

The Bias-Variance Decomposition

Aayush Bajaj

July 10, 2026

Contents

1	motivation	1
2	the decomposition	2
2.1	setup	2
2.2	derivation	2
2.3	worked closed forms	3
3	the trade-off	3
4	a simulation	4
5	beyond squared loss	5
6	see also	5

there is exactly one theorem in machine learning that every practitioner rederives on a whiteboard at least once a year, and this is it. the squared-error risk of any learned predictor splits into three non-negative pieces — irreducible noise, squared bias, and variance — and every design decision you make (model class, regularisation strength, k , ensemble size, early stopping) is secretly a transaction between the last two.

the second candidate is bayes' rule, but that one belongs to the statisticians

1 motivation

think of a learning algorithm as a dart thrower. the bullseye is the true regression function at a query point; each dart is the prediction of a model trained on a *different* random sample of data. two independent failure modes emerge:

- **bias:** the thrower's aim is systematically off — the darts cluster around the wrong spot. no amount of extra throws fixes this; the model class simply cannot represent the target.
- **variance:** the aim wobbles — the darts scatter widely around their own centre. any single throw (a single trained model) may land far from where the thrower is aiming, even if the aim is true.

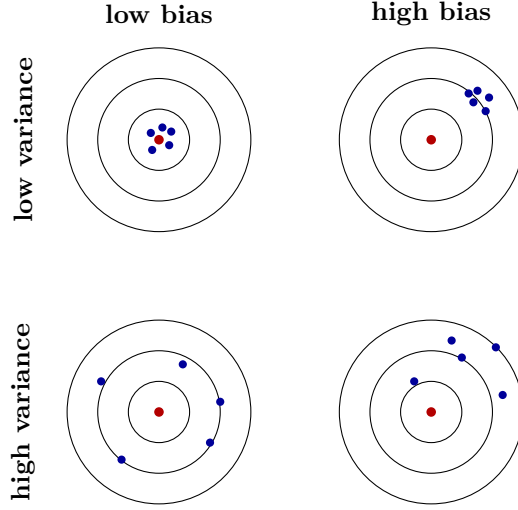


Figure 1: the classic dartboard picture: rows vary variance, columns vary bias. each dot is the prediction of a model trained on a fresh sample.

2 the decomposition

2.1 setup

fix a query point x_0 . data are generated as

$$y = f(x) + \varepsilon, \quad \mathbb{E}[\varepsilon] = 0, \quad \text{Var}(\varepsilon) = \sigma^2, \quad (1)$$

with ε independent of everything else. a learning algorithm consumes a training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ (itself random) and emits a predictor $\hat{f}_{\mathcal{D}}$. write the *average prediction* over training sets as

$$\bar{f}(x_0) = \mathbb{E}_{\mathcal{D}} [\hat{f}_{\mathcal{D}}(x_0)]. \quad (2)$$

we care about the expected squared error on a fresh observation $y_0 = f(x_0) + \varepsilon_0$:

$$\text{Err}(x_0) = \mathbb{E}_{\mathcal{D}, \varepsilon_0} [(y_0 - \hat{f}_{\mathcal{D}}(x_0))^2]. \quad (3)$$

2.2 derivation

two applications of the same trick — add and subtract a mean, watch the cross term die.¹

step 1 — peel off the noise. substitute $y_0 = f(x_0) + \varepsilon_0$:

$$\begin{aligned} \text{Err}(x_0) &= \mathbb{E} \left[(\varepsilon_0 + f(x_0) - \hat{f}_{\mathcal{D}}(x_0))^2 \right] \\ &= \mathbb{E}[\varepsilon_0^2] + 2 \mathbb{E}[\varepsilon_0] \mathbb{E} [f(x_0) - \hat{f}_{\mathcal{D}}(x_0)] + \mathbb{E}_{\mathcal{D}} \left[(f(x_0) - \hat{f}_{\mathcal{D}}(x_0))^2 \right] \\ &= \sigma^2 + \mathbb{E}_{\mathcal{D}} \left[(f(x_0) - \hat{f}_{\mathcal{D}}(x_0))^2 \right]. \end{aligned}$$

¹cross terms vanish because ε_0 is independent of $\mathcal{D}$ and both differences are centred.

step 2 — split the estimation error. add and subtract $\bar{f}(x_0)$:

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}\left[(f(x_0) - \hat{f}_{\mathcal{D}}(x_0))^2\right] &= \mathbb{E}_{\mathcal{D}}\left[(f(x_0) - \bar{f}(x_0) + \bar{f}(x_0) - \hat{f}_{\mathcal{D}}(x_0))^2\right] \\ &= (f(x_0) - \bar{f}(x_0))^2 + \mathbb{E}_{\mathcal{D}}\left[(\hat{f}_{\mathcal{D}}(x_0) - \bar{f}(x_0))^2\right],\end{aligned}$$

where the cross term is $2(f(x_0) - \bar{f}(x_0)) \mathbb{E}_{\mathcal{D}}[\bar{f}(x_0) - \hat{f}_{\mathcal{D}}(x_0)] = 0$ by definition of $\bar{f}$. hence

$$\boxed{\text{Err}(x_0) = \underbrace{\sigma^2}_{\text{noise}} + \underbrace{(f(x_0) - \bar{f}(x_0))^2}_{\text{bias}^2} + \underbrace{\mathbb{E}_{\mathcal{D}}[(\hat{f}_{\mathcal{D}}(x_0) - \bar{f}(x_0))^2]}_{\text{variance}}} \quad (4)$$

this is eq. (7.9) of (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009). three remarks:

- σ^2 is the **bayes error** at x_0 : no learner beats it, however clever.
- bias measures the gap between the truth and the *average* model — a property of the model class and algorithm, not of any single fit.
- variance measures how much a single fit wobbles around that average — a property of the data-sensitivity of the algorithm.

2.3 worked closed forms

k-nearest neighbours. for a knn regressor with neighbours $x_{(1)}, \dots, x_{(k)}$ of x_0 (inputs held fixed),

$$\text{Err}(x_0) = \sigma^2 + \left(f(x_0) - \frac{1}{k} \sum_{\ell=1}^k f(x_{(\ell)})\right)^2 + \frac{\sigma^2}{k}, \quad (5)$$

so k is a literal bias–variance dial: growing k averages away variance as σ^2/k while dragging in ever-farther neighbours whose f values pollute the mean — bias creeps up. small k reverses the trade (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009).

linear regression with p features. the fit at x_0 is $\hat{f}(x_0) = x_0^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$, a fixed linear functional $h(x_0)^\top \mathbf{y}$ of the responses, so its variance is $\|h(x_0)\|^2 \sigma^2$; averaged over the training inputs this is $\frac{p}{n} \sigma^2$. model complexity here *is* the parameter count — variance grows linearly in p , and bias is whatever the best linear approximant to f leaves behind.

an estimator, not a curve. the decomposition is really a statement about estimators. for the plug-in variance estimator $\hat{\sigma}^2 = \frac{1}{n} \sum_i (X_i - \bar{X})^2$, one computes $\mathbb{E}[\hat{\sigma}^2] = \frac{n-1}{n} \sigma^2$: bias $-\sigma^2/n$, vanishing at rate $1/n$ — the mse of any estimator likewise splits as $\text{mse} = \text{bias}^2 + \text{variance}$ (Wasserman, Larry, 2010).

3 the trade-off

sweep model complexity and the three terms move in opposite directions: bias^2 falls (richer classes approximate better), variance rises (richer classes chase noise), noise

stays put. the sum is u-shaped, and the art of supervised learning is to sit at the bottom of the u.

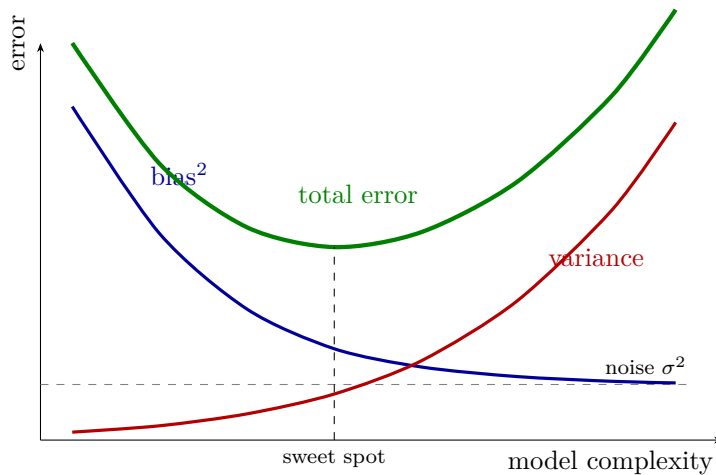


Figure 2: test error decomposed against model complexity. the optimum sits where the marginal drop in bias^2 equals the marginal rise in variance.

every regularisation device in the standard toolkit is a move along this curve:

knob	effect
ridge / lasso penalty $\lambda \uparrow$	shrinks coefficients: bias $\uparrow$, variance $\downarrow$
knn neighbours $k \uparrow$	averages more: bias $\uparrow$, variance $\downarrow \sigma^2/k$
tree depth $\uparrow$	finer partitions: bias $\downarrow$, variance $\uparrow$
bagging / ensembling	variance $\downarrow$ (approaching the correlation floor), bias $\approx \text{const}$
early stopping	implicit shrinkage: bias $\uparrow$, variance $\downarrow$

bagging deserves its footnote: averaging B identically-distributed fits with pairwise correlation ρ leaves variance $\rho\sigma_f^2 + \frac{1-\rho}{B}\sigma_f^2$ — the reason random forests decorrelate trees with feature subsampling (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009).

4 a simulation

monte carlo makes the decomposition tangible: fix $f(x) = \sin 2\pi x$, draw many training sets, fit polynomials of increasing degree, and measure bias^2 and variance empirically at a test grid.

```

1 import numpy as np
2
3 rng = np.random.default_rng(0)
4 f = lambda x: np.sin(2 * np.pi * x)
5 n, sigma, trials = 30, 0.4, 400
6 xg = np.linspace(0.05, 0.95, 50)           # test grid
7
8 def experiment(degree):
9     preds = np.empty((trials, xg.size))
10    for t in range(trials):
11        x = rng.uniform(0, 1, n)
12        y = f(x) + rng.normal(0, sigma, n)
13        coef = np.polyfit(x, y, degree)      # least-squares polynomial fit
14        preds[t] = np.polyval(coef, xg)
15    fbar = preds.mean(axis=0)                 # average model
16    bias2 = ((fbar - f(xg)) ** 2).mean()     # squared bias, averaged over grid
17    var = preds.var(axis=0).mean()           # variance, averaged over grid

```

```

18     return bias2, var
19
20 print(f"{'deg':>3} {'bias^2':>9} {'var':>9} {'sum+noise':>10}")
21 for d in (1, 2, 3, 5, 7, 9):
22     b2, v = experiment(d)
23     print(f"{'d':>3} {'b2':>9.4f} {'v':>9.4f} {'b2 + v + sigma**2:>10.4f}")

```

output — bias^2 collapses at degree 3 (a cubic can bend like one period of a sine; note degree 2 barely helps, a parabola can't), variance climbs overall — pausing for a small dip at degree 3, where the fit first stops fighting the data — then blows up violently as high-degree fits on $n = 30$ points start whipping at the grid edges:

deg	bias ²	var	sum+noise
1	0.1555	0.0257	0.3412
2	0.1508	0.0473	0.3581
3	0.0028	0.0243	0.1872
5	0.0001	0.0394	0.1994
7	0.0028	1.0079	1.1707
9	0.0365	9.1342	9.3307

(push the degree past 10 and unscaled `np.polyfit` additionally becomes numerically ill-conditioned — the vandermonde matrix is nearly singular — which inflates the measured “variance” with floating-point noise. real lesson: centre and scale your features.)

5 beyond squared loss

the clean additive split is special to squared error. two caveats worth carrying around:

- **0–1 loss does not decompose additively.** for classification the interaction is stranger: on the wrong side of the decision boundary, extra variance can *reduce* expected error by occasionally flipping a systematically-wrong prediction to the right class. squared-loss intuition transfers qualitatively, not quantitatively (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009).
- **modern over-parameterised models bend the u.** past the interpolation threshold, test error can descend a *second* time — the “double descent” phenomenon: enormously over-parameterised networks generalise despite fitting noise, because implicit regularisation of the training dynamics selects low-norm interpolants.² the decomposition still holds — it is an identity — but variance no longer explodes monotonically with parameter count.²

the probabilistic view of the same identity — decomposing posterior predictive uncertainty — is developed in ch. 8 of (Deisenroth, Marc Peter and Faisal, A. Aldo and Ong, Cheng Soon, 2020).

6 see also

- **cross validation** — estimating the test error you just decomposed
- **regularised regression** — buying bias to sell variance, with a dial

²belkin et al. (2019), *reconciling modern machine-learning practice and the classical bias-variance trade-off*, pnas 116(32).

- **loss functions** — why squared error owns this identity
- **performance metrics** — what to report once you stop decomposing

references

Deisenroth, Marc Peter and Faisal, A. Aldo and Ong, Cheng Soon (2020). *Mathematics for Machine Learning*, Cambridge University Press.

Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome (2009). *The Elements of Statistical Learning*, Springer.

Wasserman, Larry (2010). *All of Statistics: A Concise Course in Statistical Inference*, Springer.