Probabilistic ML

CS771: Introduction to Machine Learning

Purushottam Kar

Probabilistic ML

Till now we have looked at ML techniques that assign a label for every data point (the label is from the set $\{-1, +1\}$ for binary classification, $[C]$ for multiclass classification with C classes, $\mathbb{R}$ for regression etc)

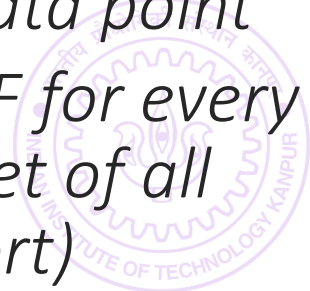
Examples include LwP, kNN, DT, linear models

Probabilistic ML techniques, given a data point, do not output a single label, they instead output a distribution over all possible labels

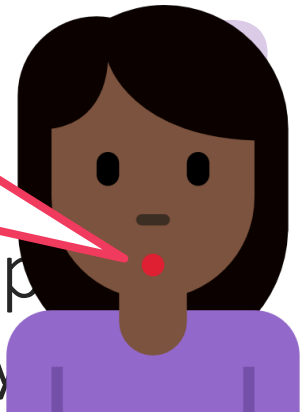
For binary classification, output a PMF over $\{-1,1\}$, for multiclassification, output a PMF over $\{1,2,\ldots,C\}$, for regression, output a PDF over $\mathbb{R}$

The probability mass/density of a label in the output PMF/PDF indicates how likely does the ML model think that label is the correct one for that data point

***Note**: the algorithm is allowed to output a possibly different PMF/PDF for every data point. However, the support of these PMFs/PDFs is always the set of all possible labels (i.e. even very unlikely labels are included in the support)*



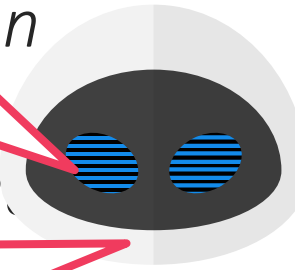
Exactly! Suppose we have three classes and for a data point, the ML model gives us the PMF $[0.3, 0.4, 0.3]$. The second class does win being the mode but the model seems not very certain about this prediction (only 40% confidence).



Say we have somehow learnt a PML model $\mathbf{w}$ which, for a data p

gives us a PMF $\mathbb{P}[Y \mid \mathbf{x}, \mathbf{w}]$ over the set of all possible labels, say

True! Suppose on another data point, the model gives us the PMF $[0.05, 0.1, 0.85]$. The third class wins being the mode and I extremely certain about this prediction (since I am giving a very high 85% confidence in this prediction).

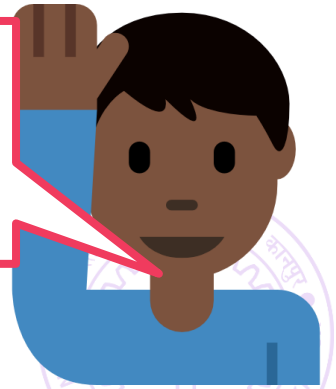


nevertheless fixed since we are looking at the *data point* $\mathbf{x}$ *using mod*

I could not agree more. However, in many ML applications (e.g. active learning) if we find that the model is making unsure predictions, we can switch to another model or just ask a human to step in. Thus, confidence info can be used fruitfully

$$\hat{y} = \arg\max \mathbb{P}[Y = y \mid \mathbf{x}, \mathbf{w}]$$

Warning! Just because a prediction is made with more confidence does not mean it must be correct. It may happen that the 40% confidence prediction in the first case is correct but the high 85% confidence prediction in the second case is wrong!



prediction or totally confused about which label is the correct one!

May use variance of $\mathbb{P}[Y \mid \mathbf{x}, \mathbf{w}]$ *to find this as well (low variance = very confident prediction and high variance = less confident/confused prediction)*

Probabilistic Binary Classification

Find a way to map every data point $\mathbf{x}$ to a Rademacher distribution

Another way of saying this: map every data point $\mathbf{x}$ *to a prob* $p_{\mathbf{x}} \in [0,1]$

Will give us a PMF $[1 - p_{\mathbf{x}}, p_{\mathbf{x}}]$ *i.e.* $\mathbb{P}[-1 \mid \mathbf{x}] = 1 - p_{\mathbf{x}}, \mathbb{P}[+1 \mid \mathbf{x}] = p_{\mathbf{x}}$

If using mode predictor i.e. $\hat{y} = \arg\max_{y \in \mathcal{Y}} \mathbb{P}[Y = y \mid \mathbf{x}]$ then this PMF will give us the correct label only if the following happens

When the true label of $\mathbf{x}$ *is* $+1$, $p_{\mathbf{x}} > 1 - p_{\mathbf{x}}$, *in other words* $p_{\mathbf{x}} > 0.5$

When the true label of $\mathbf{x}$ *is* -1, $1 - p_{\mathbf{x}} > p_{\mathbf{x}}$, *in other words* $p_{\mathbf{x}} < 0.5$

Note that if $p_{\mathbf{x}} = 0.5$, *it means ML model is totally confused about label of* $\mathbf{x}$

Data points for whom $p_{\mathbf{x}} = 0.5$ *are on decision boundary!!*

Of course, as usual we want a healthy margin

If true label of the data point $\mathbf{x}$ *is* $+1$, *then we want* $p_{\mathbf{x}} \gg 0.5$ *i.e.* $p_{\mathbf{x}} \approx 1$

If true label of the data point $\mathbf{x}$ *is* -1, *then we want* $p_{\mathbf{x}} \ll 0.5$ *i.e.* $p_{\mathbf{x}} \approx 0$

Probabilistic Binary Class

How to map feature vectors $\mathbf{x}$ to probability values $p_{\mathbf{x}} \in [0,1]$?

Could treat it as a regres … after all

Will need to modify the t… labels to 0 *since we want …*

Could use kNN, DT etc to solve this regression problem

Using linear models to do this presents a challenge

If …

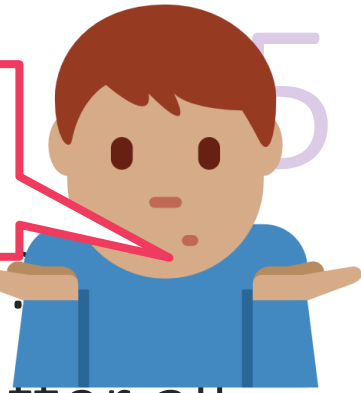
so …

$p_{\mathbf{x}}$ *wont make sense in this case – not a valid PMF!!*

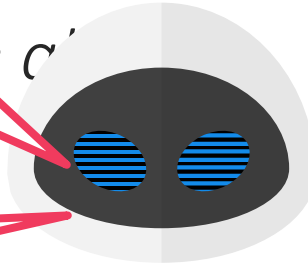
kNN, DT don't suffer from this problem since they always predict a $p_{\mathbf{x}} \in [0,1]$

kNN, DT use averages of a bunch of train labels to obtain test prediction – the average of a bunch of 0s and 1s is always a value in the range $[0,1]$

So can we never use linear models to do probabilistic ML?

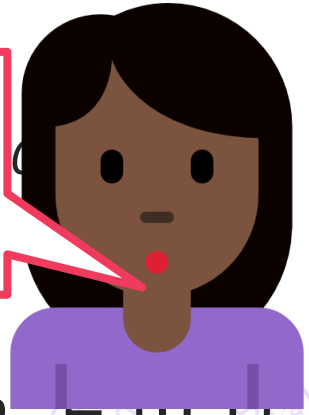


We can – one way to solve the problem of using linear methods to map $\mathbf{x} \mapsto [0,1]$ is called *logistic regression* – have seen it before



Yes, but there is a trick involved. Let us take a look at it

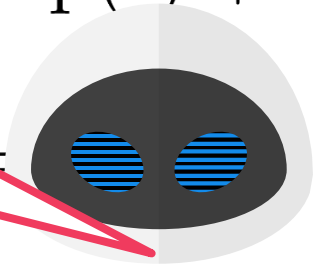
Ah! The name makes sense now – logistic regression is used to solve binary classification problems but since it does so by mapping $\mathbf{x} \mapsto [0,1]$, experts thought it would be cool to have the term "regression" in the name



Nice! So I want to learn a linear model $\mathbf{w}$ such that once I do this sigmoidal map, data points with label $+1$ get mapped to a probability value close to 1 whereas data points with label -1 get mapped to a probability value close to 0

$$\sigma(t) = \frac{1}{1+\exp(-t)} = \frac{}{\exp(t)+1}$$

There are several other such *wrapper/quashing/link/activation* functions which do similar jobs e.g. tanh, ramp, ReLU

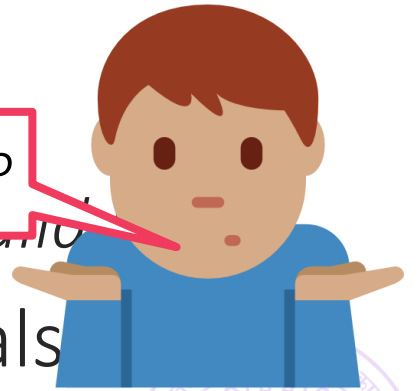


Trick: learn a linear model $\mathbf{w}$ and map $\mathbf{x} \mapsto \sigma(\mathbf{w}^\top \mathbf{x})$

May have an explicit/hidden bias ter

This will always give us a value in the range $[0,1]$*, hence give a valid*

How do I learn such a model $\mathbf{w}$?

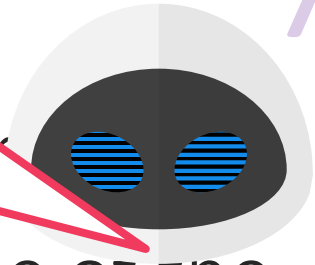


Note that $\sigma(t) > 0.5$ if $t > 0$ and $\sigma(t) < 0.5$ if $t < 0$ and als
$\sigma(t) \to 1$ as $t \to \infty$ and $\sigma(t) \to 0$ as $t \to -\infty$

This means that our sigmoidal map will predict $p_{\mathbf{x}} \approx 1$ *if* $\mathbf{w}^\top \mathbf{x} \gg 0$ *and* $p_{\mathbf{x}} \approx 0$ *if* $\mathbf{w}^\top \mathbf{x} \ll 0$

Li

Data might not actually be independent e.g. my visiting a website may not be independent from my friend visiting the same website if I have found an offer on that website and posted about it on social website. However, often we nevertheless assume independence to make life simple



Su

Given a data point $(\mathbf{x}^t, y^t)$, $\mathbf{x}^t \in \mathbb{R}^d$ and $y^t \in \{-1,1\}$, the use of the sigmoidal map gives us a Rademacher PMF $\mathbb{P}[\, y \mid \mathbf{x}^t, \mathbf{w}]$

The probability that this PMF gives to the correct label i.e. $\mathbb{P}[\, y^t \mid \mathbf{x}^t, \mathbf{w}]$ is called the *likelihood* of this model with respect to this data point

It easy to show that $\mathbb{P}[\, y^t \mid \mathbf{x}^t, \mathbf{w}] = \sigma(y^t \cdot \mathbf{w}^\top \mathbf{x}^t)$

Hint*: use the fact that* $\sigma(-t) = 1 - \sigma(t)$ *and that* $y^t \in \{-1,1\}$

If we have several points $(\mathbf{x}^1, y^1), \ldots, (\mathbf{x}^n, y^n)$ then we define the likelihood of $\mathbf{w}$ w.r.t entire dataset as $\mathbb{P}[y^1, \ldots, y^n \mid \mathbf{x}^1, \ldots, \mathbf{x}^n, \mathbf{w}]$

Usually we assume data points are independent so we use product rule to get

$\mathbb{P}[y^1, \ldots, y^n \mid \mathbf{x}^1, \ldots, \mathbf{x}^n, \mathbf{w}] = \prod_{i=1}^{n} \mathbb{P}[\, y^i \mid \mathbf{x}^i, \mathbf{w}] = \prod_{i=1}^{n} \sigma\left(y^i \cdot \mathbf{w}^\top \mathbf{x}^i\right)$

Maximum Likelihood

The expression $\mathbb{P}\left[y^i \mid \mathbf{x}^i, \mathbf{w} \right]$ tells us if the model $\mathbf{w}$ thinks the label y^i is a very likely label given the feature vector $\mathbf{x}^i$ or not likely at all!

$\mathbb{P}[y^1, \ldots, y^n \mid \mathbf{x}^1, \ldots, \mathbf{x}^n, \mathbf{w}]$ *similarly tells us how likely does the model* $\mathbf{w}$ *think the labels* $y^1, \ldots, y^n$ *are, given the feature vectors* $\mathbf{x}^1, \ldots, \mathbf{x}^n$

Since we trust our training data as clean and representative of reality, we should look for a $\mathbf{w}$ that considers training labels to be very likely

E.g. in RecSys example, let $y^t = 1$ *if customer makes a purchase and* $y^t = 0$ *otherwise. If we trust that these labels do represent reality i.e. what our customers like and dislike, then we should learn a model* $\mathbf{w}$ *accordingly*

Totally different story if we mistrust our data – different techniques for that

Maximum Likelihood Estimator (**MLE**): the model that gives highest likelihood to observed labels $\hat{\mathbf{w}}_{\text{MLE}} = \arg\max_{\mathbf{w} \in \mathbb{R}^d} \prod_{i=1}^{n} \mathbb{P}\left[y^i \mid \mathbf{x}^i, \mathbf{w} \right]$

Logistic Regression

Suppose we learn a model as the MLE while using sigmoidal map

$$\widehat{\mathbf{w}}_{\text{MLE}} = \arg\max_{\mathbf{w}\in\mathbb{R}^d} \prod_{i=1}^{n} \sigma\left(y^i \cdot \mathbf{w}^\top \mathbf{x}^i\right)$$

Working with products can be numerically unstable

Since $\sigma(\cdot) \in [0,1]$*, product of several such values can be extremely small*

Solution*: take logarithms and exploit that* $\max_{\mathbf{w}} f(\mathbf{w}) = \max_{\mathbf{w}} \ln\left(f(\mathbf{w})\right)$

$$\widehat{\mathbf{w}}_{\text{MLE}} = \arg\max_{\mathbf{w}\in\mathbb{R}^d} \ln\left(\prod_{i=1}^{n} \sigma\left(y^i \cdot \mathbf{w}^\top \mathbf{x}^i\right)\right)$$

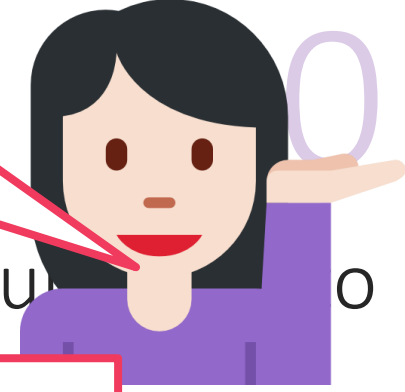
$$= \arg\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{n} \ln\left(1 + \exp\left(-y^i \cdot \mathbf{w}^\top \mathbf{x}^i\right)\right)$$

Also called *negative log-likelihood*

Thus, the logistic loss function pops out automatically when we try to learn a model that maximizes the likelihood function

Proba

Just as we had the Bernoulli distributions over the support $\{0,1\}$, if the support instead has $C > 2$ elements, then the distributions are called either *Multinoulli distributions* or *Categorical distributions*

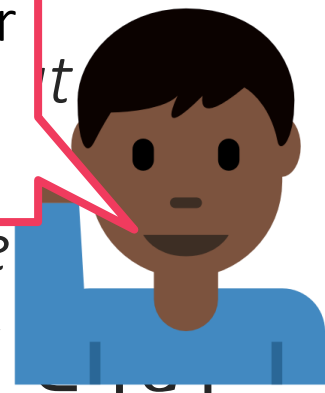


Suppose we have C classes, then for every data point we wou o output a PMF over the support $[C]$

Popular way: *assign a positive sco t scores form a proper probability d*

Common trick: *to convert any score to a positive score – exponentiate*

To specify a multinoulli distribution over C labels, we need to specify C non-negative numbers that add up to one



Learn C models $\mathbf{w}^1, \ldots, \mathbf{w}^C$, given a point $(\mathbf{x}^t, y^t)$, $\mathbf{x}^t \in \mathbb{R}^d$, $y^t \in [C]$

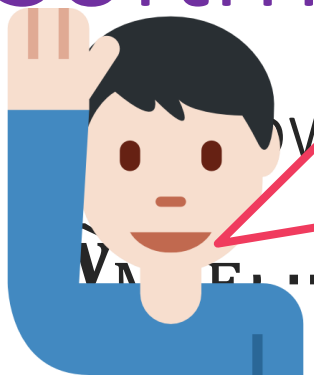
Assign a positive score per class $\eta_c = \exp(\langle \mathbf{w}^c, \mathbf{x}^t \rangle)$

Normalize to obtain a PMF $\mathbb{P}[\, y \mid \mathbf{x}^t, \{\mathbf{w}^c\}] = \eta_y / \sum_{c=1}^{C} \eta_c$ *for any* $y \in [C]$

Likelihood in this case is $\mathbb{P}[\, y^t \mid \mathbf{x}^t, \{\mathbf{w}^c\}] = \eta_{y^t} / \sum_{c=1}^{C} \eta_c$

Log-likelihood in this case is $\ln\left(\eta_{y^t} / \sum_{c=1}^{C} \eta_c\right)$

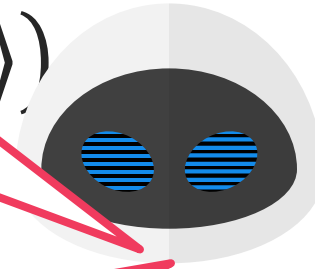
Softm



I may find other ways to assign a PMF over $[C]$ to each data point by choosing some function other than $\exp(\cdot)$ e.g. ReLU $[t]_+$ to assign positive scores i.e. let $\eta_c = [\langle \mathbf{w}^c, \mathbf{x}^t \rangle]_+$, let $\mathbb{P}[\, y \mid \mathbf{x}^t, \{\mathbf{w}^c\}] = \eta_y / \sum_{c=1}^{C} \eta_c$ and then proceed to obtain an MLE. Something similar to this is indeed used in deep learning

$$= \arg \max_{\mathbf{w}^1,\ldots,\mathbf{w}^C \in \mathbb{R}^d}$$

It should be noted that this is not the only way to do probabilistic multiclassification. It is just that this way is simple to understand, implement and hence popular

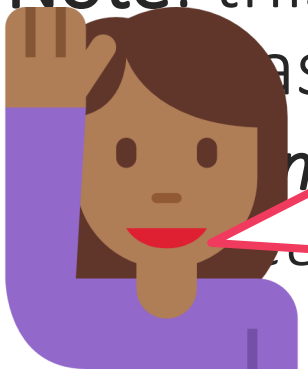


Using the negative

$$= \arg \min_{\mathbf{w}^1,\ldots,\mathbf{w}^C \in \mathbb{R}^d}$$

However, be warned that generating a PMF using DT/kNN need not necessarily be an MLE since we have not explicitly maximized any likelihood function here

Note: this is nothing but the softmax loss function we saw earlier, also



I could do also kNN or DT and invoke the "probability as proportions" interpretation to assign a test data point to a PMF that simply gives the proportion of each label in the neighbourhood/leaf of that data point!!

General Recipe for MLE Algorithms

Given a problem with label set $\mathcal{Y}$, find a way to map data features $\mathbf{x}$ to PMFs $\mathbb{P}[\cdot \mid \mathbf{x}, \mathbf{m}]$ with support $\mathcal{Y}$

The notation $\mathbf{m}$ *captures parameters in the model (e.g. vectors, bias terms)*

For binary classification, $\mathcal{Y} = \{-1, 1\}$ *and* $\mathbf{m} = \mathbf{w}$

For multiclassification, $\mathcal{Y} = [C]$ *and* $\mathbf{m} = \{\mathbf{w}^1, \ldots, \mathbf{w}^C\}$

The function $\mathbb{P}[\cdot \mid \mathbf{x}, \mathbf{m}]$ is often called the *likelihood function*

The function $-\ln \mathbb{P}[\cdot \mid \mathbf{x}, \mathbf{m}]$ called *negative log likelihood function*

Given data $\{(\mathbf{x}^i, y^i)\}_{i=1}^n$, find the model parameters that maximize likelihood function i.e. think that the training labels are very likely

$$\hat{\mathbf{m}}_{\text{MLE}} = \arg\min_{\mathbf{m}} \sum_{i=1}^{n} -\ln \mathbb{P}[y^i \mid \mathbf{x}^i, \mathbf{m}]$$

Probabilistic

In order to perform [illegible] distribution over all $\mathbb{R}$ for every data point $\mathbf{x}$ using a PDF

Suppose I decide to do that using a G[illegible] decide on a mean $\mu_{\mathbf{x}}$ and a variance [illegible]

Popular choice: Let $\mu_{\mathbf{x}} = \mathbf{w}^\top \mathbf{x}$ and $\sigma_{\mathbf{x}}^2 = \sigma^2$ i.e. $\mathcal{N}(\cdot \mid \mathbf{w}^\top \mathbf{x}, \sigma^2)$

We can also choose a different σ fo[illegible]

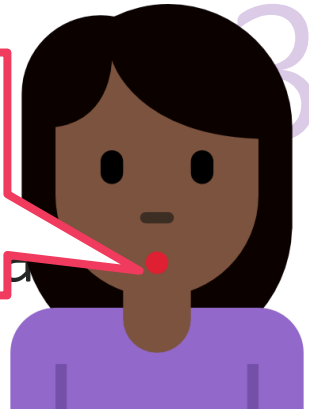
Likelihood function w.r.t a data po[illegible]

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

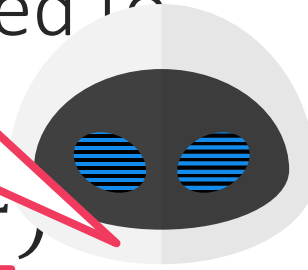
Negative log likelihood w.r.t a set of data points $\left\{\left(\mathbf{x}^i, y^i\right)\right\}_{i=1}^n$

$$\min_{\mathbf{w} \in \mathbb{R}^d} \sum_{i=1}^n \left(y^i - \mathbf{w}^\top \mathbf{x}^i\right)^2$$

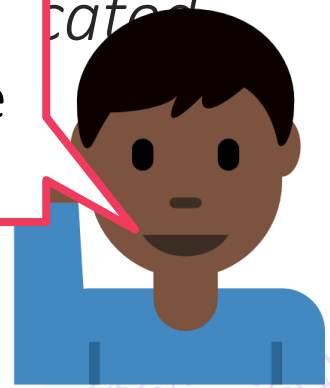
But apart from the first term and the scaling factor, both of which are constants and do not depend on the model $\mathbf{w}$ the rest is just the least squares loss term!



The MLE with respect to the Gaussian likelihood indeed the minimizes least squares loss



Also note that if we set all $\sigma_{\mathbf{x}}^2 = \sigma^2$ then it does not matter which σ we choose – will get the same model



Proba

Suppose [illegible] $\mu_{\mathbf{x}} = \mathbf{w}^\top \mathbf{x}$

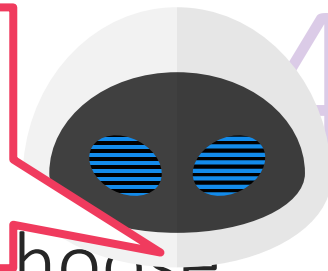
Likelihood

$$\text{Lap}(y^i \mid \mathbf{w}^\top \mathbf{x}^i, \sigma) = \frac{1}{2\sigma} \exp\left(-\left|y^i - \mathbf{w}^\top \mathbf{x}\right| / \sigma\right)$$

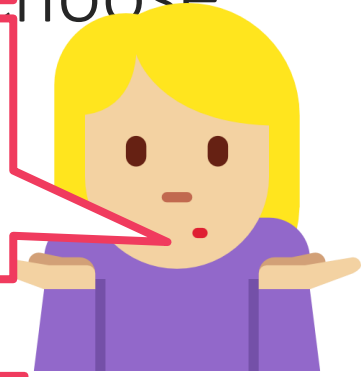
Negative log likelihood w.r.

$$\min_{\mathbf{w} \in \mathbb{R}^d} \; n \cdot \ln(\sigma) + \frac{1}{\sigma} \sum_{i=1}^{n}$$

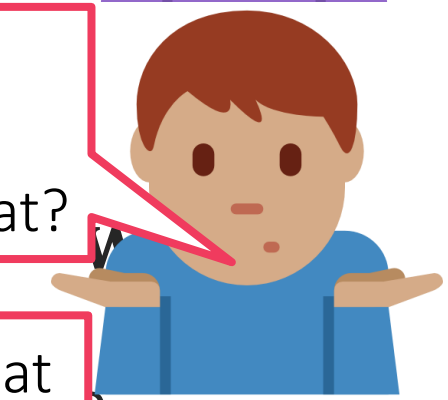
Be warned though – the σ we chose will start mattering the moment we add regularization! It is just that in these simple cases it does not matter. σ is usually treated like a hyperparameter and tuned.



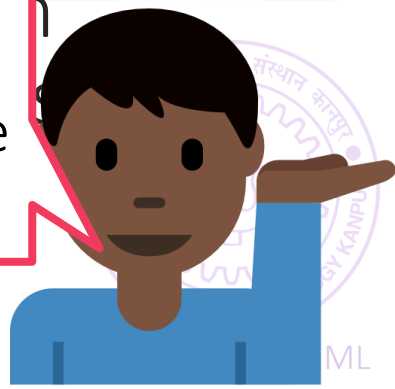
So I am a bit confused. All MLEs (classification/regression) demand a model that places maximum probability on the true label. Why don't we just ask the model to predict the true label itself?



That is like asking the PMF/PDF to place probability 1 on the true label and 0 everywhere else – why can't we do just that?



For the same reason we needed slack variables in CSVM – to allow for the fact that in realistic situations, no linear model may be able to do what we would ideally like. In probabilistic ML, allowing the model to place a less than 1 probability on the true label is much like a slack – allows us to learn good models even if not perfect ones



Probabilistic Regularization??

We have seen that MLE o… regression/least squares …

Even probabilistic methods can do regularization by way of *pri…*

Recall: regularization basically tells us which kinds of models we prefer

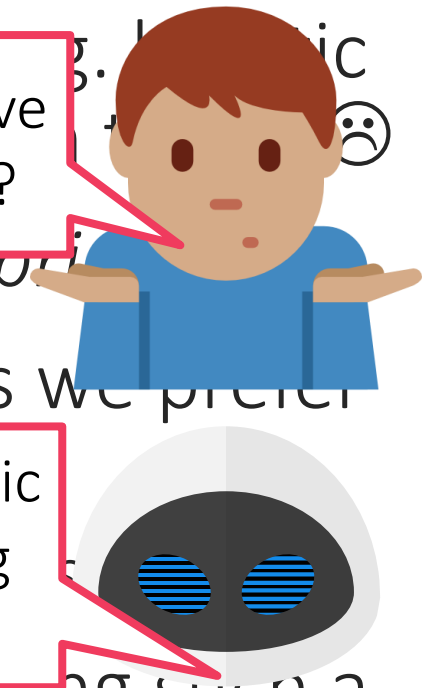
L2 regularization means w…

L1 regularization means w…

In the language of probability, the most direct way of specifying such a preference is by specifying a probability distribution itself

Prior: a probability distribution over all possible models

Just like we usually decide regularization before seeing any data, prior distribution also does not consider/condition on, any data



Can you Guess the Mean?

In this case we are said to have a *prior belief* or simply *prior*, on the models μ, in this case the uniform prior $\mathbf{UNIF}([0,2])$. This means that unless we see any data to make us believe otherwise, we will think $\mathbb{P}[\mu] = \begin{cases} 0.5 & \text{if } x \in [0,2] \\ 0 & \text{if } x \notin [0,2] \end{cases}$.

$x_1, x_2, \ldots, x_n \sim N(\mu^*, 1)$

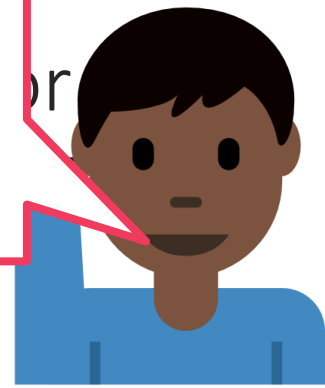
Can we estimate the "model" μ^* from these samples?

Likelihood function: for a c

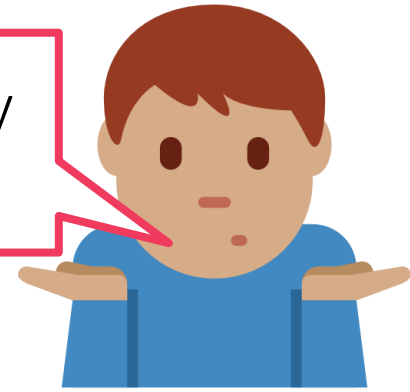
$\mathbb{P}[x_i \mid \mu, 1] = \frac{1}{\sqrt{2\pi}} \exp(-($

MLE: $\arg\max_{\mu \in \mathbb{R}} \prod_{i=1}^{n} \mathbb{P}[x_i \mid \mu, 1] = \arg\min_{\mu \in \mathbb{R}} \sum_{i=1}^{n} (x_i - \mu)^2$

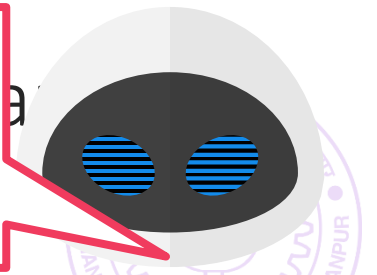
Suppose we believe (e.g
have been presented tha
could otherwise be any value within that interval)



What happens we do see some data, namely the actual samples from the distribution?



We use the samples and the rules of probability to update our beliefs about what μ can and cannot be. Let us see how to do this



Posterior

Before we see any data, we have a *prior* belief $\mathbb{P}[\mu]$ on the models

It tells us which models [illegible] en

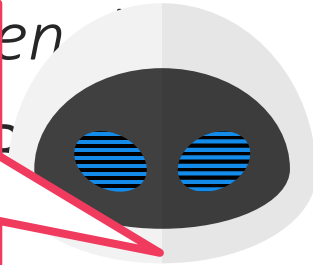
Then we see data $x_1, \ldots$ [illegible]

we want to find out $\mathbb{P}[\mu \mid x_1, \ldots, x_n]$

[illegible] has a name: [illegible]*ly* posterior

It tells us which models are more likely/less likely after we have seen data

Note that when we say $\mathbb{P}[\mu]$ or $\mathbb{P}[\mu \mid x_1, \ldots, x_n]$, we mean probability density and not probability mass since μ is a continuous r.v.



Bayes Rule

Samples are independent

Law of total probability

$\mathbb{P}[\mu] = \text{UNIF}([0,2])$

$$\mathbb{P}[\mu \mid x_1, \ldots, x_n] = \frac{\mathbb{P}[x_1, \ldots, x_n \mid \mu] \cdot \mathbb{P}[\mu]}{\mathbb{P}[x_1, \ldots, x_n]} = \frac{\mathbb{P}[\mu] \cdot \prod_{i=1}^{n} \mathbb{P}[x_i \mid \mu]}{\prod_{i=1}^{n} \mathbb{P}[x_i]}$$

$$= \frac{\mathbb{P}[\mu] \cdot \prod_{i=1}^{n} \mathbb{P}[x_i \mid \mu]}{\prod_{i=1}^{n} \int_{\mathbb{R}} \mathbb{P}[x_i \mid t] \cdot \mathbb{P}[t] \, dt} = \frac{0.5 \cdot \prod_{i=1}^{n} \mathbb{P}[x_i \mid \mu]}{\prod_{i=1}^{n} 0.5 \cdot \int_0^2 \mathbb{P}[x_i \mid t] \, dt} \text{ if } \mu \in [0,2]$$

else if $\mu \notin [0,2]$, *then* $\mathbb{P}[\mu \mid x_1, \ldots, x_n] = 0$

Maximum a Posteriori (MAP) Estimate

Just as MLE gave us the mode

us the model $\arg\max_{\mu \in \mathbb{R}} \mathbb{P}[\mu \mid x_1, \ldots, x_n, 1] = \arg\max_{\mu \in \mathbb{R}} \frac{\mathbb{P}[\mu] \cdot \prod_{i=1}^{n}$ [illegible] $\mu]}{\prod_{i=1}^{n} 0.5 \cdot \int_0^$ [illegible] $dt}$

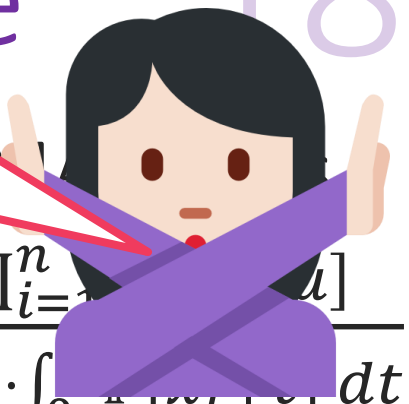
Thus, MAP returns the [illegible] one [illegible] we have seen some da

Note: *posterior probability (density) of some models may be larger t[illegible] prior probability (density) i.e. after seeing data these models seem more likely, for other models, it [illegible] data*

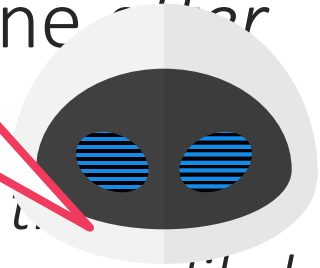
Note: *However, if pr[illegible] probability (density) [illegible] ut [illegible]*

Warning: *Do not read too much into these names* likelihood, prior, p[illegible] *All of them tell us how likely something is, given or not given something else*

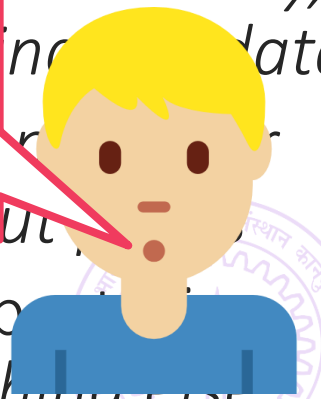
It is better to choose priors that do not completely exclude some models by giving them 0 probability (as we did)



True! Even in general, if your priors are bad, or too strong, then you may end up getting funny models as a result of doing MAP estimation



Indeed! For example if we were wrong and μ^* was actually not $\in [0,2]$ then not matter how many samples we see, we will never estimate μ^* correctly!!



MAP vs Regularization

$$\arg\max_{\mu\in\mathbb{R}} \frac{\mathbb{P}[x_1,\ldots,x_n \mid \mu]\cdot\mathbb{P}[\mu]}{\mathbb{P}[x_1,\ldots,x_n]} = \arg\max_{\mu\in\mathbb{R}} \mathbb{P}[x_1,\ldots,x_n \mid \mu]\cdot\mathbb{P}[\mu]$$

Taking negative log likelihoods on both sides $\mathbb{P}[\mu]\cdot\prod_{i=1}^{n}\mathbb{P}[x_i \mid \mu]$

$$\arg\min_{\mu\in\mathbb{R}} -\ln\mathbb{P}[\mu] + \frac{1}{2}\sum_{i=1}^{n}(x_i-\mu)^2$$

However, $\mathbb{P}[\mu]$ is constant for $\mu \in [0,2]$ and 0 otherwise ($\ln 0 \to \infty$)

$$\arg\min_{\mu\in\mathbb{R}} \sum_{i=1}^{n}(x_i-\mu)^2 \text{ s.t. } \mu \in [0,2]$$

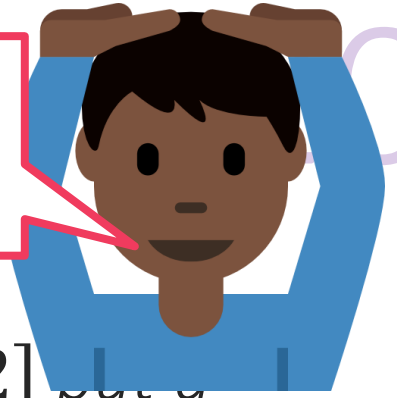
Thus, even MAP solutions can correspond to optimization problems!

In this case, what was the prior became a constraint

In general, the prior becomes a regularizer

MAP vs Regula

Similarly, had we used a Laplacian prior, we would have obtained L1 regularization instead

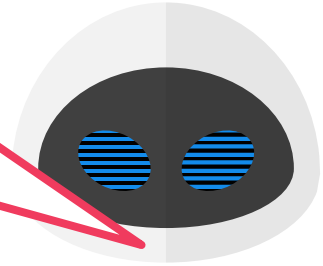


Consider the same problem as before but a different prior

This time we do not believe μ must have been in the interval $[0,2]$ but a much milder prior that μ is not too large

A good way to

The regularization constant is dictated by the strength of the regularization. Be careful not to have strong priors (uninformed strong opinions are bad in real life too ☺)



$$\text{MAP: } \arg\min_{\mu\in\mathbb{R}} -\ln \mathbb{P}[\mu] + \frac{1}{2}\sum_{i=1}^{n}(x_i-\mu)^2$$

$$= \arg\min_{\mu\in\mathbb{R}} \frac{\mu^2}{2\sigma^2} + \frac{1}{2}\sum_{i=1}^{n}(x_i-\mu)^2 = \arg\min_{\mu\in\mathbb{R}} \frac{\mu^2}{\sigma^2} + \sum_{i=1}^{n}(x_i-\mu)^2$$

Thus, a Gaussian prior gave us L2 regularization!

***Note**: σ effecitely dictates the regularization constant – not useless!!*

***Note**: this is basically ridge regression except in one dimension!!*

Random Vectors

Random vectors can be thought of as simply a collection of random variables arranged in an array $\mathbf{X} = [X_1, X_2, \ldots, X_d]^\top$

No restriction on the random variables being independent or uncorrelated

PMF/PDF of $\mathbf{X}$ is simply the joint PMF/PDF of $\{X_1, X_2, \ldots, X_d\}$

Can talk about marginal/conditional prob among $X_1, \ldots X_d$

Think of $X_1, X_2, \ldots, X_d$ as just a bunch of r.v.s

$\mathbb{P}[X_2, X_3 \mid X_1, X_4, X_5]$

Since PMF/PDF of $\mathbf{X}$ is simply a joint PMF/PDF, all probability laws we learnt earlier continue to hold if we apply them correctly

Chain Rule, Sum Rule, Product Rule, Bayes Rule

Conditional/marginal variants of all these rules

Random Vectors

Expectation of a random variable is simply another vector (of same dim) of the expectations of the individual random variables

$$\mathbb{E}\mathbf{X} = [\mathbb{E}X_1, \mathbb{E}X_2, \ldots \mathbb{E}X_d]^\top$$

Linearity of expectation continues to hold: if $\mathbf{X}, \mathbf{Y}$ any two vector r.v. (not necessarily independent, then $\mathbb{E}[\mathbf{X} + \mathbf{Y}] = \mathbb{E}\mathbf{X} + \mathbb{E}\mathbf{Y}$

Scaling Rule: If $c \in \mathbb{R}$ is a constant then $\mathbb{E}[c \cdot \mathbf{X}] = c \cdot \mathbb{E}\mathbf{X}$

Dot Product Rule: If $\mathbf{a} \in \mathbb{R}^d$ is a constant vector, then $\mathbb{E}[\mathbf{a}^\top \mathbf{X}] = \mathbf{a}^\top \mathbb{E}\mathbf{X}$

Proof: $\mathbb{E}[\mathbf{a}^\top \mathbf{X}] = \mathbb{E}\left[\sum_{i=1}^d a_i X_i\right] = \sum_{i=1}^d \mathbb{E}[a_i X_i] = \sum_{i=1}^d a_i \cdot \mathbb{E}[X_i] = \mathbf{a}^\top \mathbb{E}\mathbf{X}$

Matrix Product Rule: If $A \in \mathbb{R}^{n \times d}$ is a constant matrix then

$$\mathbb{E}[A\mathbf{X}] = A\mathbb{E}\mathbf{X}$$

Proof: Use Dot Product Rule n times

Random Vectors

Mode easy to define: $\arg\max_{X_1,\ldots,X_d} \mathbb{P}[X_1, \ldots, X_d]$

Median not easy to define – no unique definition

Definition 1: $\text{med}(\mathbf{X}) = [\text{med}(X_1), \text{med}(X_2), \ldots, \text{med}(X_d)]^\top$

Definition 2: *minimizer of absolute distance (in this case L1 norm)*

$$\text{med}(\mathbf{X}) = \arg\min_{\mathbf{v}\in\mathbb{R}^d} \mathbb{E}[\|\mathbf{X} - \mathbf{v}\|_2]$$

Note: even here we still have $\mathbb{E}[\mathbf{X}] = \arg\min_{\mathbf{v}\in\mathbb{R}^d} \mathbb{E}[\|\mathbf{X} - \mathbf{v}\|_2^2]$

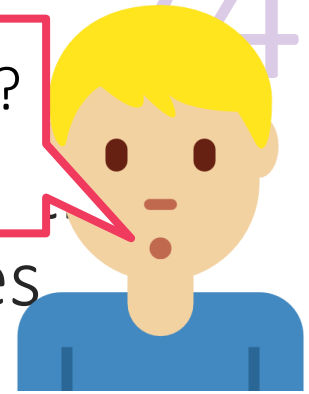
Proof: $\mathbb{E}[\|\mathbf{X} - \mathbf{v}\|_2^2] = \mathbb{E}[\|\mathbf{X}\|_2^2] + \mathbb{E}[\|\mathbf{v}\|_2^2] - 2 \cdot \mathbf{v}^\top \mathbb{E}[\mathbf{X}]$

Taking derivative w.r.t $\mathbf{v}$ *and using first order optimality does the trick*

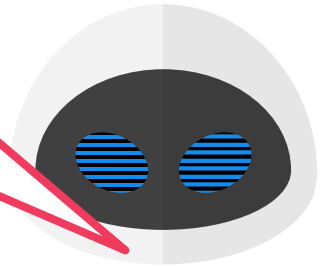
Random Vectors

Since random vectors are a bunch of r.v.s, to capture variance of this collection, need to have all pairwise covariances

$$\mathbf{Cov}(\mathbf{X}) = \begin{bmatrix} \mathbb{V}X_1 & \mathrm{Cov}(X_1, X_2) & \dots & \mathrm{Cov}(X_1, X_d) \\ \mathrm{Cov}(X_2, X_1) & & & \\ \vdots & & & \\ \mathrm{Cov}(X_d, X_1) & \mathrm{Cov}(X_d, X_2) & \dots & \mathbb{V}X_d \end{bmatrix}$$



If $\mathbf{X}$ is a vector, isn't $\mathbf{X}\mathbf{X}^\top$ a matrix? What does $\mathbb{E}[\mathbf{X}\mathbf{X}^\top]$ even mean?



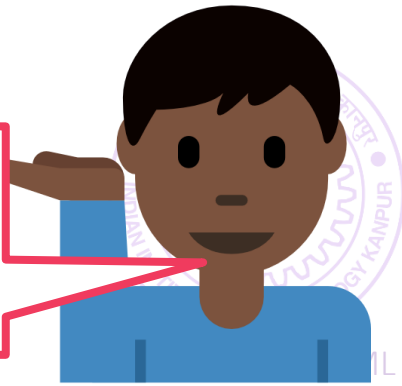
Just as a random vector is a collection of random variables arranged as a 1D array, a random matrix is a collection of r.v.s arranged as a 2D array!

Another cute formula

$$\mathbf{Cov}(\mathbf{X}) = \mathbb{E}[(\mathbf{X} - \boldsymbol{\mu})(\mathbf{X} - \boldsymbol{\mu})^\top] = \mathbb{E}[\mathbf{X}\mathbf{X}^\top] - \boldsymbol{\mu}\boldsymbol{\mu}^\top, \text{ where } \boldsymbol{\mu} = \mathbb{E}\mathbf{X}$$

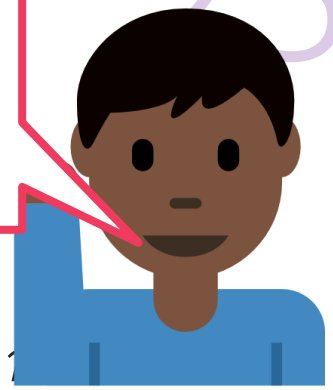
$$\mathbf{Cov}(c \cdot \mathbf{X}) = c^2 \cdot \mathbf{Cov}(\mathbf{X})$$

Note that (i, j)-th entry of matrix $(\mathbf{X} - \boldsymbol{\mu})(\mathbf{X} - \boldsymbol{\mu})^\top$ is $(X_i - \mu_i)(X_j - \mu_j)$. Thus, (i, j)-th entry of $\mathbb{E}[(\mathbf{X} - \boldsymbol{\mu})(\mathbf{X} - \boldsymbol{\mu})^\top]$ is $\mathbb{E}[(X_i - \mu_i)(X_j - \mu_j)] = \mathrm{Cov}(X_i, X_j)$



Useful Operations on V

Can you prove that the covariance matrix of any random vector is always a PSD matrix?



If $\mathbf{X} \in \mathbb{R}^m, \mathbf{Y} \in \mathbb{R}^n$ are two random
independent), then

$$\mathbf{Cov}(\mathbf{X}, \mathbf{Y}) = \mathbb{E}[(\mathbf{X} - \boldsymbol{\mu}_\mathbf{X})(\mathbf{Y} - \boldsymbol{\mu}_\mathbf{Y})^\top] = \mathbb{E}[\mathbf{X}\mathbf{Y}^\top] - \boldsymbol{\mu}_\mathbf{X}\boldsymbol{\mu}_\mathbf{Y}^\top \in \mathbb{R}$$

where $\boldsymbol{\mu}_\mathbf{X} = \mathbb{E}\mathbf{X}$ and $\boldsymbol{\mu}_\mathbf{Y} = \mathbb{E}\mathbf{Y}$, $\mathbf{Cov}(\mathbf{X}, \mathbf{Y})$

Dot Product Rule: If $\mathbf{a} \in \mathbb{R}^d$ is a constant vector, then $\mathbb{V}[\mathbf{a}^\top\mathbf{X}] = \mathbf{a}^\top\mathbf{Cov}[\mathbf{X}]\mathbf{a}$

Proof: $\mathbb{V}[\mathbf{a}^\top\mathbf{X}] = \mathbb{E}[(\mathbf{a}^\top\mathbf{X})^2] - (\mathbf{a}^\top\boldsymbol{\mu}_\mathbf{X})^2 = \mathbb{E}[\mathbf{a}^\top\mathbf{X}\mathbf{X}^\top\mathbf{a}] - \mathbf{a}^\top\boldsymbol{\mu}_\mathbf{X}\boldsymbol{\mu}_\mathbf{X}^\top\mathbf{a}$

$= \mathbf{a}^\top\mathbb{E}[\mathbf{X}\mathbf{X}^\top]\mathbf{a} - \mathbf{a}^\top\boldsymbol{\mu}_\mathbf{X}\boldsymbol{\mu}_\mathbf{X}^\top\mathbf{a} = \mathbf{a}^\top\left(\mathbb{E}[\mathbf{X}\mathbf{X}^\top] - \boldsymbol{\mu}_\mathbf{X}\boldsymbol{\mu}_\mathbf{X}^\top\right)\mathbf{a} = \mathbf{a}^\top\mathbf{Cov}[\mathbf{X}]\mathbf{a}$

Matrix Product Rule: If $A \in \mathbb{R}^{n\times d}$ is a constant matrix then

$$\mathbf{Cov}[A\mathbf{X}] = A\mathbf{Cov}[\mathbf{X}]A^\top \in \mathbb{R}^{n\times n}$$

***Proof**: Try arguing similarly as the dot product rule*

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*

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

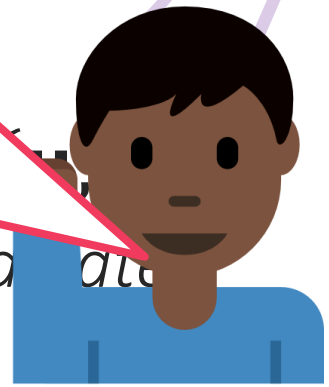
However, $\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x_i^2\right)$ is simply $\mathcal{N}(0,1)$ i.e. we indeed have

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

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

Gaus

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 a more detailed analysis to show that these are actually Gaussian.



Given a

Every

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 $j \in [d]$

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

Given values $\mathbf{x}_k = v_k$ *for all other coordinates* $k \neq j$, $\mathbf{x}_j$ *is still Gaussian*

Expression a bit complicated – refer to DFO Sec 6.5.1 (see the reference section on the course webpage)

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)

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

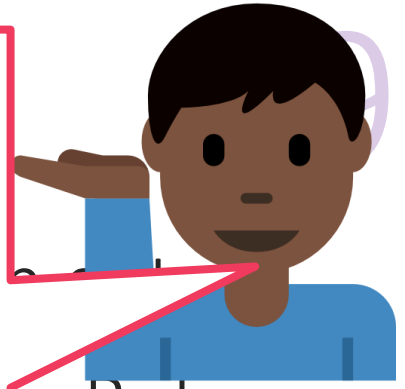
Properties of (univariate) Gaussian tells us that this is same as saying

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

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

You might be wondering why conditioned as $\mathbb{P}[y^i \mid \mathbf{x}^i, \mathbf{w}]$ and not $\mathbb{P}[y^i, \mathbf{x}^i \mid \mathbf{w}]$. This is because we are currently assuming that features $\mathbf{x}^i$ do not depend on the model $\mathbf{w}$. Thus, the chain rule gives us $\mathbb{P}[y^i, \mathbf{x}^i \mid \mathbf{w}] = \mathbb{P}[y^i \mid \mathbf{x}^i, \mathbf{w}] \cdot \mathbb{P}[\mathbf{x}^i \mid \mathbf{w}]$ and $\mathbb{P}[\mathbf{x}^i \mid \mathbf{w}]$ is just $\mathbb{P}[\mathbf{x}^i]$ which does not depend on the model $\mathbf{w}$. Note that we also assume in calculations that $y^i, \mathbf{x}^i$ are independent of $y^j, \mathbf{x}^j$ for $i \neq j$



es Rule

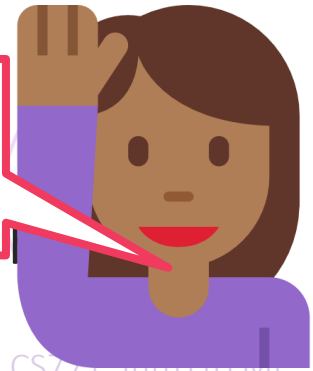
$$\mathbb{P}[\mathbf{w} \mid y^1, \ldots, y^n, \mathbf{x}^1, \ldots, \mathbf{x}^n] = \frac{\mathbb{P}[y^1,\ldots,y^n \mid \mathbf{x}^1,\ldots,\mathbf{x}^n,\mathbf{w}] \cdot \mathbb{P}[\mathbf{w}]}{\mathbb{P}[y^1,\ldots,y^n,\mathbf{x}^1,\ldots,\mathbf{x}^n]}$$

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

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

$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} \ln \mathbb{P}[\mathbf{w}] + \sum_{i=1}^{n} \ln \mathbb{P}[y^i \mid \mathbf{x}^i, \mathbf{w}]$

We will soon study "generative" models where the features $\mathbf{x}^i$ themselves would become random variables dependent on a (more complicated) model

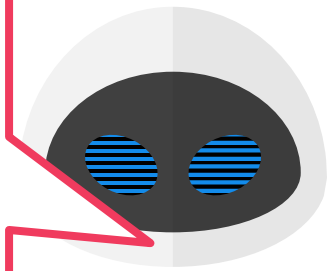


and posterior are distributions over models i.e. over $\mathbb{R}^d$

MAP for Probabilistic Regression

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

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



Be careful, there are two variance terms here σ_l, σ_p

The above is $\arg\min_{\mathbf{w}\in\mathbb{R}^d} \left(\frac{\sigma_l}{\sigma_p}\right)^2 \cdot \|\mathbf{w}\|_2^2 + \sum_{i=1}^{n}\left(y^i - \mathbf{w}^\top \mathbf{x}^i\right)^2$ *i.e. ridge regression*

Thus, σ_l, σ_p *together decide the regularization constant*

There is a multivariate version of the Laplace distribution too!

Using it as a prior with $\boldsymbol{\mu} = \mathbf{0}$ *(the zero vector) will give us LASSO!*

***Warning**: expression for the Laplace PDF for general covariance is a bit tricky)*

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

Bayesian Learning

Before we started doing probabilistic ML, we used to 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, we 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 need models to predict labels, 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*

***Note**: 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