# Notes on susceptibilities

Andy Arditi & Claude 4.5 Opus

January 30, 2026

**Abstract**

The *susceptibility* of an observable to a data perturbation measures how that observable—a function of the model weights—responds when we shift the data distribution. If we trained on slightly different data, how would this quantity change? Baker et al. [2025] show that we can answer this question without retraining: the susceptibility equals a covariance, computed by sampling from the posterior induced by the *original* data.

One natural application: for each model component (attention head, MLP layer), define an observable measuring that component's contribution to loss. Computing susceptibilities for all components yields a vector per datapoint—a profile of which components matter for that prediction, and by how much. Crucially, datapoints that the model processes similarly will have similar profiles. Clustering datapoints by these profiles therefore groups them by how the model computes on them, revealing both the patterns the model has learned to distinguish and the internal structure it uses to do so.

## Contents

# 1 Background: the Bayesian view

To understand susceptibilities, we need to think about things from a Bayesian perspective.

## 1.1 Setup and notation

- **Data:** $n$ training examples $D_n = \{(x_i, y_i)\}_{i=1}^n$ drawn from distribution $q(x, y)$
- **Model:** Probability $p(y|x, w)$ parameterized by weights $w$
- **Losses:**

$$L_n(w) = -\frac{1}{n}\sum_{i=1}^{n} \log p(y_i|x_i, w) \quad \text{(empirical)}$$
$$L(w) = \mathbb{E}_{(x,y)\sim q}\left[-\log p(y|x, w)\right] \quad \text{(population)}$$

## 1.2 The Bayesian view: a distribution over weights

Normally, we think of training as finding a single $w^*$ that minimizes loss. The Bayesian perspective is different: our training data induces a *probability distribution* over weights, where low-loss configurations are more probable.

## 1.3 The posterior distribution

From Bayes' rule:

$$\underbrace{p(w|D_n)}_{\text{posterior}} \propto \underbrace{p(D_n|w)}_{\text{likelihood}} \cdot \underbrace{\varphi(w)}_{\text{prior}}.$$

Since $p(D_n|w) = \prod_i p(y_i|x_i, w) = e^{-nL_n(w)}$, the posterior is

$$p_n(w) = \frac{1}{Z_n} \cdot e^{-n\cdot L_n(w)} \cdot \varphi(w),$$

where $\varphi(w)$ is the prior and $Z_n = \int e^{-nL_n(w)}\varphi(w)\, dw$ is the normalizing constant (also called the *marginal likelihood* or *evidence*).

**Intuition**

Low loss ⇒ high probability. With lots of data (large $n$), the distribution becomes sharply peaked around the best-fitting weights.

## 1.4 Adding temperature

The *tempered posterior* adds a parameter $\beta \geq 0$:

$$p_n^\beta(w) = \frac{1}{Z_n^\beta} e^{-n\beta\cdot L_n(w)} \cdot \varphi(w), \qquad Z_n^\beta = \int e^{-n\beta L_n(w)}\varphi(w)\, dw.$$

The parameter $\beta$ is called the *inverse temperature*. It controls how much the data constrains our beliefs: $\beta = 0$ ignores the data entirely (giving the prior), $\beta = 1$ is standard Bayesian inference, and as $\beta \to \infty$ the distribution concentrates at the loss minimizer.

# 2 Observables and susceptibilities

## 2.1 What is an observable?

An **observable** $\phi(w)$ is any measurable function of the weights. Examples include:

- The total loss: $\phi(w) = L(w)$
- A single weight value: $\phi(w) = w_{17}$
- The norm of a layer: $\phi(w) = \|W_{\text{layer}}\|$
- How much loss increases when we perturb a specific component (see Section 4)

We will denote the expected value of an observable under the posterior as $\langle\phi\rangle$:

$$\langle\phi\rangle := \mathbb{E}_{w\sim p_n^\beta}[\phi(w)] = \int \phi(w)\cdot p_n^\beta(w)\,dw.$$

## 2.2 Perturbing the data distribution

Let $q$ be the original data distribution and let $q'$ be some other data distribution of interest ($q'$ is sometimes referred to as a **probe** distribution). For example, $q'$ might be a distribution over GitHub code, or a single example.

We define a mix of these two distributions, parameterized by $h \in [0, 1]$:

$$q_h = (1-h)\cdot q + h\cdot q'.$$

As we vary $h$, a chain of dependencies unfolds:

$$q_h \xrightarrow{\text{changes}} L^h(w) \xrightarrow{\text{changes}} p_n^\beta(w|h) \xrightarrow{\text{changes}} \langle\phi\rangle_h$$

**Definition 1** (Susceptibility)**.** *The* susceptibility of $\phi$ to the perturbation $q \to q_h$ *is defined as*

$$\chi := \frac{1}{n\beta}\frac{\partial}{\partial h}\langle\phi\rangle_h\Big|_{h=0}.$$

*It measures how fast the expected value of $\phi$ changes as we perturb the data toward $q'$.*

# 3 Computing susceptibilities

How do we actually compute $\chi$ in practice? The following lemma gives a feasible way to do so:

**Lemma 1** (Susceptibility as covariance)**.**

$$\chi = -\text{Cov}_{w\sim p_n^\beta}[\phi(w),\ \Delta L(w)],$$

*where $\Delta L(w) = L^{q'}(w) - L^q(w)$ is the difference in loss between the probe and original distributions, and the covariance is over the posterior $p_n^\beta$.*

This is useful because it means we can estimate $\chi$ from posterior samples: draw samples $w_1, \ldots, w_m$ from the posterior, evaluate $\phi(w_i)$ and $\Delta L(w_i)$ for each, and compute the sample covariance.

**Intuition:** Shifting data toward $q'$ reweights the posterior: weights that perform well on $q'$ (low $\Delta L$) gain probability. Whether $\langle\phi\rangle$ increases or decreases depends on whether high-$\phi$ weights are winners or losers under this reweighting. For example, if high-$\phi$ weights tend to have low $\Delta L$ (e.g., if there's negative covariance between $\phi$ and $\Delta L$), they gain probability, so $\langle\phi\rangle$ increases.

See Appendix A for the proof.

# 4 Application: component-wise susceptibilities

The general theory above applies to any observable. A particularly useful application, developed in Baker et al. [2025], Wang et al. [2025], Gordon et al. [2026], is to study how individual model components (attention heads, MLP layers) respond to data perturbations.

## 4.1 The component-wise loss observable

For neural network interpretability, we want to understand what each component does. A natural question: *does component $C$ help predict pattern $P$ more than average?*

To answer this, we use a specific observable. Consider component $C$ (say, attention head 1:3). Write the weights as $w = (u, v)$ where $v$ are $C$'s weights and $u$ are everything else. Define

$$\phi_C(w) = L(w) - L(w^*).$$

This measures how much loss increases when we perturb away from the trained weights $w^*$.

To make this specific to component $C$, we **restrict the sampling**: only vary $C$'s weights while keeping everything else frozen at $u^*$. The expected value becomes

$$\langle\phi_C\rangle = \int \phi_C(u^*, v) \cdot p_C^\beta(v)\, dv,$$

where $p_C^\beta(v) \propto e^{-n\beta L(u^*,v)}\varphi(v)$ is a posterior over just $C$'s weights.

**Why this is useful:** If $\langle\phi_C\rangle$ increases when we shift data toward pattern $P$, it means perturbations to $C$ hurt more on $P$ than on average data. In other words: $C$ is disproportionately important for $P$. This lets us build a "job description" for each component—which patterns it helps with, which it ignores, and which it actively suppresses.

**A note on notation.** The original papers write $\phi_C(w) = \delta(u - u^*)[L(w) - L(w^*)]$ with a Dirac delta. This formally expresses "integrate only over $C$'s weights." Operationally, the delta isn't literally evaluated—we restrict sampling to $C$'s subspace.

## 4.2 Per-token susceptibility

A particularly useful special case: the susceptibility of component $C$ to a single example $(x, y)$. Taking the probe to be $q' = \delta_{(x,y)}$ and writing $\ell_{(x,y)}(w) = -\log p(y|x, w)$:

$$\chi^C_{(x,y)} = -\mathrm{Cov}\left[\phi_C(w),\ \ \ell_{(x,y)}(w) - L(w)\right].$$

**Interpretation:** When we perturb $C$, does $(x \to y)$ suffer more than average, less, or about average?

| Sign | Term | Meaning |
|---|---|---|
| $\chi < 0$ | **Expression** | $C$ helps this prediction **more** than average |
| $\chi = 0$ | Neutral | $C$ helps about average |
| $\chi > 0$ | **Suppression** | $C$ helps **less** than average |

**Concrete Example**

For "The cat sat on the mat. The cat sat on the ___", an induction head should have $\chi < 0$ (strong expression) for predicting "mat."

Note: "Suppression" doesn't necessarily mean $C$ is hurting the prediction—it could just be irrelevant, or actively predicting something else.

### 4.3 The response matrix

Compute susceptibilities for many (component, probe) pairs to get the **response matrix**:

$$X_{ij} = \text{susceptibility of component } C_j \text{ to probe } q_i.$$

**What patterns reveal:**

- **Similar columns** ⇒ components with similar functions (same "circuit")
- **Similar rows** ⇒ probes involving similar patterns
- **PCA** reveals latent modes organizing both data and components

In a 2-layer transformer with 16 heads, PCA revealed:

| PC | Var. | What it captures |
|---|---|---|
| PC1 | 89.6% | Universal patterns (word boundaries)—all heads load similarly |
| PC2 | 3.8% | Separates **multigram heads** from **induction circuit** |
| PC3 | 1.3% | Separates two induction heads: code vs. natural language |

**Key Insight**

PCA on susceptibilities *automatically rediscovered* known circuits without being told they exist—purely from response patterns.

## 5 Summary

### 5.1 The method in one paragraph

To understand what a neural network component does, we ask: "If the training data shifted toward some probe (e.g., more code), how would this component's contribution change?"

This is the susceptibility—mathematically, the derivative of an expected observable with respect to a data perturbation. A key lemma shows this derivative equals a covariance that can be estimated from samples drawn from the posterior induced by the *original* data. The resulting susceptibilities reveal whether components help certain predictions more or less than average. By computing susceptibilities across many (component, probe) pairs and analyzing the response matrix, we automatically discover functional structure.

### 5.2 The five key ideas

1. **Bayesian framing:** Think of a trained model as inducing a distribution over weights, not just a single point. This lets us reason about "what if the data were different?"

2. **The causal chain:** Changing data changes the loss landscape, which changes which weights are optimal, which changes observables.

3. **The key lemma:** The effect of a data perturbation equals a covariance that can be estimated from samples drawn from the posterior induced by the *original* data. This makes the method practical.

4. **Expression vs. suppression:** The sign of a component's susceptibility to a datapoint reveals whether that component helps that prediction more or less than average.

5. **Structure discovery:** Patterns in the response matrix (found via PCA or clustering) automatically reveal functional organization—circuits, specialization, competition.

### 5.3 The one-liner

**The susceptibility of a component to a datapoint measures how much that component disproportionately helps (or doesn't help) that prediction.**

## 6 Where to go next

This document covered the core ideas. Topics for further study:

- **Connection to generalization:** Susceptibilities relate to the "local learning coefficient" from singular learning theory, which directly connects to Bayesian generalization error.
- **Modes:** The data distribution can be decomposed into "modes" (via SVD), and susceptibilities measure the model's response to these modes.
- **Scaling:** Promising results up to Pythia-1.4B Gordon et al. [2026].
- **Connection to singular learning theory:** See Watanabe [2009] for the mathematical foundations.

**Further reading:** Baker et al. [2025], Wang et al. [2025], Gordon et al. [2026].

## References


Garrett Baker, George Wang, Jesse Hoogland, and Daniel Murfet. Structural inference: Interpreting small language models with susceptibilities. *arXiv preprint arXiv:2504.18274*, 2025.

Andrew Gordon, George Wang, Garrett Baker, and Daniel Murfet. Towards spectroscopy: Susceptibility clusters in language models. *arXiv preprint*, 2026. Forthcoming.

George Wang, Garrett Baker, Andrew Gordon, and Daniel Murfet. Embryology of a language model. *arXiv preprint arXiv:2508.00331*, 2025.

Sumio Watanabe. *Algebraic Geometry and Statistical Learning Theory*. Cambridge University Press, 2009.

Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning*, pages 681–688, 2011.

# A Proof of the covariance formula

We prove that $\chi = -\mathrm{Cov}[\phi, \Delta L]$.

**Setup.** The expected value of $\phi$ under the perturbed posterior is

$$\langle \phi \rangle_h = \frac{N_h}{Z_h},$$

where we define:

$$N_h = \int \phi(w)\, e^{-n\beta L^h(w)} \varphi(w)\, dw,$$

$$Z_h = \int e^{-n\beta L^h(w)} \varphi(w)\, dw.$$

**Differentiating.** We want to compute $\frac{\partial}{\partial h}\langle \phi \rangle_h$. By the quotient rule:

$$\frac{\partial}{\partial h}\langle \phi \rangle_h = \frac{1}{Z_h}\frac{\partial N_h}{\partial h} - \frac{N_h}{Z_h^2}\frac{\partial Z_h}{\partial h}.$$

To compute $\frac{\partial N_h}{\partial h}$, we interchange the derivative and integral (see remark below):

$$\begin{aligned}
\frac{\partial N_h}{\partial h} &= \frac{\partial}{\partial h} \int \phi(w)\, e^{-n\beta L^h(w)} \varphi(w)\, dw \\
&= \int \phi(w) \cdot \frac{\partial}{\partial h} \left[ e^{-n\beta L^h(w)} \right] \varphi(w)\, dw \\
&= \int \phi(w) \cdot \left( -n\beta \frac{\partial L^h}{\partial h} \right) e^{-n\beta L^h(w)} \varphi(w)\, dw \\
&= -n\beta \int \phi(w) \frac{\partial L^h}{\partial h}\, e^{-n\beta L^h(w)} \varphi(w)\, dw.
\end{aligned}$$

Similarly for $Z_h$:

$$\begin{aligned}
\frac{\partial Z_h}{\partial h} &= \int \left( -n\beta \frac{\partial L^h}{\partial h} \right) e^{-n\beta L^h(w)} \varphi(w)\, dw \\
&= -n\beta \int \frac{\partial L^h}{\partial h}\, e^{-n\beta L^h(w)} \varphi(w)\, dw.
\end{aligned}$$

**Combining terms.** Substituting into the quotient rule:

$$\begin{aligned}
\frac{\partial}{\partial h}\langle\phi\rangle_h &= \frac{1}{Z_h}\left(-n\beta\int \phi\frac{\partial L^h}{\partial h}\, e^{-n\beta L^h}\varphi\, dw\right) - \frac{N_h}{Z_h^2}\left(-n\beta\int \frac{\partial L^h}{\partial h}\, e^{-n\beta L^h}\varphi\, dw\right) \\
&= -n\beta\cdot\frac{1}{Z_h}\int \phi\frac{\partial L^h}{\partial h}\, e^{-n\beta L^h}\varphi\, dw + n\beta\cdot\frac{N_h}{Z_h}\cdot\frac{1}{Z_h}\int \frac{\partial L^h}{\partial h}\, e^{-n\beta L^h}\varphi\, dw \\
&= -n\beta\left\langle \phi\cdot\frac{\partial L^h}{\partial h}\right\rangle_h + n\beta\langle\phi\rangle_h\left\langle\frac{\partial L^h}{\partial h}\right\rangle_h \\
&= -n\beta\left(\left\langle \phi\cdot\frac{\partial L^h}{\partial h}\right\rangle_h - \langle\phi\rangle_h\left\langle\frac{\partial L^h}{\partial h}\right\rangle_h\right) \\
&= -n\beta\cdot\mathrm{Cov}_h\left[\phi, \frac{\partial L^h}{\partial h}\right].
\end{aligned}$$

**Evaluating at $h = 0$.** Since $L^h(w) = (1-h)L^q(w) + hL^{q'}(w)$, we have:

$$\begin{aligned}
\frac{\partial L^h}{\partial h} &= \frac{\partial}{\partial h}\left[(1-h)L^q(w) + hL^{q'}(w)\right] \\
&= -L^q(w) + L^{q'}(w) \\
&= L^{q'}(w) - L^q(w) \\
&= \Delta L(w).
\end{aligned}$$

Therefore, evaluating at $h = 0$:

$$\frac{\partial}{\partial h}\langle\phi\rangle_h\Big|_{h=0} = -n\beta\cdot\mathrm{Cov}_{h=0}[\phi, \Delta L].$$

Since $\chi = \frac{1}{n\beta}\frac{\partial}{\partial h}\langle\phi\rangle_h\big|_{h=0}$ by definition, we conclude:

$$\chi = -\mathrm{Cov}[\phi, \Delta L],$$

where the covariance is under the unperturbed posterior ($h = 0$). □

**Remark on regularity.** The interchange of derivative and integral requires regularity conditions—typically that $\phi(w)\frac{\partial L^h}{\partial h}e^{-n\beta L^h(w)}\varphi(w)$ is dominated by an integrable function uniformly in $h$ near 0 (Leibniz integral rule). This is satisfied under mild assumptions on $\phi$, $L$, and $\varphi$, but we do not verify the details here. See Baker et al. [2025] for the original presentation, which also does not explicitly verify these conditions.

# B How to sample from the posterior: SGLD

The susceptibility formula $\chi = -\mathrm{Cov}[\phi, \Delta L]$ requires samples from the posterior. But neural network posteriors are high-dimensional and intractable. How do we actually get samples?

### B.1 The problem with the global posterior

The full Bayesian posterior $p(w|D) \propto e^{-nL_n(w)}\varphi(w)$ integrates over *all* weight configurations. For a neural network with millions of parameters, this is computationally infeasible. We can't enumerate all possible weights.

Moreover, we usually care about a *specific trained model* $w^*$, not the global posterior. We want to understand what this particular checkpoint does.

### B.2 The local posterior

The solution is to **localize** the posterior around $w^*$. Replace the prior $\varphi(w)$ with a Gaussian centered at the trained weights:

$$p_{\text{local}}(w) \propto \exp\left\{-n\beta L_n(w) - \frac{\gamma}{2}\|w - w^*\|^2\right\}.$$

The term $\frac{\gamma}{2}\|w - w^*\|^2$ acts like a "leash" that keeps samples close to $w^*$. The parameter $\gamma$ controls the leash strength:

- Large $\gamma$: Samples stay very close to $w^*$
- Small $\gamma$: Samples can wander further

This local posterior asks: "In the neighborhood of $w^*$, which weight configurations are probable?"

### B.3 Stochastic gradient Langevin dynamics (SGLD)

SGLD Welling and Teh [2011] is a method to sample from the local posterior. The idea: gradient descent finds modes (minima), but if we add noise, we can *sample* from the distribution instead.

**The SGLD update rule:**

$$w_{t+1} = w_t - \varepsilon\nabla_w\left[n\beta L_n(w_t) + \frac{\gamma}{2}\|w_t - w^*\|^2\right] + \sqrt{2\varepsilon}\cdot\eta_t,$$

where:

- $\varepsilon$ is the step size (learning rate)
- $\eta_t \sim \mathcal{N}(0, I)$ is Gaussian noise
- The gradient term pulls toward high-probability regions
- The noise term enables exploration (prevents collapse to a single point)

**Intuition**

**SGLD = SGD + noise.** Without noise, gradient descent converges to a point. With calibrated noise, we get samples from the posterior instead.

### B.4 In practice

Typical hyperparameters from Baker et al. [2025]:

- Localization strength: $\gamma = 300$
- Inverse temperature: $n\beta = 30$

- Step size: $\varepsilon = 0.001$
- Number of samples: 100–200 draws per chain
- Multiple chains (e.g., 4) for better coverage

For component-specific susceptibilities, SGLD is run only on the component's weights, with all other weights frozen at $w^*$.

### B.5 Why this works

Under mild conditions, as $\varepsilon \to 0$ and the number of steps $\to \infty$, SGLD samples converge to the true posterior distribution. In practice, we use finite $\varepsilon$ and finite steps, so we get *approximate* samples. The quality of the approximation depends on hyperparameter tuning.

The key insight: we don't need perfect samples. We just need samples good enough that the covariance estimate $\hat{\chi} = -\widehat{\mathrm{Cov}}[\phi, \Delta L]$ is reliable. Empirically, SGLD with the above hyperparameters works well for small transformers.