)}$$

$$\propto p(y_* = k) \times p(\boldsymbol{x}_*|y_* = k)$$

$$\propto \hat{\pi}_k \times p(\boldsymbol{x}_*|\hat{\theta}_k)$$

Probability of x_* belonging to class k is proportional to the fraction of training inputs from class k times the probability of x_* under the distribution of inputs from class k

- If we assume the class-marginal to be uniform $(p(y_* = k) = 1/K)$ then

$$p(y_* = k|\boldsymbol{x}_*) \propto p(\boldsymbol{x}_*|\hat{\theta}_k)$$

- The most likely label is $y_* = \operatorname{argmax}_{k \in \{1,2,\ldots,K\}} p(y_* = k|\boldsymbol{x}_*)$

Gen. Class. using Gaussian Class-conditionals

- The generative classification model $p(y=k|\boldsymbol{x}) = \dfrac{p(y=k)p(\boldsymbol{x}|y=k)}{p(\boldsymbol{x})}$

A benefit of modeling each class by a distribution

- Assume each class-conditional $p(\boldsymbol{x}|y=k)$ to be a Gaussian

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma}_k|}}\exp[-(\boldsymbol{x}-\boldsymbol{\mu}_k)^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)]$$

Since the Gaussian's covariance models its shape, we can learn the shape of each class

- Class marginal is multinoulli $p(y=k) = \pi_k, \pi_k \in (0,1), \sum_{k=1}^K \pi_k = 1$
- Let's denote the parameters of the model collectively by $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$
  - Can estimate these using MLE/MAP/Bayesian inference
  - Already saw the MLE solution for $\boldsymbol{\pi}$: $\pi_k = N_k/N$ (can also do MAP)
  - MLE solution for $\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{y_n=k}\boldsymbol{x}_n$, $\boldsymbol{\Sigma}_k = \frac{1}{N_k}\sum_{y_n=k}(\boldsymbol{x}_n-\boldsymbol{\mu}_k)(\boldsymbol{x}_n-\boldsymbol{\mu}_k)^\top$

Can also do MAP estimation for $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ using a Gaussian prior on $\boldsymbol{\mu}_k$ and inverse Wishart prior on $\boldsymbol{\Sigma}_k$

Exercise: Try to derive this. I will provide a separate note containing the derivation

- If using point est (MLE/MAP) for $\boldsymbol{\theta}$, predictive distribution will be

Can predict the most likely class for the test input $\boldsymbol{x}_*$ by comparing these probabilities for all values of k

$$p(y_* = k|\mathbf{x}_*, \theta) = \frac{\pi_k|\boldsymbol{\Sigma}_k|^{-1/2}\exp\left[-\frac{1}{2}(\mathbf{x}_* - \boldsymbol{\mu}_k)^\top\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_* - \boldsymbol{\mu}_k)\right]}{\sum_{k=1}^K \pi_k|\boldsymbol{\Sigma}_k|^{-1/2}\exp\left[-\frac{1}{2}(\mathbf{x}_* - \boldsymbol{\mu}_k)^\top\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_* - \boldsymbol{\mu}_k)\right]}$$

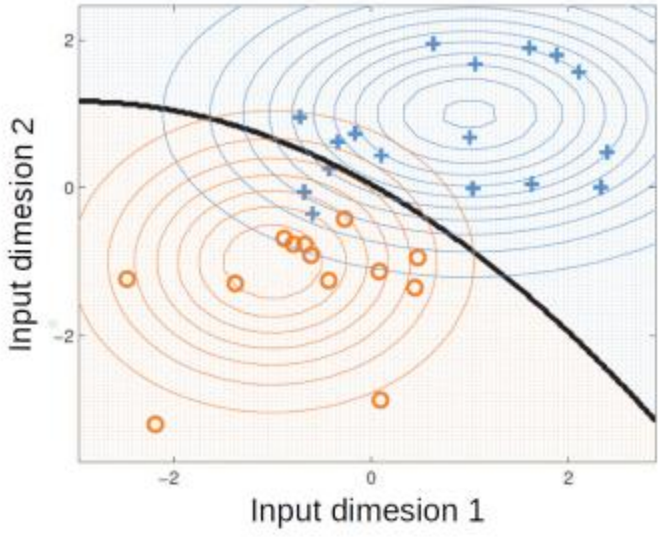
Note that the exponent has a Mahalanobis distance like term. Also, accounts for the fraction of training examples in class k

Decision Boundary with Gaussian Class-Conditional

- As we saw, the prediction rule when using Gaussian class-conditional

$$p(y = k|\mathbf{x}, \theta) = \frac{\pi_k |\boldsymbol{\Sigma}_k|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]}{\sum_{k=1}^{K} \pi_k |\boldsymbol{\Sigma}_k|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]}$$

- The decision boundary between any pair of classes will be a quadratic curve



Reason: For any two classes k and k' at the decision boundary, we will have $p(y = k|x, \theta) = p(y = k'|x, \theta)$. Comparing their logs and ignoring terms that don't contain $\boldsymbol{x}$, can easily see that

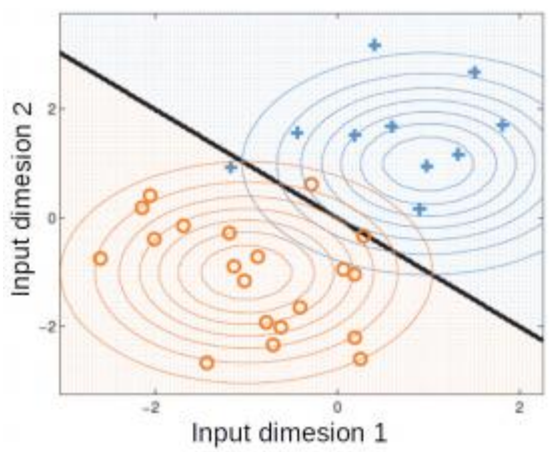
$$(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k) - (\mathbf{x} - \boldsymbol{\mu}_{k'})^\top \boldsymbol{\Sigma}_{k'}^{-1}(\mathbf{x} - \boldsymbol{\mu}_{k'}) = 0$$

Decision boundary contains all inputs $\boldsymbol{x}$ that satisfy the above

This is a quadratic function of $\boldsymbol{x}$ (this model is sometimes referred to Quadratic Discriminant Analysis)

Decision Boundary with Gaussian Class-Conditional

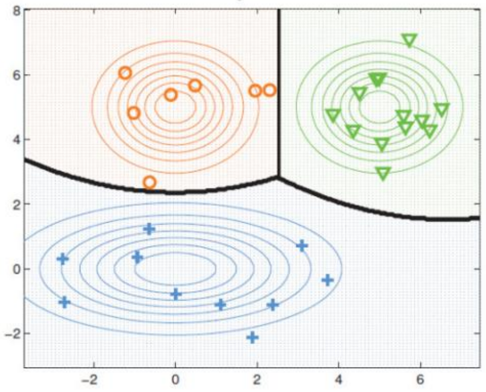
- Assume all classes are modeled using the same covariance matrix $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}, \forall k$
- In this case, the decision boundary b/w any pair of classes will be linear

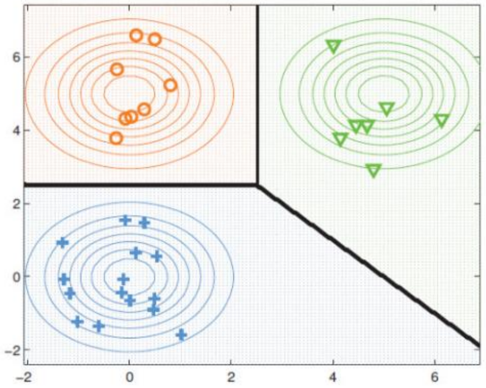


Reason: Again using $p(y=k|x,\theta) = p(y=k'|x,\theta)$, comparing their logs and ignoring terms that don't contain $\boldsymbol{x}$, we have

$$(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) - (\mathbf{x}-\boldsymbol{\mu}_{k'})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{k'}) = 0$$

Quadratic terms of $\boldsymbol{x}$ will cancel out; only linear terms will remain; hence decision boundary will be a linear function of $\boldsymbol{x}$ (Exercise: Verify that we can indeed write the decision boundary between this pair of classes as $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$ where $\boldsymbol{w}$ and b depend on $\boldsymbol{\mu}_k$, $\boldsymbol{\mu}_{k'}$ and $\boldsymbol{\Sigma}$)





If we assume the covariance matrices of the assumed Gaussian class-conditionals for any pair of classes to be equal, then the learned separation boundary between this pair of classes will be linear; otherwise, quadratic as shown in the figure on left

Generative Models for Regression

- We can even model regression problems using a generative approach
- Note that the output $\boldsymbol{y}$ is not longer discrete (so no notion of a class-conditional)
- However, the basic rule of recovering a conditional from joint would still apply

$$p(y|\boldsymbol{x},\theta) = \frac{p(\boldsymbol{x},y|\theta)}{p(\boldsymbol{x}|\theta)}$$

A benefit of modeling each class by a distribution

- Thus we can model the joint distribution $p(\boldsymbol{x},y|\theta)$ of features $\boldsymbol{x}$ and outputs $\boldsymbol{y} \in \mathbb{R}$
  - If features are real-valued the we can model $p(\boldsymbol{x},y|\theta)$ using a $(D+1)$-dim Gaussian
  - From this $(D+1)$-dim Gaussian, we can get $p(y|\boldsymbol{x},\theta)$using Gaussian conditioning formula
  - If joint is Gaussian, any subset of variables ($\boldsymbol{y}$ here), given the rest ($\boldsymbol{x}$ here) is also a Gaussian!
  - Refer to the Gaussian results from maths refresher slides for the result

References

- **Section 15.3** Kevin Murphy, Probabilistic Machine Learning: Advanced Topics, MIT Press, 2022 (freely available online)
- **Section 1-3** Michael E. Tipping, "Bayesian inference: An introduction to principles and practice in machine learning." Summer school on machine learning. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. 41-62.
- **Chapter 9**, Kevin Murphy, Probabilistic Machine Learning: An Introduction
- **Lectures 5, 6, 8** Piyush Rai, Probabilistic Machine Learning (CS772A)
- **Lecture 15**, Piyush Rai, Introduction to Machine Learning (CS771A)