PROF. DR.-ING. MARK SCHUTERA

NUMERICAL METHODS

UNFINISHED LECTURE NOTES

PUBLISHED BY UNFINISHED LECTURE NOTES

These notes are, by their very nature, unfinished, and they improve with every reader. If you spot an error, disagree with a framing, or want to add a source, a question, or a position, open a pull request at https://github.com/Quillstacks/LectureMaterial/tree/main/lecturenotes/notes_numerischemethoden. Every contribution is welcome.

2026-06-23 · earnest nectarine Eichhörnchen

Training a Model from First Principles

2026-06-23 · elegant huckleberry Falke

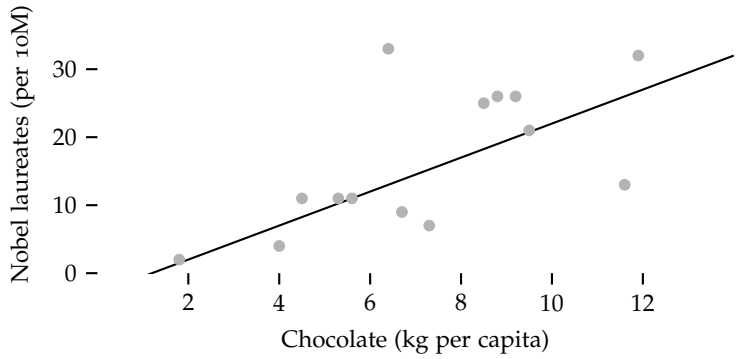
Oh, numerical methods are actually useful.

The Why

The previous six chapters developed numerical methdos. These topics may appear distinct on a first reading, but they share a wide-ranging application domain [1].

The running dataset for this chapter consists of fourteen countries with two associated quantities: Per-capita chocolate consumption and Nobel laureates per ten million inhabitants [2].

The model is a univariate linear regression of Nobel laureates on chocolate consumption (Figure 1), fit by gradient descent on the mean squared error. The five correspondences below are illustrated against this model and dataset.



The Learning Objectives are: Identify numerical method in application, apply standard numerical methods in practical contexts, and finally think in frameworks.

[1] A. Karpathy. A recipe for training neural networks. Blog post, https://karpathy.github.io/2019/04/25/recipe/, 2019. https://scholar.google.com/scholar?q=A+Recipe+for+Training+Neural+Networks+Karpathy

[2] F. H. Messerli. Chocolate consumption, cognitive function, and nobel laureates. *New England Journal of Medicine*, 367(16):1562–1564, 2012. https://scholar.google.com/scholar?q=Chocolate+Consumption+Cognitive+Function+Nobel+Laureates+Messerli

Figure 1: The dataset and an approximate by-eye fit. The fit computed by gradient descent will not differ substantially. The Messerli correlation is real but spurious; chocolate consumption does not cause Nobel prizes, and plausible confounders include per-capita wealth and education.

Model
$\hat{y}(x) = wx + b.$

Loss
$L(w, b) = \frac{1}{n} \sum_{i=1}^{n} (wx_i + b - y_i)^2.$

Overfitting as a Sanity Check

A STANDARD RECOMMENDATION IN TRAINING is to verify, that the model can drive the loss to approximately zero on a small subset of the training data. A failure to achieve this condition indicates an error in the model definition, the loss formulation, or the data pipeline.

RECALL CHAPTER 4. Newton's method converges in a single step from any starting point when applied to a quadratic function, because the linear approximation of the derivative is exact in that case.

AN EXAMPLE CALCULATION demonstrates the principle on a stripped down version of the dataset. Pick just two countries, Italy at $(4.0, 4)$ and the United Kingdom at $(9.5, 21)$, and extend the model by adding parameters,

$$\hat{y}(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4,$$

with $n = 4$ here, giving five free parameters for only two training points. The system is underdetermined: infinitely many parameter settings drive the training loss to zero, since any function passing through both training points qualifies. A correctly implemented training loop on this overparameterized model must reach one of these zero-loss solutions (within machine precision). Figure 2 shows one such solution:

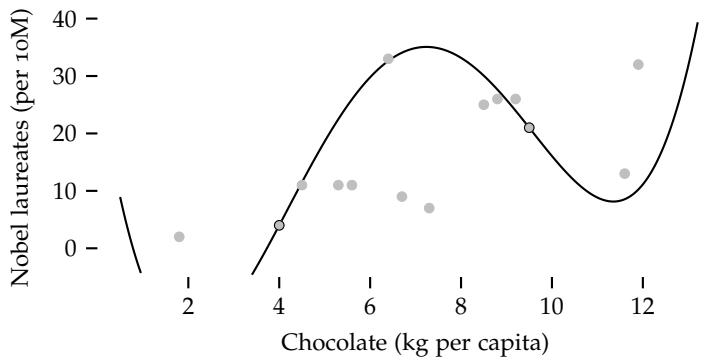


Figure 2: Two training points (circled gray dots) selected from the dataset. Any function through both points is a zero-loss solution. It interpolates the two training points exactly but oscillates elsewhere, a textbook illustration of overfit capacity.

THE SANITY-CHECK PRINCIPLE. The recommendation to drive the loss to zero on an overparameterized setup, where optimization is easy to achieve, is the machine-learning instance of testing a solver on a problem with a known exact solution.

THIS SEPARATES THE QUESTION OF WHETHER THE IMPLEMENTATION IS CORRECT FROM THE QUESTION OF WHETHER THE METHOD IS SUITABLE FOR THE PROBLEM AT HAND.

QUESTION. Figure 2 draws one zero-loss curve through the two training points. How many such curves exist? And what does that count imply about the geometry of the loss landscape this optimizer is asked to navigate? Welcome to the multiverse of global optima.

Loss at Initialization

A STANDARD RECOMMENDATION IN TRAINING is to inspect the loss before any optimization step has been taken. The expected initial value follows from the data statistics and the chosen initialization; a substantially different observed value points to a bug in the loss, the data, or the initialization.

RECALL CHAPTER 4. A standard check for an iterative method, such as Newton's Method or SGD is to evaluate it at a configuration whose answer can be computed by hand.

AN EXAMPLE CALCULATION on the chocolate dataset proceeds directly. Initializing the regression with $w = 0$ and $b = \bar{y}$ makes the model predict $\bar{y}$ for every input, so the mean squared error collapses to the mean squared deviation of the targets from their mean, that is the empirical variance of the targets in the $1/n$ convention. With $\bar{y} = 231/14 = 16.5$ and squared deviations summing to 1401.5,

$$\begin{aligned} L(0, \bar{y}) &= \frac{1}{n} \sum_{i=1}^{n} (y_i - \bar{y})^2 \\ &= \mathrm{Var}(y) \\ &\approx 100.1. \end{aligned}$$

CLASSIFIER BASELINE. For a K-class classifier initialized to uniform output over the classes, cross-entropy at step zero is $\log K$: $\log 2 \approx 0.69$ for binary, $\log 10 \approx 2.30$ for ten-way. The first printed value is fixed by the architecture choice, not by the data.

A correct training loop therefore prints approximately 100 as its first loss value. A value of 5 000 indicates a summed rather than averaged loss; a value of 0.0001 indicates an unintended normalization; a value far from 100 in either direction indicates that the bias was not educatedly initialized (Figure 3).

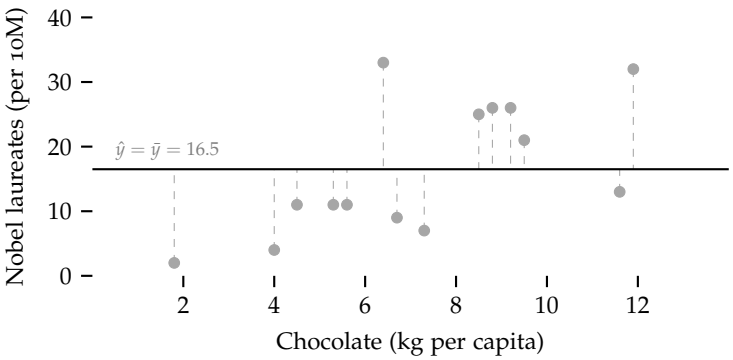


Figure 3: Initialization with $w = 0$ and $b = \bar{y}$ predicts the constant $\bar{y} = 16.5$ for every input (horizontal line). The vertical segments are the residuals $y_i - \bar{y}$; the mean of their squares is the initial loss, $\mathrm{Var}(y) \approx 100$.

THE BOUNDARY-VALUE PRINCIPLE. A hand-computable value at a known starting configuration is the universal first verification step for any numerical procedure; inspecting the initial loss against the variance of the targets is one instance. It separates a failure of implementation, visible at step zero, from a failure of optimization, which can only show up after the first step is taken.

Mini-Batch Stochastic Gradient Descent

THERE IS SAMPLE x_n AND FEATURE DIMENSION d, use mini-batch stochastic gradient descent in place of full-batch gradient descent when dealing with data at scale. At each iteration the gradient is estimated from a small random subset of the training data rather than from the entire dataset. The loss is an expectation,

$$L(\theta) = \mathbb{E}_{x \sim p_{\text{data}}}\big[\ell(x;\theta)\big],$$

and so is its gradient. This substitution is not new. The secant method in Chapter 4 replaced the exact derivative with a two-point finite-difference estimate, and Newton's method replaced the objective with a truncated Taylor expansion. Mini-batch stochastic gradient descent applies the same pattern on the data axis.

CURSE OF DIMENSIONALITY Monte Carlo integration averages the integrand at random samples and has standard error $1/\sqrt{N}$ independently of the dimension of the domain. That dimension independence is what distinguishes it from grid quadrature, whose cost grows exponentially.

AN EXAMPLE CALCULATION on the chocolate dataset makes the variance visible. Two mini-batches of size $B = 2$ are drawn from the fourteen countries; each yields the line through its two points,

$$\hat{y}_P = 3.54x - 10.18,$$
$$\hat{y}_Q = -1.43x + 17.43.$$

The approximated gradient is the delta between the mini-batch line and the current overall approximation, so each mini-batch supplies its own estimate of the same update direction, and the spread of those deltas quantifies the variance of the stochastic estimator.

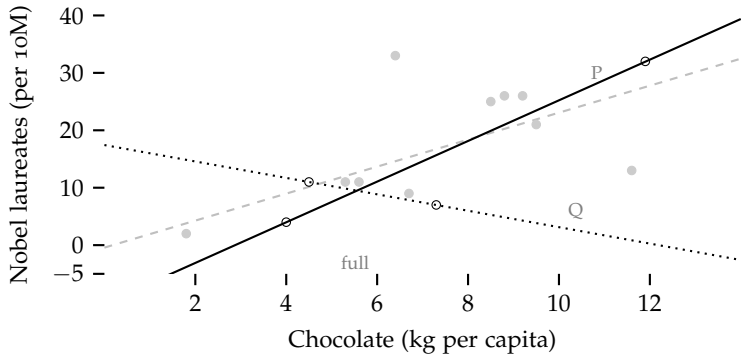


Figure 4: Two mini-batch fits to the chocolate dataset, each passing through its two sampled countries (P solid, Q dotted), together with the current overall approximation (gray dashed). Open circles mark the points used by each fit; the remaining countries are grayed out. The approximated gradient is the delta between a mini-batch line and the current overall approximation.

THE STOCHASTIC-GRADIENT PRINCIPLE. Mini-batch SGD is a moving target per step but not in expectation: iterates fluctuate in a neighborhood of the minimum whose radius shrinks with the step size and with the batch. The same noise that bars exact convergence dislodges iterates from shallow local minima where full-batch descent would settle.

QUESTION. For $n = 14$ a batch of $B = 4$ buys nothing; for $n = 10^9$ the full batch never fits in memory. At which n does the bottleneck flip with $1/\sqrt{B}$?

Training Multiple Random Seeds

REPEAT training from several random initializations of the model parameters. The variance across runs is diagnostic about the loss landscape, and the run with the lowest loss is the candidate global solution [3].

RECALL CHAPTER 5. For a non-convex objective, gradient descent terminates at a local minimum that depends on the initial point. An effective global-optimization strategy is random restarts.

[3] D. Picard. Torch.manual_seed(3407) is all you need: On the influence of random seeds in deep learning architectures for computer vision. *arXiv preprint arXiv:2109.08203*, 2021. https://scholar.google.com/scholar?q=Torch+manual+seed+3407+is+all+you+need+Picard

COMPOSING the slope from two parameters, with a quadratic trick on w to introduce a symmetry, gives a concrete example of the principle.

$$\hat{y}(x) = w^2 x + b.$$

with minima at $w \approx \pm 1.53$ ($L \approx 57.4$) and a saddle at $w = 0$ ($L \approx 100$).

WITH two random starts, shown as circles in Figure 5. Both basins recover the same fit $\hat{y} = 2.34x - 0.43$; a single seed would have surfaced only one solution in the whole space.

WHICH WOULD NOT HAVE HURT AT ALL to be fair.

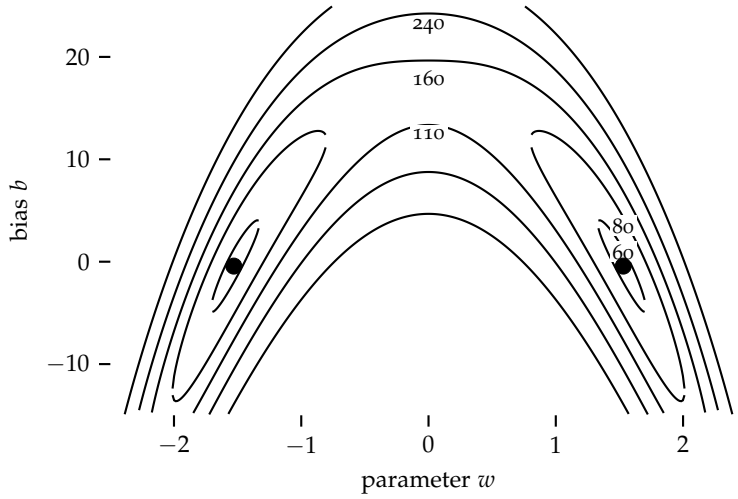


Figure 5: Loss landscape $L(w, b)$ for $\hat{y}(x) = w^2 x + b$ on the chocolate data, drawn as iso-loss contours. Filled circles mark the two converged minima at $(w, b) = (\pm 1.53, -0.43)$ with $L \approx 57.4$, sitting on the curved valley $b^*(w) = 16.5 - 7.221\, w^2$. My wife also says it looks like the whale from "Hitchhiker's Guide to the Galaxy". Okay, I added the book reference.

COST BENEFITS. Purely tuning the learning rate to chase variance that is actually basin-selection variance, is a common failure mode.

COST CAVEAT. Multi-seed training multiplies compute by the number of seeds. Routine at small-scale; at scale the method only survives in partial-data form. However, an empirical observation shows that, distinct trained minima are typically joined by low-loss paths in parameter space as model-scale grows [4].

[4] T. Garipov, P. Izmailov, D. Podoprikhin, D. Vetrov, and A. G. Wilson. Loss surfaces, mode connectivity, and fast ensembling of DNNs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. https://scholar.google.com/scholar?q=Loss+Surfaces+Mode+Connectivity+Fast+Ensembling+Garipov

Quantization for Inference

The savings fall in two areas. At training time, parameters and gradients move less data through memory; per-step throughput on bandwidth-bound hardware rises in proportion, and even when low precision demands more steps to reach a comparable loss the product, operations until convergence, often remains favourable. At inference the win compounds: floating-point multiplications collapse into cheap bitwise operations on dedicated hardware, and a model whose parameters fit in cache stops paying the memory-latency tax that dominates inference for anything larger than the L2.

Recall Chapter 2. Floating-point formats trade range against precision against bits. The appropriate format for a given computation is the smallest one whose range and precision suffice.

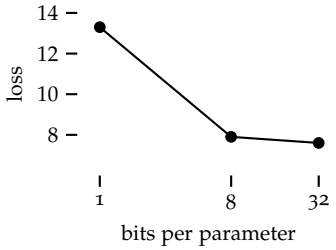


Figure 6: Loss against bits per parameter on the chocolate dataset. Binary precision pays a sharp penalty; the curve drops fast and flattens by integer precision. Most of the win sits in the first few bits.

An example calculation on the chocolate model makes the trade visible. Full-precision fitting, integer rounding to the nearest integer, and binary rounding to the nearest representative in $\{-1, +1\}$ produce three lines,

$$\hat{y}_{\text{full}} = 2.35x - 0.43,$$
$$\hat{y}_{\text{int}} = 2x,$$
$$\hat{y}_{\text{binary}} = x - 1.$$

with root-mean-square errors

$$\text{L}_{\text{full}} \approx 7.6,$$
$$\text{L}_{\text{int}} \approx 7.9,$$
$$\text{L}_{\text{binary}} \approx 13.3,$$

measured in Nobel laureates per ten million. Figure 7 overlays the three on the dataset.

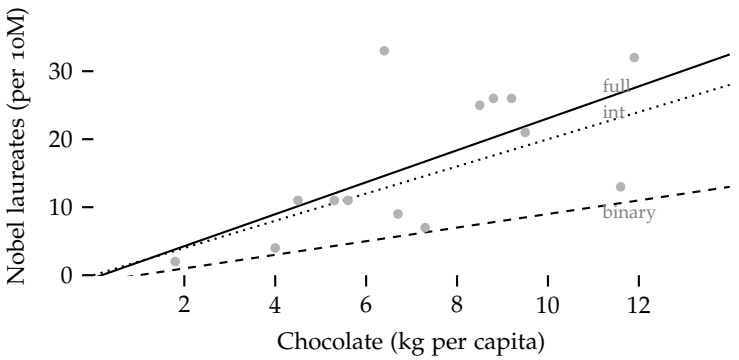


Figure 7: Full-precision fit (solid), integer approximation $(w, b) = (2, 0)$ (dotted), and binary approximation $(w, b) = (+1, -1)$ (dashed) on the chocolate dataset. Integer rounding barely moves the root-mean-square error; binary near doubles it.

And another tradeoff, where we give up precision and predictive performacnce, for runtime, memory and energy savings.

Question. Integer precision matched the full-precision error on this dataset. Why does any production model still ship in float32? Where does the extra precision actually earn its bits, and where is it just inertia?

Self-Reflection and Recap

SELF-REFLECTION Questions to guide your understanding:

- Why does driving the loss to zero on an overparameterized model detect bugs in the pipeline rather than in the optimizer?
- For the chocolate model initialized at $w = 0$, $b = \bar{y}$, the predicted first loss is $\mathrm{Var}(y) \approx 100$. What does a printed value of 5 000 tell you, and what does a printed value of 0.0001 tell you?
- In what precise sense is mini-batch stochastic gradient descent a Monte Carlo estimator? What is being averaged, and over what?
- For the factored model $\hat{y}(x) = w^2 x + b$, why is the origin a saddle of the loss rather than a minimum, and what makes zero initialization a practical trap even so?
- Why can a model trained at higher precision be deployed at much lower precision without retraining, while training at the lower precision tends to fail?

RECAP of Key Concepts:

- Driving the loss to zero on an overparameterized model is the training version of testing a solver on a problem with a known exact answer.
- Inspecting the initial loss against the variance of the targets separates a failure of implementation, visible at step zero, from a failure of optimization, which can only appear after the first step.
- Mini-batch stochastic gradient descent is Monte Carlo applied to the gradient integral, with standard error $1/\sqrt{B}$ independent of the parameter dimension.
- Training from multiple random seeds is the random-restarts heuristic from Chapter 5 applied to a non-convex landscape; the seed picks the basin through the sign of the initial fluctuation at the origin saddle, and mini-batch noise supplies the annealing component implicitly.
- Quantization for inference is the precision-selection decision from Chapter 2 applied to a different computation: inference tolerates far less precision than training because it does not accumulate gradient updates, so the appropriate format is the smallest whose error budget is acceptable for the deployment.

OUTLOOK. The same numerical methods (often) scale to billion-parameter models. The cost and the engineering grow; the methods do not.

From numerical methods to modeling at scale. The toolbox established in Chapters 1 to 6 is sufficient to describe the standard practices of model training at the scale of these notes. At larger scales the parameter count grows by orders of magnitude and the data, too. The engineering surrounding the training run grows correspondingly, and the vocabulary drifts further from the language of classical numerical analysis. None of this displaces the underlying methods. The mechanics that decide whether a model's learning succeeds remain those introduced earlier.