

Model Fitting and the GLM

Study Guide

by Richard J. Young, Ph.D.

richard@deepneuro.ai

Chapter 10 of "The NeuroAI Handbook"

Three choices, a hypothesis, a ruler for error, and a search, and the one framework that stretches a line to spikes.

One-sentence thesis. Fitting a model is three choices, a hypothesis, a ruler for wrongness, and a search for the parameters that minimize it, and the generalized linear model is the framework that stretches this from straight lines to spike counts and binary choices.

PART 1: What Fitting Actually Is

Which Parameters Best Describe the Data?

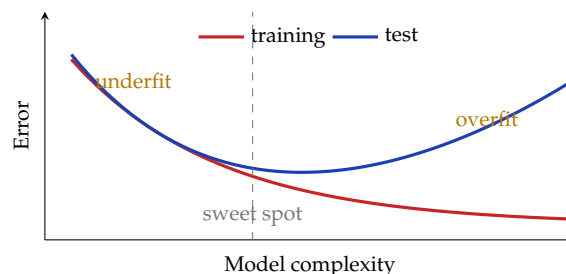
Every fitted model is the answer to a single question: given this data, which parameter values make my hypothesis describe it best? Answering it is always the same three moves. First, **choose a model**, a mathematical statement of how you think the data was generated, from a straight line to a deep network. Second, **choose a cost function**, a ruler that scores how far the model's predictions sit from reality. Third, **fit**, search for the parameters that make that score as small as possible, analytically when you can, numerically when you must. Then, crucially, **evaluate** on

data the fit never saw.

The deepest trap in that fourth step has a name: the **bias-variance tradeoff**. **Bias** is error from a model too simple to capture the pattern, it underfits, missing real structure. **Variance** is error from a model so flexible it chases the noise, it overfits, memorizing accidents of this particular sample. Total error is their sum, and it bottoms out not at maximum flexibility but at a sweet spot in between. Cranking up complexity always lowers training error and eventually *raises* the error that matters, on new data.

ANALOGY

Think of throwing darts. High **bias** is a tight cluster landing off the bullseye, consistent but wrong. High **variance** is darts scattered all around it, right on average but unreliable any single throw. You want the cluster both tight and centered, and reducing one often costs the other.



Training error falls forever with complexity; test error is U-shaped. The gap between them is variance, the height of the floor is bias, and the minimum of the test curve is the model you actually want.

When a Straight Line Is the Wrong Shape

The simplest model assumes the output is a linear combination of the inputs plus Gaussian noise:

$$y = \mathbf{X}\boldsymbol{\beta} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

Here $\mathbf{X}$ is the design matrix of features, $\boldsymbol{\beta}$ the coefficients to estimate. It is a fine model for an fMRI BOLD signal. It is the *wrong shape* for spike counts, which are non-negative integers, a line will happily predict -2.3 spikes and assume symmetric Gaussian scatter that count data never has.

The **generalized linear model (GLM)** fixes this by

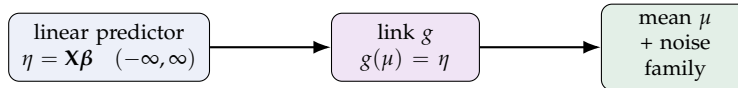
splitting the model into three swappable parts: a **linear predictor** $\eta = \mathbf{X}\boldsymbol{\beta}$, a **link function** g that connects that straight-line quantity to the *mean* of the response, and a **noise family** matched to the data type. Keep the linear machinery; change the link and the family to fit the data you actually have.

For spike counts the natural choice is a **Poisson** family with a **log** link, so the predicted rate $\lambda = \exp(\mathbf{x}^\top \boldsymbol{\beta})$ is always positive. It is the workhorse of neural encoding, fit by **maximum likelihood**, finding the $\boldsymbol{\beta}$ that makes the observed spikes most probable:

$$\ell(\boldsymbol{\beta}) = \sum_i [y_i \log \lambda_i - \lambda_i - \log(y_i!)]$$

ANALOGY

The link function is a power adapter. The linear predictor speaks one voltage, from $-\infty$ to $+\infty$; the response speaks another, a rate that must stay positive or a probability trapped in $[0, 1]$. The log and logit links are the plug shapes that let the two connect without frying anything.



- **Identity** $\mu = \eta$ (Gaussian, fMRI BOLD)
- **Log** $\log \mu = \eta$ (Poisson, spike counts)
- **Logit** $\log \frac{\mu}{1-\mu} = \eta$ (Bernoulli, choices)

The GLM keeps the linear predictor and swaps the link and noise family to match the data. One framework covers continuous, count, and binary responses.

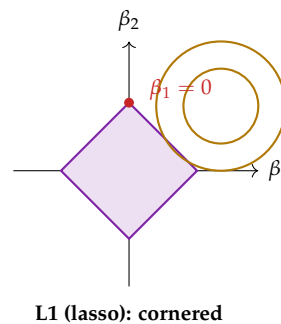
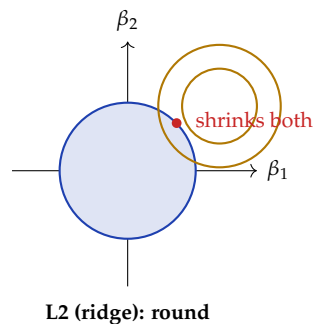
Catching, and Curbing, a Model That Memorizes

A model that fits the training data too well has learned its noise, and will stumble on anything new. You catch this with **cross-validation**: fit on part of the data, score on a held-out part, and if training performance towers over held-out performance, the model is overfit.

Catching it is half the job; curbing it is the other. **Regularization** adds a penalty on large coefficients to the cost, telling the search to prefer simpler solutions. The two standard penalties behave very differently:

L2 (ridge): $\sum_j \beta_j^2$

L1 (lasso): $\sum_j |\beta_j|$



Ridge (L2) shrinks every coefficient toward zero but rarely to exactly zero, good when many features each contribute a little. **Lasso (L1)** drives some coefficients to *exactly* zero, performing automatic feature selection, good when most features are irrelevant. The geometry explains why: L1's diamond-shaped constraint has sharp corners on the axes, and the solution tends to land on one, killing a coefficient outright.

There is an elegant reframing here. Regularization is a Bayesian **prior** on the weights: L2 is a Gaussian prior centered at zero, L1 a sharper Laplace prior. Penalizing large weights is just believing, before seeing data, that they are probably small.

Why L1 gives sparsity. The error contours (amber) first touch L2's smooth circle at an interior point (both weights nonzero) but tend to touch L1's diamond at a vertex on an axis, zeroing a coefficient.

PART 4: Choosing Between Models

Rewarding Fit, Punishing Complexity

Cross-validation compares models empirically; **information criteria** do it analytically, by scoring fit and charging a fee for each parameter. Both take the form “badness of fit plus a complexity tax,” and for both, **lower is better**:

$$\text{AIC} = 2k - 2 \ln \hat{L}$$

$$\text{BIC} = k \ln n - 2 \ln \hat{L}$$

Here k is the parameter count, n the sample size, and $\hat{L}$ the maximized likelihood. Adding parameters always raises $\hat{L}$ and lowers the fit term, but each new parameter also adds to the penalty, so the criterion falls, then rises, echoing the bias-variance U-curve from PART 1. **BIC** charges $\ln n$ per parameter versus AIC’s flat 2, so with any real dataset ($n > 7$) BIC penalizes complexity harder and prefers leaner models. Use them to referee competing GLMs without needing a separate held-out set for every comparison.

Data type	Noise family	Link	Example
Continuous	Gaussian	Identity: $\mu = \eta$	fMRI BOLD signal
Counts (spikes)	Poisson	Log: $\log \mu = \eta$	V1 orientation tuning
Binary	Bernoulli	Logit: $\log \frac{\mu}{1-\mu} = \eta$	Left/right choice

Quick Self-Test

1. Name the three choices every model fit requires, and say which one the cost function is.
2. In the bias-variance tradeoff, which term does a too-simple model inflate and which does a too-flexible one inflate? Why does training error mislead you about both?
3. Why is a Poisson GLM with a log link the right tool for spike counts, and what specifically goes wrong if you use plain linear regression?
4. L1 and L2 both fight overfitting, but only one selects features. Which, and what about its geometry causes that?
5. AIC and BIC share a form. What is that shared form, and why does BIC prefer simpler models on any real dataset?

If You Remember One Thing

Every model fit is a hypothesis, a ruler for error, and a search, and more flexibility is not more truth: it lowers training error while raising the error on new data. The GLM shows how far one clean idea travels, keep the linear predictor, change the link and noise family, and a straight line becomes the right tool for spikes, choices, and counts alike.

Richard J. Young, Ph.D. · richard@deepneuro.ai · Senior AI Research Scientist, UnitedHealth Group · UNLV Lee Business School · ORCID 0000-0002-1109-7552