# Bayesian ML

CS771: Introduction to Machine Learning

Purushottam Kar

# Announcement



- Some groups submitted code using homepages of non-group members
  - *These submissions will be examined "more closely"*
  - *From next time onwards – please use homepage of a group member only*
  - *From next time, will also introduce use of password-protected archives*
- Code submissions were downloaded from the links ~11:10PM, Sep 07
  - *Any changes you made after that wont be graded since we won't have it!*
- Second assignment coming up in a day or two
  - *Will have no restrictions on library/external code usage so long as you cite/acknowledge the usage*
  - *Will essentially ask you to create a recommendation system*

# Recap of Last Lecture



Notion of random vectors

- *Defined simply as an ordered array of random variables*
- *However, beautiful and powerful results emerge from this ordering*

Notions of mean, mode, median

- *No unique notion of median for random vectors*

Notion of random matrices

- *Defined simply as a bunch of random variables ordered as a matrix*
- *Covariance matrix of a random vector*
- *Covariance matrix between two random vectors*

Useful operations involving random vectors/covariance matrices

# Gaussian Random Vector



As in the scalar case, the *multivariate* Gaussian requires just the mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and the covariance $\Sigma \in \mathbb{R}^{d \times d}$ to be specified $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$

$$\mathbb{P}[\mathbf{x} \mid \boldsymbol{\mu}, \Sigma] = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

Special case $\boldsymbol{\mu} = \mathbf{0}$ and $\Sigma = I_d$ called *standard Gaussian/Normal dist*

$$\mathbb{P}[\mathbf{x} \mid \mathbf{0}, I_d] = \frac{1}{\sqrt{(2\pi)^d}} \exp\left(-\frac{1}{2}\|\mathbf{x}\|_2^2\right) = \prod_{i=1}^d \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x_i^2\right)$$

However, $\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x_i^2\right)$ is simply $\mathcal{N}(0,1)$ i.e. we indeed have

$$\mathbb{P}[x_1, \ldots, x_d \mid \mathbf{0}, \mathrm{I}] = \prod_{i=1}^d \mathbb{P}[x_i \mid 0,1]$$

All $d$ coordinates of a standard Gaussian r.vec. are independent!

# Gaussian Random Vector



Given a (possibly non-standard) Gaussian vector $\mathbb{R}^d \ni \mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$

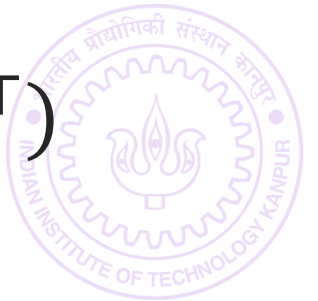
- *Every coordinate of $\mathbf{x}$ is a (real valued) Gaussian r.v. – warning: coordinates need not be independent if the Gaussian is non-standard*
- *The above holds true even if conditioned on all other coordinates of $\mathbf{x}$*

Consider any coordinate of the vector, say $j \in [d]$

- *$x_j$ is distributed as the Gaussian $\mathcal{N}(\mu_j, \Sigma_{jj})$*
- *Given values $x_k = v_k$ for all other coordinates $k \neq j$, $x_j$ is still Gaussian*
- *Expression a bit complicated – refer to DFO Sec 6.5.1*

If $\mathbf{a} \in \mathbb{R}^d$ is a constant vector, then $\mathbb{R} \ni \mathbf{a}^\top \mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}^\top \mathbf{a}, \mathbf{a}^\top \Sigma \mathbf{a})$

If $A \in \mathbb{R}^{n \times d}$ is a constant matrix then $\mathbb{R}^d \ni A\mathbf{x} \sim \mathcal{N}(A\boldsymbol{\mu}, A\Sigma A^\top)$



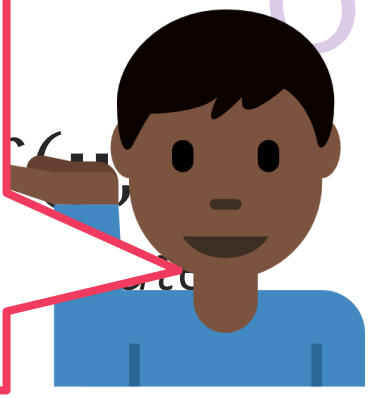
# Gau

Given

*Eve*
*nee*

*The above holds true even if conditioned on all other coordinates of* $\mathbf{x}$

**Note**: Just as before, we can derive results such as $\mathbb{E}[\mathbf{a}^\top\mathbf{x}] = \boldsymbol{\mu}^\top\mathbf{x}$ and $\mathbb{V}[\mathbf{a}^\top\mathbf{x}] = \mathbf{a}^\top\boldsymbol{\Sigma}\mathbf{a}$ using rules we studied earlier. However, those rules do not assure us that $\mathbf{a}^\top\mathbf{x}$ or $A\mathbf{x}$ must be Gaussian (they just assure us that these are some r.v./r.vec. with such and such mean and (co)-variance. It takes special analysis to show that these are actually Gaussian.



Consider any coordinate of the vector, say $j \in [d]$

*$x_j$ is distributed as the Gaussian $\mathcal{N}(\mu_j, \Sigma_{jj})$*

*Given values $x_k = v_k$ for all other coordinates $k \neq j$, $x_j$ is still Gaussian*

*Expression a bit complicated – refer to DFO Sec 6.5.1*

If $\mathbf{a} \in \mathbb{R}^d$ is a constant vector, then $\mathbb{R} \ni \mathbf{a}^\top\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}^\top\mathbf{a}, \mathbf{a}^\top\boldsymbol{\Sigma}\mathbf{a})$

If $A \in \mathbb{R}^{n\times d}$ is a constant matrix then $\mathbb{R}^d \ni A\mathbf{x} \sim \mathcal{N}(A\boldsymbol{\mu}, A\boldsymbol{\Sigma}A^\top)$

# Probabilistic Regression Revisited



To perform probabilistic regression, need to assign a label distribution over all $\mathbb{R}$ for every data point $\mathbf{x}$

*Had it been binary classification, we would have assigned a dist over* $\{-1,1\}$

*We assume a observation model (likelihood function)*

$$\mathbb{P}\left[y^i \mid \mathbf{x}^i, \mathbf{w}\right] = \mathcal{N}\left(y^i \mid \mathbf{w}^\top \mathbf{x}^i, \sigma_l^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\left(y^i - \mathbf{w}^\top \mathbf{x}^i\right)^2 / 2\sigma_l^2\right)$$

Properties of (univariate) Gaussian tells us that this is same as saying

$y^i = \mathbf{w}^\top \mathbf{x}^i + \epsilon^i$ where $\epsilon^i \sim \mathcal{N}(0, \sigma^2)$

Also, let us assume that we like model vectors $\mathbf{w}$ with small L2 norm better than those with larger L2 norm i.e. have a prior $\mathcal{N}(\mathbf{0}, I_d)$

$$\mathbb{P}[\mathbf{w}] = \mathcal{N}\left(\mathbf{w} \mid \mathbf{0}, \sigma_p^2 \cdot I_d\right) = \frac{1}{\left(2\pi\sigma_p^2\right)^{d/2}} \exp\left(-\frac{1}{2\sigma_p^2} \|\mathbf{w}\|_2^2\right)$$

# Probabilistic Regression Revisited



Recall that the prior encodes our beliefs before we have seen data

Posterior encodes our beliefs after we have seen data – Bayes Rule

$$\mathbb{P}[\mathbf{w} \mid y^1 \dots y^n, \mathbf{x}^1 \dots \mathbf{x}^n] = \frac{\mathbb{P}[y^1 \dots y^n, \mathbf{x}^1 \dots \mathbf{x}^n \mid \mathbf{w}] \cdot \mathbb{P}[\mathbf{w}]}{\mathbb{P}[y^1 \dots y^n, \mathbf{x}^1 \dots \mathbf{x}^n]}$$

Using independence gives us $= \frac{\mathbb{P}[\mathbf{w}] \cdot \prod_{i=1}^{n} \mathbb{P}[y^i, \mathbf{x}^i \mid \mathbf{w}]}{\prod_{i=1}^{n} \mathbb{P}[y^i, \mathbf{x}^i]}$

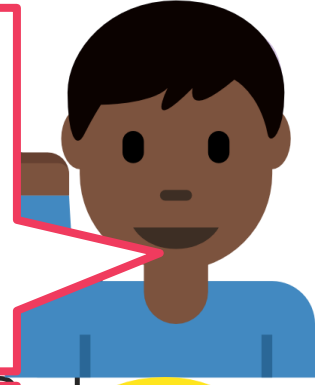
Ignoring terms that don't involve $\mathbf{w}$, taking logs gives us MAP estimate

$$\widehat{\mathbf{w}}_{\text{MAP}} = \arg\min_{\mathbf{w} \in \mathbb{R}^d} -\ln \mathbb{P}[\mathbf{w}] - \sum_{i=1}^{n} \ln \mathbb{P}[y^i, \mathbf{x}^i \mid \mathbf{w}]$$

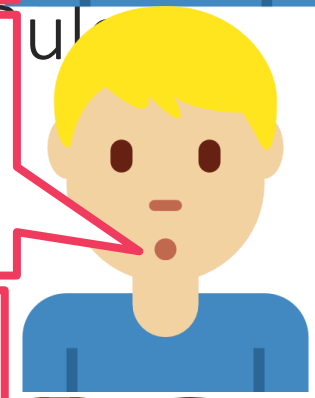
Note: Likelihood is a distribution over labels i.e. over $\mathbb{R}$ but the prior and posterior are distributions over models i.e. over $\mathbb{R}^d$

If we do these steps for Gaussian likelihood $\mathcal{N}\left(y^i \mid \mathbf{w}^\top \mathbf{x}^i, \sigma_l^2\right)$ and Gaussian prior $\mathcal{N}\left(\mathbf{w} \mid \mathbf{0}, \sigma_p^2 \cdot I_d\right)$ then we get (careful, there are two variance terms here $\sigma_l, \sigma_p$

$$\hat{\mathbf{w}}_{\text{MAP}} = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{\sigma_p^2}\|\mathbf{w}\|_2^2 + \frac{1}{\sigma_l^2}\sum_{i=1}^{n}\left(y^i - \mathbf{w}^\top \mathbf{x}^i\right)^2$$



This is the same as $\arg\min_{\mathbf{w}\in\mathbb{R}^d} \left(\frac{\sigma_l}{\sigma_p}\right)^2 \cdot \|w\|_2^2 + \sum_{i=1}^{n}\left(y^i - \mathbf{w}^\top \mathbf{x}^i\right)^2$. This means that $\sigma_l, \sigma_p$ together decide the regularization constant!

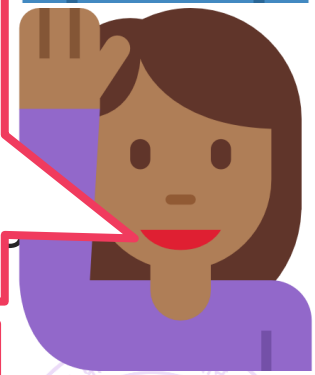


Using indep

Ignoring ter

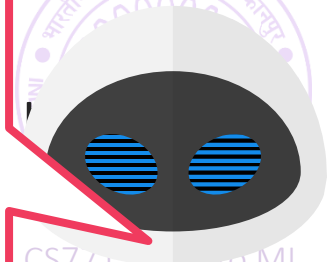
There is a multivariate version of the Laplace distribution as well! Using it as a prior with $\boldsymbol{\mu}=\mathbf{0}$ will give us L1 regularization! (**Warning**: expression for the Laplace PDF for general covariance is a bit tricky)

$$\mathcal{L}(\mathbf{w} \mid \boldsymbol{\mu}, \sigma) = \frac{1}{(2\sigma)^d}\exp\left(-\frac{\|\mathbf{w}-\boldsymbol{\mu}\|_1}{\sigma}\right)$$



Just so that we are clear, nothing special about $\mathcal{N}\left(\mathbf{w} \mid \mathbf{0}, \sigma_p^2 \cdot I_d\right)$. If we believe $\mathbf{w}$ is close to a vector $\mathbf{v}$, should use $\mathcal{N}\left(\mathbf{w} \mid \mathbf{v}, \sigma_p^2 \cdot I_d\right)$ instead. MAP will become

$$\hat{\mathbf{w}}_{\text{MAP}} = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{\sigma_p^2}\|\mathbf{w}-\mathbf{v}\|_2^2 + \frac{1}{\sigma_l^2}\sum_{i=1}^{n}\left(y^i - \mathbf{w}^\top \mathbf{x}^i\right)^2$$



# Bayesian Learning



Before we started doing probabilistic ML, we output a single label. With PML we started giving a distribution over labels instead

However, we still do so using a single model

- *In MLE we use the model with highest likelihood function value to do so*
- *In MAP we use the mode of the posterior distribution to do so*

In Bayesian learning, we take this philosophy further – instead of trusting a single model, place partial trust, possibly over all models

- *Models with high posterior probability (density) value get high trust*
- *Models with low posterior probability (density) value get low trust*

We use Bayes rule yet again to perform these calculations

# From PML to BML



I have with me data points $\left(\mathbf{x}^i, y^i\right)$, and a prior over models $\mathbb{P}[\mathbf{w}]$

*For a test point $\mathbf{x}^t$, I wish to output a distribution over set of all labels i.e. $\mathbb{P}\left[y \mid \mathbf{x}^t, \left\{\mathbf{x}^i, y^i\right\}\right]$ – we condition on available data and $\mathbf{x}^t$ as we know these*

*Since we cannot predict labels in absence of models, let us introduce them*

$$\mathbb{P}\left[y \mid \mathbf{x}^t, \left\{\mathbf{x}^i, y^i\right\}\right] = \int_{\mathbb{R}^d} \mathbb{P}\left[y, \mathbf{w} \mid \mathbf{x}^t, \left\{\mathbf{x}^i, y^i\right\}\right] \, d\mathbf{w}$$

$$= \int_{\mathbb{R}^d} \mathbb{P}\left[y \mid \mathbf{w}, \mathbf{x}^t, \left\{\mathbf{x}^i, y^i\right\}\right] \cdot \mathbb{P}\left[\mathbf{w} \mid \mathbf{x}^t, \left\{\mathbf{x}^i, y^i\right\}\right] d\mathbf{w}$$

$$= \int_{\mathbb{R}^d} \mathbb{P}\left[y \mid \mathbf{w}, \mathbf{x}^t\right] \cdot \mathbb{P}\left[\mathbf{w} \mid \left\{\mathbf{x}^i, y^i\right\}\right] d\mathbf{w}$$

*Step 1 (law of total probability) Step 2(chain rule of probability), Step 3(get rid of conditionings that did not matter)*

***Note**: $\mathbb{P}\left[y \mid \mathbf{w}, \mathbf{x}^t\right]$ is the distribution we would have given had $\mathbf{w}$ indeed been the true model and $\mathbb{P}\left[\mathbf{w} \mid \left\{\mathbf{x}^i, y^i\right\}\right]$ is our faith in $\mathbf{w}$ being the true model!*

# BML Trivia



$\mathbb{P}\left[y \mid \mathbf{x}^t, \{\mathbf{x}^i, y^i\}\right]$ is called the *predictive posterior*

*Careful: predictive posterior is a distribution over labels (not models)*

For some very well behaved cases, the posterior and the predictive posterior distributions have closed form expressions

*The special cases where we have something called* conjugate priors *are one such example*

In all the other cases, we must use other techniques to work with the (predictive) posteriors in an approximate manner

Powerful sampling algorithms e.g. MCMC, Gibbs etc exist

Discussion beyond the scope of CS771 – courses like CS772 discuss this

# Bayesian Regression



Suppose we have Gaussian likelihood $\mathcal{N}\left(y^i \mid \mathbf{w}^\top \mathbf{x}^i, \sigma_l^2\right)$ and Gaussian prior $\mathcal{N}\left(\mathbf{w} \mid \mathbf{0}, \sigma_p^2 \cdot I_d\right)$, then we have $\mathbb{P}\left[\mathbf{w} \mid \left\{\mathbf{x}^i, y^i\right\}\right] = \mathcal{N}\left(\mathbf{w}; \hat{\boldsymbol{\mu}}, \hat{\Sigma}\right)$

$$\hat{\boldsymbol{\mu}} = \left(X^\top X + \left(\frac{\sigma_l}{\sigma_p}\right)^2 \cdot I_d\right)^{-1} \cdot X^\top \mathbf{y} \text{ and } \hat{\Sigma} = \frac{1}{\sigma_p^2}\left(X^\top X + \left(\frac{\sigma_l}{\sigma_p}\right)^2 \cdot I_d\right)^{-1}$$

Note that the $\hat{\boldsymbol{\mu}}$ is simply the MAP solution – makes sense since MAP is the mode of the posterior and for Gaussian, mean is mode

**Predictive Posterior**: $\mathbb{P}\left[y \mid \mathbf{x}, \left\{\left\{\mathbf{x}^i, y^i\right\}\right\}\right] = \mathcal{N}\left(y; \hat{\mu}_{\mathbf{x}}, \hat{\sigma}_{\mathbf{x}}^2\right)$ where

$\hat{\mu}_{\mathbf{x}} = \hat{\boldsymbol{\mu}}^\top \mathbf{x}$ and $\hat{\sigma}_{\mathbf{x}}^2 = \sigma_l^2 + \mathbf{x}^\top \hat{\Sigma} \mathbf{x}$

**Note:** in this case, variance of the predicted distribution $\hat{\sigma}_{\mathbf{x}}^2$ depends on the point itself – can deduce if the prediction is confident or not!

# Conjugate Priors



Bayesian Logistic Regression or Softmax Regression is not nearly as pretty – no closed form solutions for posterior or predictive posterior

However, some likelihood-prior pairs are special

- *Always yield a posterior that is of the same family as the prior*
- *Such pairs of distributions are called* conjugate pairs
- *The prior in such cases is said to be* conjugate *to the likelihood*
- *Gaussian-Gaussian is one example – warning: one Gaussian is a likelihood over reals, the other Gaussian is a prior over models (i.e. vectors)*
- *Other conjugate pairs exist too*

Discussion beyond the scope of CS771 – courses like CS772 discuss this

# Probabilistic Clustering??



Recall the k-means algorithm: we assign every data point to the closest cluster and update

Can we make this assignment "softer" by predicting a distribution over all centers?

To do this properly, again need some notion of likelihood!

What is probability that blah centroid $\boldsymbol{\mu}$ thinks blah data point $\mathbf{x}$ belongs to its cluster?

**Popular choice**: Gaussian!!

K-MEANS/LLOYD'S ALGORITHM

1. Initialize means $\{\boldsymbol{\mu}^c\}_{c=1\ldots C}$
2. For $i \in [n]$, update $z_i$ using $\{\boldsymbol{\mu}^c\}$
   1. Let $z_i = \arg\min_c \left\|\mathbf{x}^i - \boldsymbol{\mu}^c\right\|_2^2$
3. Let $n_c = \#$ points assigned to $c$
4. Update $\boldsymbol{\mu}^c = \frac{1}{n_c}\sum_{i:z_i=c} \mathbf{x}^i$
5. Repeat until convergence

$$\mathbb{P}[\mathbf{x} \mid \boldsymbol{\mu}] = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, I_d)$$

# Soft k-means



At every time step, instead of choosing one value of $z_i \in [C]$, output a distribution over all $C$ possible values of $z_i$

$$\mathbb{P}\left[z_i = c \mid \mathbf{x}^i, \{\boldsymbol{\mu}^j\}\right] \propto \mathbb{P}\left[\mathbf{x}^i \mid z_i = c, \{\boldsymbol{\mu}^j\}\right] \cdot \mathbb{P}\left[z_i = c \mid \{\boldsymbol{\mu}^j\}\right] \propto \mathcal{N}\left(\mathbf{x}^i; \boldsymbol{\mu}^c, I_d\right)$$

*For sake of simplicity, assume* $\mathbb{P}\left[z_i = c \mid \{\boldsymbol{\mu}^j\}\right] = \mathbb{P}[z_i = c] = \frac{1}{C}$

*Akin to assuming that all clusters are equi-probable (may change this too)*

*Denote* $\pi_c^i = \mathbb{P}\left[z_i = c \mid \mathbf{x}^i, \{\boldsymbol{\mu}^j\}\right]$ *i.e.* $\pi_c^i \geq 0$ *and* $\sum_{c=1}^{C} \pi_c^i = 1$

One these "partial" assignments are made, proceed as usual

*Pretend that* $\pi_c^i$*-th portion of the* $i$*-th data point went to* $c$*-th cluster*

*"Number" of points assigned to cluster* $c$ *becomes* $n_c = \sum_{i=1}^{n} \pi_c^i$

*Rest of the update modified accordingly* $\boldsymbol{\mu}^c = \frac{1}{n_c} \sum_{i=1}^{n} \pi_c^i \cdot \mathbf{x}^i$

Much more principled derivation of this algorithm exists – later!

# Soft k-means



At every time st… ∈ [C], output a distribution ove…

$\mathbb{P}[z_i = c \mid \mathbf{x}^i, \{$ … $] \propto \mathcal{N}(\mathbf{x}^i; \boldsymbol{\mu}^c, I_d)$

*For sake of sim…* $= \frac{1}{C}$

*Akin to assumin… ange this too)*

*Denote* $\pi_c^i = \mathbb{P}$ … 1

One these "part… usual

*Pretend that* $\pi_c^i$ … *h cluster*

*"Number" of p…* $\sum_{i=1}^n \pi_c^i$

*Rest of the upd…* $i$

## SOFT K-MEANS ALGORITHM

1. Initialize means $\{\boldsymbol{\mu}^c\}_{c=1\ldots C}$
2. For $i \in [n]$, update $\pi_c^i$ using $\{\boldsymbol{\mu}^c\}$
   1. Let $p_c^i = \exp\left(-\frac{\|\mathbf{x}^i - \boldsymbol{\mu}^c\|_2^2}{2}\right)$
   2. Let $\pi_c^i = \frac{p_c^i}{\sum_{c=1}^C p_c^i}$ (normalize)
3. Let $n_c = \sum_{i=1}^n \pi_c^i$
4. Update $\boldsymbol{\mu}^c = \frac{1}{n_c} \sum_{i=1}^n \pi_c^i \cdot \mathbf{x}^i$
5. Repeat until convergence

Much more principled derivation of this algorithm exists – later!

# Generative Models



So far, we looked at probability theory as a tool to express the belief of an ML algorithm that the true label is such and such

*Likelihood: given model $\theta$ it tells us $\mathbb{P}[y \mid \mathbf{x}, \theta]$*

We also looked at how to use probability theory to express our beliefs about which models are preferred by us and which are not

*Prior: this just tells us $\mathbb{P}[\theta]$*

Notice that in all of this, the data features were always considered constant and never questions as being random or flexible

*Can we also talk about $\mathbb{P}[\mathbf{x} \mid y, \theta]$?*

*Very beneficial: given label $y$, this would allow us to generate a new $\mathbf{x}$ from the distribution $\mathbb{P}[\mathbf{x} \mid y, \theta]$?*

*Can generate new cat images, new laptop designs (GANs do this very thing!)*