Methods of Data Analysis
Probabilistic models and inference

Week 3

1 Motivation

One of the basic tasks in data analysis consists of building parametric models for the observed phenomena. Mathematically, these models can take on a variety of forms: they can be “stylized” models that capture just the essence of the observations (e.g., scaling laws, simple agent-based models like the “Game of Life” etc.) or precise mathematical models that distill “laws of nature” as would have been observed with ideal experimental instrumentation (e.g., Newton’s laws of motion or Maxwell’s equations for electromagnetic fields, written down as differential equations). Many processes that we wish to describe are, however, probabilistic, and the models that we build them should reflect this probabilistic nature. Furthermore, we may desire a probabilistic description also because that is a good effective description of the process—while the underlying reality may even be deterministic (sometimes we simply don’t know!), the model we build will likely coarse-grain across some microscopic events due to our ignorance, and, as a result, will necessarily have a statistical character. Lastly, it is important to realize that often the process of measurement itself, which provides the data for our model-fitting exercise, is stochastic. This may be due to physical measurement errors (whose magnitude can often be determined by separate calibration measurement), due to random, and sometimes even biased, event sampling (e.g., not all particles might be detected in a particle detector, or not all mRNA molecules counted in an RNASeq experiment), or due to other causes. For any combination of these three reasons, the actual data that we observe can be seen as a random draw from some probability distribution. The aim of probabilistic modeling is to come up with a functional form (a parametrization) of such a distribution, and infer the parameters.

Even within the class of probabilistic models one can distinguish various approaches. One approach is to look for as compact a description as possible, in terms of the number of parameters to be inferred. In this case, we are less concerned with the interpretation of the parameters themselves; the measure of modeling success is either some intrinsic notion of the simplicity (or low dimensionality) of the model, and/or its predictive power; for example, recent advances in deep learning consider models with many thousands or even millions of free parameters which often lack direct interpretation. If the complexity of the model is a central parameter of the model-building approach that can be easily tuned, we can talk about dimensionality reduction approaches (e.g., Principal Component Analysis, or Gaussian mixtures modeling). As we will see, the tradeoff between model complexity and its ability to precisely fit the data can be put into a formal framework; in one commonly practiced version of this framework the tradeoff is navigated by asking how well the model generalizes to test data that hasn’t been used in the model’s construction.
Another probabilistic modeling approach is based on prior knowledge of the modeled processes. Here, the parameters and the form of the model are chosen to explicitly represent and parametrize the reality of the events that lead to the observed data—including events intrinsic to the system of study as well as the events that constitute the process of measurement or data acquisition. While these models may sometimes underperform the statistical modeling (but may also outperform it if the prior knowledge is sufficiently structured and predictive), the advantage is in that the parameters are interpretable. As a consequence, such models may suggest new experiments in which the parameter values can be changed, and for which the model makes unambiguous new predictions.

Whatsoever the approach taken, probabilistic modeling consists of: (i) postulating a parametric model for the probability of observing some data given the parameter values; (ii) inferring the parameters of the model from data; (iii) checking how well the model (with the inferred parameters) predicts new phenomena, or at least how well it performs on a repeat of the same experiment; (iv) sometimes also distinguishing formally between different models of varying complexity to pick the most predictive model (known also as “model selection”). How inference is done in practice depends very much in the tradition of the field, and on the kind of data (e.g. how many measurements are available compared to the number of parameters to be learned) and prior knowledge that is available about the problem (e.g., do laws of physics / theory put constraints on the parameters or the properties of the measurement process). Even within a discipline, some subfields might use frequentist approaches and statistics (e.g., particle physics), while Bayesian inference is accepted in other fields (e.g., cosmology).

2 Goals

- Introduce the basic concepts and the terminology of statistical inference.
- Explain and demonstrate overfitting, regularization, and cross-validation.
- Illustrate the connection between the traditional $\chi^2$ parameter fitting, maximum likelihood inference, and maximum a posteriori inference. Discussion of choice of priors.
- Introduce generalized linear models (GLMs), a versatile class of probabilistic models. Illustrate inference on neural data.

3 Data

The data are neural recordings of retinal ganglion cells from the lab of Ronen Segev at Ben-Gurion University, Israel; they have been used as part of analysis in Ref [4]. The stimulus is a sequence of spatially uniform light intensity frames displayed on the screen every $\Delta = 33.33$ ms, and is stored in a vector called MDA3_stim.mat, so that, e.g., frame number 1 is displayed from $t = 0$ to $t = 33.33$ ms. Spikes of 23 neurons are stored in MDA3_spikes.mat. For every neuron you will get a list of spike times expressed in seconds.

4 Maximum likelihood and Bayesian inference

Suppose one is able to write down a probabilistic model for a physical process that depends on a vector of parameters $\theta$ (which we will ultimately want to infer from data). A probabilistic model means that given the parameters, the model predicts the probability of observing an
experimental outcome \( \vec{y} \), that is, \( P(\vec{y}|\theta) \). Such models are also called “generative models,” because one can generate new simulated data by drawing from \( P \). The outcome can be a single value, a vector of values, a timeseries, or any other object, so long as one can define and tractably manipulate probability distributions over such class of objects.

Note that in many cases such a model is not easy to arrive at a priori: it is supposed to fully account for the probability distribution \( P \), which should capture intrinsic stochasticity in the physical process of interest, as well as any noise in the measurements that were performed to obtain the \( \vec{y} \)'s. For example, if we observe emissions of \( \gamma \) rays from a radioactive decay and measure the time intervals between successive \( \gamma \) ray arrivals at the detector with finite precision, then the probability of recording a particular interval is the probability that a certain interval happened (which is exponentially distributed in time), convolved with the finite precision of the recording apparatus (e.g., a gaussian error on time measurement). Often, the processes that we observe are not simple enough for us to know their probability distribution, or the measurement process is not characterized well enough to know its error model (this is common in biology, for example). There exist inference methods to deal with such situations, but they are beyond the scope of this lecture: here, we will only deal with examples where a generative model can be written down.

### 4.1 Maximum likelihood and Cramer-Rao bound.

Let the data from which we will infer the model consist of a collection of such experimental outcomes, or measurement samples, denoted by \( D \). Commonly, the data \( D \) is a series of \( N \) independent measurements, so that the total probability of a particular data vector is a product of probabilities for individual measurements, \( P(\vec{y}|\theta) = \prod_i P(y_i|\theta) \).

A basic way to infer model parameters is to do maximum likelihood inference. This procedure consists of viewing the conditional probability \( P \) as a function of model parameters, and thus define the (log)-likelihood as \( \mathcal{L}(\theta) = \log \prod_i P(y_i|\theta) = \sum_{i=1}^N \log P(y_i|\theta) \). It is convenient to deal with logs, as in that case the log-likelihood is extensive (linear) in the number of samples; often, it is normalized to per-sample log-likelihood. Inference then requires us to look for parameters that would maximize the likelihood of observing the data that we measured:

\[
\theta^* = \arg\max_\theta \mathcal{L}(\theta). \tag{1}
\]

If one can write down the likelihood, then the best parameters can be found by numerical optimization, e.g., gradient ascent or simulated annealing (it is common for the log-likelihood to be a non-trivial function of the parameters with multiple maxima).

Finding the parameters that maximize the log-likelihood doesn’t tell us anything directly about how constrained the parameters are by the data: would varying a particular parameter away from its optimum change the fit significantly? In the neighborhood of the extremum we can expand the log-likelihood in Taylor series:

\[
\mathcal{L}(\theta) = \mathcal{L}_0(\theta^*) + \frac{1}{2} \sum_{a,b} \frac{\partial^2 \mathcal{L}}{\partial \theta_a \partial \theta_b} \bigg|_{\theta^*} (\theta - \theta^*_a)(\theta - \theta^*_b), \tag{2}
\]

where \( \theta^*_a,b \) are the components of the optimal vector of parameters, \( \theta^* \); the linear expansion term in \( \theta \) is zero, because we assumed \( \theta^* \) is an extremum. Remember that Eq. (2) is still a function of the particular data observed, \( D \). We can average the second derivative to obtain the “Fisher information”, an important quantity in inference (despite the name, “Fisher information” is not
an information-theoretic quantity):

$$\mathcal{I}_{ab} = -\left\langle \frac{\partial^2 \mathcal{L}(\theta)}{\partial \theta_a \partial \theta_b} \right\rangle = \left\langle \frac{\partial \mathcal{L}}{\partial \theta_a} \frac{\partial \mathcal{L}}{\partial \theta_b} \right\rangle,$$

(3)

where the last equality follows directly from the first. Angular braces denote an average over the generating distribution, $P(\mathcal{D}|\theta)$; this is sometimes replaced by an empirical average over the data points.

Intuitively, Fisher information is the curvature of the log-likelihood function. Formally, it puts bounds on how well parameters $\theta$ can be estimated from data. Suppose we consider an estimator, $\hat{\theta}_a(\mathcal{D})$ for the parameter $\theta_a$; note that this notation explicitly emphasizes that estimators are explicit functions of the observed data. Let this be an unbiased estimator, i.e.,

$$\left\langle \hat{\theta}_a(\mathcal{D}) - \theta_a^* \right\rangle_{P(\mathcal{D}|\theta^*)} = 0,$$

(4)

where $\theta_a^*$ is the true value of the parameter and the expectations are taken over the distribution that generated the data (so in this sense “unbiased estimator” means that when averaged over many draws of the data, the estimator will produce the true value of the underlying parameter). For unbiased estimators, Fisher information tells us how precisely the parameters can be constrained using our estimators given the data, by the Cramer-Rao bound:

$$\text{Cov}_{\hat{\theta}} = \left\langle (\hat{\theta} - \theta^*)(\hat{\theta} - \theta^*)^T \right\rangle_{P(\mathcal{D}|\theta^*)} \geq \mathcal{I}^{-1},$$

(5)

that is, the covariance on the estimators is greater or equal to the inverse of the Fisher matrix. Specifically, for each estimator, this gives us the expected error bar given the data, as the corresponding (square root of the) diagonal value of the inverse Fisher matrix. The estimators achieving equality are called efficient, and they provide the lowest mean-square-error among all unbiased estimators.

### 4.2 Bayesian inference

A generalization of maximum likelihood inference is Bayesian inference. Here we first use the Bayes’ rule to invert the likelihood and write

$$P(\theta|\mathcal{D}) = \frac{P(\mathcal{D}|\theta)P(\theta)}{P(\mathcal{D})} \propto P(\mathcal{D}|\theta)P(\theta)$$

(6)

The quantity at left is the posterior, the degree of belief (or probability) that the parameter $\theta$ takes on a certain value, given the measured data $\mathcal{D}$. Bayes rule tells us that this is proportional to the likelihood and the prior $P(\theta)$, the distribution of values over $\theta$ that we consider likely before we made measurements that yielded $\mathcal{D}$. Why would the inclusion of the prior distribution be important? Consider the case where the likelihood formally permits a value for some parameter $\theta$ across the whole real range, but we know—from theory, previous experiments, or some other source—that $\theta \in [\theta_0, \theta_1]$; the most simple example would be that we have prior knowledge that $\theta > 0$. This knowledge can be formally encoded into the prior, by e.g., assigning prior probability of 0 to any value of $\theta$ outside the specified range.

A more non-trivial but attractive feature of the Bayesian approach is also that it can be used to formalize the concept of sequentially updating the knowledge of the parameters as the new data come in, by combining existing estimates and the new data in an optimal way. In this case, one can write the posterior $P(\theta|\{y_{t+1}, y_t, \ldots , y_1\})$ as a function of the prior, $P(\theta|\{y_t, \ldots , y_1\})$, and seeing how the distribution over $\theta$ (or any of its moments) change as more and more data is
collected and the inference proceeds. Numerically interesting is the case where the distribution for the prior is chosen so that after the update (via the likelihood) the posterior has the same distribution of the prior; such priors are known as **conjugated** priors to a given likelihood function. The simplest case is perhaps one in which we imagine the inference for the mean of a normal distribution of known variance, \( \sigma^2 \), and assume that the prior for the mean is Gaussian as well, with mean \( \mu_0 \) and variance \( \sigma_0^2 \). In equations,

\[
P(\mu) = \frac{1}{\mathcal{Z}_0} e^{-\frac{1}{2} \frac{(\mu - \mu_0)^2}{\sigma_0^2}} \tag{7}
\]

\[
P(y|\mu) = \frac{1}{\mathcal{Z}_0} e^{-\frac{1}{2} \frac{(y - \mu)^2}{\sigma^2}} \tag{8}
\]

where we think of \( y \) as a data sample that we observed. With these choices and a little algebra, one can write out the posterior as

\[
P(\mu|y) = \frac{1}{\mathcal{Z}_1} \exp \left( -\frac{1}{2 \sigma^2} \left( \frac{1}{\sigma^2} + \frac{1}{\sigma_0^2} \right) \mu^2 + \mu \left( \frac{x}{\sigma^2} + \frac{\mu_0}{\sigma_0^2} \right) \right). \tag{9}
\]

Since Eq. (9) is quadratic in the exponent, it is a Gaussian again, with an updated mean and variance; in short, both the prior and posterior are Gaussian. This means that the sequential process of updating the prior with new observations reduces to a process of updating the means and variances as we observe more and more samples. Suppose we have observed \( t \) samples already which provided us with the Gaussian posterior with mean \( \mu_t \) and variance \( \sigma_t^2 \). Upon a new observation \( y_{t+1} \) we can read off from Eq. (9) how the Gaussian parameters need to be updated:

\[
\frac{1}{\sigma_{t+1}^2} = \frac{1}{\sigma_t^2} + \frac{1}{\sigma^2} \tag{10}
\]

\[
\mu_{t+1} = \frac{y_{t+1} / \sigma^2 + \mu_t / \sigma^2}{1/\sigma_t^2 + 1/\sigma^2}. \tag{11}
\]

Note the intuitive features of this result: (i) the new data and the previous estimate of the mean are weighted by their variances, as expected; (ii) as new samples are collected, the variance of the posterior shrinks; specifically, (iii) as \( t \to \infty \), the prior is “forgotten” and

\[
\sigma_t^2 \to \frac{\sigma^2}{t} \tag{12}
\]

\[
\mu_{t+1} = \frac{t}{t+1} \mu_t + \frac{1}{t+1} y_{t+1}. \tag{13}
\]

Note that the last recursion is satisfied by \( \mu_t = \frac{1}{t} \sum_{t'=1}^{t} y_{t'} \). So in the limit of a large number of samples, the Bayesian estimator for the mean approaches the ML estimator, which is just the empirical average over the samples. The variance of the Bayesian estimator is also the standard error of the mean, i.e., the standard deviation from the generating distribution divided by the square root of the number of samples.

Choosing a “correct” prior is often, and depending on the field, a controversial topic, which is beyond the scope of this lecture. A simple way to imagine what the priors are doing is to think in the limit where our data are very limited, perhaps consisting of one measurement only. Then the posterior is dominated about our prior guess about what the parameters should be, but the problem remains well posed and, as we shall see, we do not overfit, as can happen
with maximum likelihood inference. As more data is accumulated, the particular choice of prior matters less and less, until in the data-dominated regime the role of the prior disappears (unless the prior assigns probability of exactly 0 to some parameters).

One possible choice of priors is, as above, guided by computational convenience, if we suspect that normal applications will put us in the data-dominated regime. If the prior chosen is conjugate to the likelihood, probabilistic updates (where one needs to keep track of the full distribution) reduce to updates of the sufficient statistics of the distribution (e.g., mean and variance), which can often be done analytically and very quickly. In particular, we may often think of any likelihood function that, in the vicinity of likelihood-maximizing parameters, can be approximated by a Gaussian (as we have done when discussing the Cramer-Rao bound) and which will, assuming Gaussian prior, yield Gaussian parameter updates, as in Eq. (11).

There has been a lot of discussion on how to choose “uninformative priors” that shouldn’t bias the inference, or alternatively, how to choose priors such that some statistic of the posterior of interest is as unbiased as possible (e.g., Nemenman-Shaffee-Bialek estimator for the entropy). Regarding “uninformative” prior, the notion of a flat (i.e., uniform) prior is often floated, but is not without difficulty: for instance, one cannot define a uniform prior across infinite parameter ranges since such prior would be non-normalizable; furthermore, even if one sets up a uniform prior over a range, by reparametrizing the variable \( \theta \) into some \( f(\theta) \), one has to change the corresponding prior over \( f \) into something non-uniform by the law of transformation of probabilities. This is partially taken into account by Jeffreys prior, where

\[
P(\theta) \propto \sqrt{\det \mathcal{I}(\theta)},
\]

which is invariant to reparametrization.

The Bayes posterior contains all information about the parameter values that can be extracted from the data and the prior. But often we need single “best” estimates for the parameters, not the complete joint distribution. What constitutes the “best” estimate is, however, dependent on the task at hand. Formally, we should specify a risk (or utility) function \( R(\hat{\theta}(D), \theta) \), and choose the Bayes estimator \( \hat{\theta}_B \) such that it minimizes the expected risk in the posterior distribution:

\[
\hat{\theta}_B(D) = \arg\min_{\theta} \langle R(\hat{\theta}(D), \theta) \rangle_{P(\theta|D)}.
\]

For instance, if the risk function is minimum squared error (MSE), \( R = (\hat{\theta}(D) - \theta)^2 \), then the Bayesian estimator of \( \theta \) is the posterior mean:

\[
\hat{\theta}_B(D) = \int d\theta \, \theta P(\theta|D);
\]

various other picks for the risk function will yield alternative Bayesian estimators.

Bayesian estimators under MSE risk are asymptotically unbiased and efficient, as are maximum likelihood estimators.

Another common choice is also to pick the \( \theta \) that maximizes the posterior, which is equal to the maximization of likelihood when the prior distribution is flat; in this case the inference is known as MAP (maximum a posteriori) estimation. Note that MAP, although widely used, is strictly not a Bayesian estimate for continuous parameters (but one can form a Bayes estimate that converges to MAP estimate for unimodal distributions).

In addition to point estimates for best parameters, the posterior contains the full information about our knowledge of the parameters, including the statistical uncertainty over their values. Suppose we are interested in the covariance matrix of our estimates. In that case, if we can
compute the posterior, we can always compute the covariance directly (without reference to the Cramer-Rao bound or the estimators):

$$\text{Cov}_{\theta} = \int d\theta (\theta - \langle \theta \rangle)(\theta - \langle \theta \rangle)^T P(\theta|\mathcal{D}).$$  \hspace{1cm} (17)

5 \(\chi^2\) parameter fitting as inference, overfitting and regularization

Consider the problem of nonlinear data fitting. We are given measurement pairs \((x_t, y_t)\), \(t = 1, \ldots, T\), where \(T\) is the number of samples in the dataset \(\mathcal{D}\). Suppose we have a theoretical model that predicts the values of \(y\) given \(x\), and which depends on parameters \(\theta\), i.e., \(\hat{y} = f(x, \theta)\), where \(\hat{y}\) is the prediction. We would like to fit the best choice of \(\theta\) so as to minimize the squared error of the prediction. We define \(\chi^2\) (the goodness of fit) correspondingly:

$$\chi^2(\theta) = \frac{1}{T} \sum_{t=1}^{T} (y_t - f(x_t, \theta))^2,$$  \hspace{1cm} (18)

where we normalized the \(\chi^2\) per datapoint. How does this fitting problem relate to probabilistic inference?

We can easily show that if \(y\) have IID Gaussian errors of fixed variance, the maximum likelihood inference and the least-squares fitting are equivalent. Gaussian errors of fixed variance, \(\sigma^2\), mean that the probability of the observations given the parameters is:

$$P(\vec{y}|\theta) = \prod_t P(y_t|\theta) = \prod_t (2\pi\sigma^2)^{-1/2} \exp\left\{ -\left(\frac{y_t - f(x_t, \theta)}{\sigma^2}\right)^2 \right\}.$$  \hspace{1cm} (19)

Consequently, it is trivial to check that the log-likelihood is \(\mathcal{L}(\theta) = -\frac{1}{2}\sigma^2 \chi^2(\theta) + \text{const}\), where the constant does not depend on the parameters, \(\theta\). So in this case minimizing the \(\chi^2\), as in usual fitting, is equal to maximizing the log-likelihood, as in likelihood inference. This also means that if errors deviate from Gaussian IID assumptions (e.g., errors have extreme outliers or are correlated), normal fitting will produce suboptimal results – in that case, one either switches to likelihood inference, or uses robust fitting methods.

What happens when the number of parameters grows to become comparable to the number of (independent) samples in the data? Usually, maximum likelihood or parameter fitting will tend to overfit the data, i.e., select the parameters such that the objective function over the fitting (training) data achieves an excellent value, but were one to evaluate the objective function over the data not used for model fitting, the performance of the inferred model would drop substantially; the proverbial example is of (over)fitting polynomials to the smooth data corrupted by noise.

To guard against overfitting, one can use regularization together with cross-validation. The main idea is to trade off optimization of the objective function \((\chi^2\) or \(\mathcal{L}\)) against the penalty for making the parameters too big (or using too many of them). For instance, we could introduce

$$\chi'^2(\theta, \lambda) = \chi^2(\theta) + \lambda||\theta||^2.$$  \hspace{1cm} (20)

In the new objective function, \(\chi'^2\), we penalize parameter values that are too large (in L2 norm), and the tradeoff between goodness-of-fit and the penalization is given by the parameter \(\lambda\). How should one choose this parameter? The answer is by cross-validation: split the dataset many
times into the separate “training set” and the “testing set”. Minimize $\chi^2(\lambda)$ over the training set for different choices of $\lambda$, to get a family of optimized solutions, $\theta^*(\lambda)$. For every $\lambda$, use the set of parameters to evaluate the objective function over the testing set, and lastly choose such $\lambda^*$ for which the objective function on the testing set takes on the best value. This selects for a level of regularization that best performs on the data that was withheld during model fitting, i.e., for parameters that best generalize over unseen data, which is what a good model should do.

A common procedure to perform splitting the data into testing and training sets is called *K-fold crossvalidation*. Here, the data is split into $K$ blocks, and the model using a particular choice of $\lambda$ is fit $K$ times on $K-1$ blocks with one block always withheld for testing. The testing performance is evaluated as the mean performance at a given $\lambda$ over $K$ withheld blocks, and we choose $\lambda^*$ that maximizes such testing performance. One should always be mindful about how the splits are made in case we suspect the data is not IID (i.e., not stationary or is correlated in time).

Is regularization related to optimal inference in any principled way? That turns out to be the case! Suppose we do MAP inference with a prior, and let’s for simplicity choose a Gaussian prior over parameters, $P(\theta) \propto \exp\{-||\theta||^2/(2\sigma^2_P)\}$. By taking a logarithm of the Bayes rule, Eq (6), we find for the log posterior

$$\log P(\theta|D) = \mathcal{L}(\theta) + \log P(\theta)$$

which for the case of Gaussian IID errors and Gaussian prior results in:

$$\log P(\theta|D) = -\frac{1}{2\sigma^2} \chi^2(\theta) - \frac{1}{2\sigma^2_P} ||\theta||^2 \propto -\chi^2(\theta).$$

(22)

That is, maximum a posteriori inference with Gaussian priors is equal to an ad hoc regularization with L2 norm! Note that in the example above the data (in $\chi^2$) and the prior contribute in proportion to their reliability (or in inverse proportion to their variances), as could be expected. Bayesian inference is the most general way to state the problem of inference, making it explicitly clear about how data and our prior assumptions combine to affect the inferred parameter values. There are many possible priors that can be chosen either for their computational convenience (e.g., conjugate priors), or because of good generalization performance (sparse, or L1, priors); what prior distribution pick when and what should be the parameters of the priors is beyond the scope of our exercise.

In sum, the usual nonlinear fitting is a special instance of maximum likelihood inference, and regularized fitting is a special instance of Bayesian / MAP inference. Bayesian inference is an optimal way of combining data (sequentially or in a batch mode) with the prior beliefs about the parameters, but—because it is dealing with full probability distributions—it might be expensive to carry out unless assumptions about the distributions are made (e.g., taking Gaussians).

### 6 Basic model selection example

As an example we consider here the problem of fitting polynomials to data in the probabilistic framework. The point is to illustrate, in the simplest case, the tradeoff between model complexity and goodness of fit. Above, we considered a given model with possibly a lot of parameters, and applied regularization to keep overfitting in check. Here we consider not a single model, but a class of models (polynomials of varying degrees) and write:

$$f_d(x) = \sum_{k=0}^{K} a_k x^k,$$

(23)
where $x$ is the data point, $\vec{a}$ are $K+1$ coefficients of the fitting polynomial, and $f$ provides a model for the measurements $y$. For different $K$, polynomials constitute a nested set of models (since all polynomials of degree $K$ contain polynomials of a lower degree). Instead of fitting a model using large $K$ and regularization, we can now think of successively fitting higher and higher degree polynomials and comparing their performance. If we did that using the crossvalidation method described above, we could empirically find the $K^*$ that maximized the performance on test data, and would choose that value. Here, alternatively, we will make some analytical progress to illustrate where the tradeoff between fitting and model complexity originates from; in the process, we will also derive what is known as BIC (Bayes Information Criterion), a simple way to do model selection without cross-validation.

To proceed, let’s further assume that the likelihood is Gaussian, i.e., that the errors on data points are equal and normally distributed with variance $\sigma^2$. Then, the likelihood of our model is

$$P(y_t|\vec{a}, K) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(y_t-f_a(x_t))^2}.$$  \hspace{1cm} (24)

This is a joint likelihood of the model parameters, $\vec{a}$, and the model class (here given by polynomial degree, $K$). Let us now ask about the likelihood of the model itself, assuming we observe $T$ IID samples, $D = \{y_t\}$. We obtain this by marginalizing the likelihood over the parameter values $\vec{a}$:

$$P(D|K) = \int d^K\vec{a} \prod_{t=1}^T P(y_t|\vec{a}, K)$$

$$= \int d^K\vec{a} (2\pi\sigma^2)^{-T/2} \exp\left(-\frac{1}{2\sigma^2} \sum_t (y_t - f_a(x_t))^2\right)$$

$$= \int d^K\vec{a} (2\pi\sigma^2)^{-T/2} \exp\left(-TF(\vec{a})\right),$$

where $F(\vec{a}) = \frac{1}{2\sigma^2}((y_t-f_a(x_t))^2)_t$. In the limit of large number of samples ($T \to \infty$), the integral can be approximated using the saddle point method, i.e., it is dominated by $\vec{a}^*$ that solves:

$$\frac{\partial F}{\partial a_k} = 0.$$ \hspace{1cm} (28)

We can then approximate

$$P(D|K) \approx (2\pi\sigma^2)^{-T/2} \exp\left(-TF(\vec{a}^*)\right) \sqrt{\frac{2\pi}{T}} \frac{K+1}{|F''|^{-1}},$$ \hspace{1cm} (29)

where $|F''|$ is the determinant of the Hessian (second derivate matrix) of $F$ (and is of dimension $(K+1) \times (K+1)$, because polynomials of degree $K$ have $K+1$ free parameters). We can take a log on both sides and write:

$$\log P(D|K) \approx \log P^*(D|\vec{a}^*, K) - \frac{K+1}{2} \log \frac{T}{2\pi} - \frac{1}{2} |F''|.$$ \hspace{1cm} (30)

This equation has a very natural interpretation. On the left hand side, we have the likelihood of a model ($K$) given the data. On the right hand side, the first term corresponds to the best likelihood of the data given the model class (e.g., it is a likelihood evaluated at the best-fit parameters, $\vec{a}^*$ given $K$). The second term, however, is the penalty for model complexity. We
Figure 1: Left: True model, a polynomial of degree $K = 4$ (red), as well as 30 simulated data points, obtained by corrupting the true values with Gaussian distributed noise of $\sigma = 0.3$ magnitude. Middle: Using $\chi^2$ fitting to obtain the log likelihood on training data using $T = 30$ simulated data points, in black. Means and std are over 200 replicate draws of simulated data. Overfitting is evident since the fit continues to improve with the degree of the fitted polynomial. In contrast, evaluating the fit on withheld (testing) data, shown in blue, shows a sharp peak, indicating best generalization performance, at $K^* = 4$, which is the true degree of the generating polynomial. Similarly, BIC-corrected log-likelihood (red) also shows a peak at the correct value of $K^* = 4$. Right: analogous plot constructed using more simulated data points, $T = 300$. Here, overfitting is less pervasive, although still noticeable: the data is sufficient to pin the coefficients of the $K > 4$ orders close to zero. Cross-validation (blue) and BIC still correctly identify $K^* = 4$ as the best degree.

see that this penalty is linear in the number of model parameters, and grows logarithmically with the number of samples. The likelihood given the model (which in this example is directly related to $\chi^2$ as we showed above) will clearly increase as we use higher and higher $K$, but the model complexity term will decrease with $K$, leading to an optimal value of $K$ where the LHS of Eq. (30) is optimized. The prescription that in order not to overfit, and without doing cross-validation, one should compare nested models by comparing their likelihood minus the second term of Eq. (30) is known as the Bayes Information Criterion or BIC. An illustrative application of the cross-validation method and BIC is shown in Fig. 1.

7 Generalized Linear Models (GLMs)

GLMs are widely used in statistics; in sensory neuroscience, these models are used to predict the neural response (i.e., the number of spikes generated in a small time bin) given the stimulus $\tilde{s}$ preceding the response (i.e., a sequence of image frames in the recent past of a spike) [1]. The question about how neurons transform stimuli into spikes is known as the neural encoding problem.

The crucial realization is that the neural encoding process is probabilistic: even when exactly the same stimulus is presented many times, the response of a given neuron exhibits some amount of variability. For example, during certain times during the stimulus presentation, the neuron might be highly likely to fire spikes, but the exact number of spikes emitted will vary from trial to trial. A complete description of neural encoding thus requires a probabilistic model.

In the GLM framework, spike generation is assumed to be a point process. A point process is a random process where realizations consist of isolated points in time and/or space. Concretely,
this modeling assumption amounts to treating individual neural spikes as infinitely localized (in reality, a spike waveform is typically extended in time across $\sim 1 \text{ ms}$). Point process in time can be characterized by a rate function $f(t)$: this means that in a small time bin $\Delta$, the expected number of emitted spikes will be $f(t)\Delta$.

One common modeling assumption is that the probability distribution of the number of observed spikes $y_t$ in a time bin $[t, t + \Delta]$ is Poisson (remember, we are counting spikes, so $y_t$ is an integer count) whose mean is given by the rate function, that is

$$P(y_t | t) = \text{Poisson}(y_t; f(t)\Delta) = \frac{[f(t)\Delta]^y}{y!} \exp(-f(t)\Delta).$$ (31)

You should think of this distribution as a distribution of outcomes at time $t$ if the experiment (i.e. the stimulus) were repeated exactly many times (which can be done in sensory neuroscience). To fully specify the model, we need to (i) choose the nonlinear function $f$; (ii) specify how $f$ depends on time. When we assume that $f$ depends on $t$ only through the stimulus shortly preceding time $t$, we talk about Linear-Nonlinear-Poisson models of neural spiking (because we will assume that the rate function depends on the linear projection of the stimulus preceding time $t$, and the linear projection is “passed through” the nonlinear function $f$, which governs spiking according to the Poisson process. Alternatively, $f$ might depend not only on the stimulus, but on the past realization of spiking of a focal neuron (or other modeled neurons in a coupled network) as well. In this case the process is no longer Poisson, but still falls within the framework of Generalized Linear models. Crucially, the activity of the focal neuron $y_t$ at time $t$ is conditionally independent given the events in the past, an assumption that must be correct in the limit of $\Delta \to 0$ due to causality, and one that makes inference of GLMs for many coupled neurons tractable.

A further simplification consists of choosing $f(t) = \exp(x(t))$, where $x(t) = \tilde{k} \cdot \tilde{s}(t) + \mu$. That is, the average firing rate is an exponential function of the projection of the recent stimulus, $\tilde{s}(t)$, onto the neuron’s preferred feature (or filter), $\tilde{k}$. $\mu$ is the offset that modulates the overall spontaneous probability of spike generation. These choices can be relaxed to other nonlinear functions and higher-order dependence on the stimulus (e.g., quadratic projections, linear projections of nonlinear functions of the stimulus, or a nonlinear function of several different linear projections that need to be inferred).

The simplest probabilistic model of the neuronal response without history dependence is thus

$$\mathcal{L}(\tilde{k}, \mu) = \sum_{t=1}^{T} \log P(y_t | \tilde{s}(t)) = \sum_{t} \left[ -\log y_t! - \Delta \cdot \exp(\tilde{k} \cdot \tilde{s}(t) + \mu) + y_t(\tilde{k} \cdot \tilde{s}(t) + \mu) + y_t \log \Delta \right],$$ (32)

where we are given the spike train, $\{y_t\}$, and the stimulus sequence, $\tilde{s}(t)$, for time bins $t = 1, \ldots, T$, and the stimulus sequence. The task is to infer the filter $\tilde{k}$ and the offset $\mu$, the two parameters characterizing the LNP model. With the choice of an exponential function for the nonlinearity, the problem of maximizing the log-likelihood in Eq. (32) can be shown to be convex in the parameters, so a single global extremum of $\mathcal{L}$ exists, and simple gradient climbing can provide best estimates of the parameters.

Because time is discretized, the linear filter $\tilde{k}$ is a vector, usually spanning around half a second into the past for retinal neurons (this is the time-scale of the neurons’ primary stimulus sensitivity). At typical discretizations $\Delta$, $\tilde{k}$ will then have $\sim 20$ components, so likelihood maximization must be performed over a high-dimensional space. Often, one either includes
priors over the parameters to regularize, or parametrizes the filters in a lower-dimensional space (e.g., using basis functions), but such advanced topics are beyond the scope of this exercise.

An important generalization of the LNP model is the GLM model which includes the dependence of spiking at time $t$ on the past realization of spikes from a given neuron (or others in a coupled network). The key is to write the probability of a response as

$$P(\{y_t, y\_{t-1}, \ldots\}|t) = \prod_t P(y_t|t, \{y\_{t-1}, y\_{t-2}, \ldots\}),$$

(33)

where the explicit dependence on time is through stimulus, $\vec{s}(t)$. One can now again make simplifying assumptions about how past spiking may affect the rate of spike generation, for instance by keeping the Poisson assumption and postulating the functional form for $f$:

$$f(t, y_t^-) = f(\vec{k} \cdot \vec{s}(t) + \vec{h} \cdot [y_{t-1}, y_{t-2}, \ldots, y_{t-K}] + \mu).$$

(34)

Here, $y_t^-$ indicates the past realization (up until but excluding timebin $t$) of the spike train, and $h$ is a new linear filter acting on the past spike train; we assume that the causal influence of past spikes extends at most $K$ time bins into the past. Using this ansatz, the log likelihood of Eq. (32) remains essentially unchanged, and the inference now is carried over jointly for $\vec{k}$, $\vec{h}$, and $\mu$.

The primary neuroscience interest is to extract the linear filter(s) used by the neuron, because they mathematically capture the neuron’s preferred stimulus features: the neurons usually respond strongly to those stimuli that have a high overlap with the filter. Furthermore, if the GLM model recapitulates neural behavior precisely, one can proceed to ask questions about how much information the whole neural population provides about the stimulus, and how the stimulus information could be reconstructed, or decoded, from the neural spikes. With methods similar to ours, it has been established that in the retina, the typical filters respond strongly to a (spatially localized) local increase or decrease of the luminance; in the primary visual cortex, the neurons respond strongly to oriented bars of light at a given location and with a given width (edge fragments).

8 Study literature


9 Homework

1. Inferring the decay length constant. This problem is due to MacKay [2]. Unstable particles are emitted from a source and decay at a distance $x$, a real number that has an exponential probability distribution with characteristic length $\lambda > 0$. Decay events can only be observed if they occur in a window extending from $x = 1$ cm to $x = 20$ cm. $N$ decays are observed at locations $\{x_1, x_2, \ldots, x_N\}$ (this constitutes your data, $\mathcal{D}$). Write down the formula for the likelihood for the parameter $\lambda$ given data $\mathcal{D}$?

2. Let the true value of decay constant be $\lambda^* = 5$ cm. Generate simulated data $\mathcal{D}$ for the detection events (you can reuse your code from MDA Week 2!). Suppose we did not know
the true value of decay constant in an experiment, but prior (imprecise) measurements indicated that its value is $\bar{\lambda} = 8$ cm with a Gaussian error bar of $\sigma_{\lambda} = 4$ cm. Taking $N = 2, 4, 8, 16, 32, \ldots$ sequential data points from your simulated data, plot the posteriors $P(\lambda|D_N)$ for different number of data points. Plot the mode of the posterior, and its width (standard deviation) as a function of $N$. How many data points do you need, on average, to claim that the true value of $\lambda$ is less than 8 cm with 95% certainty?

3. Optional. Suppose that every measurement of the position $x_i$ comes with an IID drawn Gaussian error with $\sigma_x = 2$ cm. How do the results of the previous problem change?

4. Neural decoding. Let’s try to “decode” the stimulus, $s(t)$, by applying a linear filter to a vector of neural responses $\vec{r}(t)$. This vector consists of responses of neurons in a time window $[t - \tau, t + \tau]$, for some choice of $\tau$ (let us use $\tau = 333.3$ ms, or 10 stimulus frames, here). The stimulus is naturally discretized into frames of duration $\Delta$; discretize also the responses of the neurons using the same time binning, and set the response in a given time bin to be the number of spikes emitted by the neuron in that time bin. In this representation, the vector of responses will be of size $n \times (2\tau/\Delta)$, where $n$ is the number of neurons from which you want to decode. In linear decoding, the model for the stimulus prediction, $\hat{s}(t)$, is a linear filter $K$ applied to the responses, i.e., $\hat{s}(t) = \vec{K} \cdot \vec{r}(t)$, where the linear filter is chosen such that the prediction is closest to the real signal in the least-squares sense, $\chi^2 = T^{-1} \sum_{t=1}^{T} (s(t) - \hat{s}(t))^2$. Find the closed-form expression for $K$ (you can consult Ref [3], be careful with the different notations!) given the responses and the stimulus (it is just linear algebra). Take each neuron, one by one, and decode the stimulus from each one. Plot the typical filter $K$. How does it look like? Where is the filter zero, and would you expect that it has to be zero somewhere, and if so, why? To quantify how well the decoding works, compute the normalized cross-correlation between the real stimulus and the decoded stimulus for every neuron and report the average performance and std across the neuron population.

5. Let’s now try predicting from all 23 neurons at the same time; to this end your response vector, $\vec{r}$, as well as the filter vector, $\vec{K}$, will now be $23 \times$ longer than for a single neuron. Find $K$ by minimizing $\chi^2$, as before, but use only the first 2000, 4000, 8000 frames of the stimulus. Compute the performance (in terms of cross-correlation) between the true and predicted stimulus, over the same frames on which you computed the filter, and compare that with cross-correlation between the stimulus and the corresponding prediction over the frames 10000, ..., 20000 (that were not used in $\chi^2$ minimization). What is happening? What happens to the performance on both sets of frames if you do decoding while using the response vector for the time window $[t, t + \tau]$ instead of $[t - \tau, t + \tau]$?

6. To prevent overfitting, one regularizes the problem. The simplest (though often not the best) method is $L2$ regularization, also known as Ridge regression or Tikhonov regularization, where the function that one minimizes is $\chi^2 = T^{-1} \sum_{t=1}^{T} (s(t) - \hat{s}(t))^2 + \lambda \| \vec{K} \|_2$, where $\lambda$ is a tunable parameter (a Lagrange multiplier). $\lambda = 0$ gives us back the normal, unregularized solution; increasing $\lambda$ penalizes big filters. By learning the filter on the first 1000 frames and testing on frames 10000, ..., 20000, plot the performance on the testing set as a function of $\lambda$, and find the $\lambda$ that maximizes the performance. Look at the corresponding filters and compare them to the filters without regularization. Hint: the analytical solution for the filter, even with regularization, is still a simple linear algebra problem, so you might want to solve it that way. Optional. (i) If you have the analytical...
solution, can you interpret how adding a nonzero $\lambda$ “regularizes” the problem? (ii) What function, $\chi''^2$, would you minimize, if you wanted to penalize not the amplitude, but the ruggedness of the filters (i.e., you want your filters to fit the data and be as smooth as possible)? Is the solution to such regularization still analytically solvable?

7. Optional. **Inferring a Generalized Linear Model for a retinal ganglion cell.** Implement likelihood inference to find the linear filter, $\vec{k}$, and the offset, $\mu$, for neurons in the dataset provided (try it out on the first or first two neurons in the dataset, no need to run it for all the neurons). Use the discretization of the spike train into time bins that are of equal length to the frame durations, $\Delta$. Let the filter be a vector of 20 components, i.e., extending $20\Delta$ before the spike. You can maximize the likelihood either by using existing general purpose optimization routines (like Matlab’s `fminunc`, or routines in Numerical Recipes), or you can analytically compute the gradient of the likelihood and implement a simple gradient climbing.

References


