

A Catalogue of Loss Functions

Aayush Bajaj

July 10, 2026

Contents

1 regression losses: which functional are you asking for?	1
1.1 squared error targets the mean	1
1.2 absolute error targets the median	2
1.3 huber buys robustness with a dial	2
2 classification losses: everything is a function of the margin	2
2.1 what each surrogate's minimiser is	3
2.2 classification calibration	3
3 the mle correspondence	4
4 robustness: read the tails	4
5 the minimisers, verified numerically	4
6 choosing a loss	5
7 see also	6

a loss function is not a detail of training — it is the *definition of the problem*. choose squared error and you have asked for the conditional mean; choose absolute error and you have asked for the median; choose hinge and you have asked only for the decision boundary; choose cross-entropy and you have asked for the whole probability. this page catalogues the standard losses, proves what each one's minimiser actually is, and draws the classic picture that unifies the classification zoo: every one of them is a bribe paid to make the 0–1 loss differentiable.

corollary: two teams 'solving the same problem' with different losses are solving different problems

1 regression losses: which functional are you asking for?

let $r = y - f(x)$ denote the residual. the workhorse losses:

$$L_{\text{sq}}(r) = r^2 \quad L_{\text{abs}}(r) = |r| \quad L_{\delta}^{\text{huber}}(r) = \begin{cases} \frac{1}{2}r^2 & |r| \leq \delta \\ \delta|r| - \frac{1}{2}\delta^2 & |r| > \delta \end{cases}$$

1.1 squared error targets the mean

minimise the pointwise risk over a constant prediction c at a given x :

$$\frac{\partial}{\partial c} \mathbb{E}[(Y - c)^2 \mid X = x] = -2\mathbb{E}[Y \mid X = x] + 2c \stackrel{!}{=} 0 \implies c^* = \mathbb{E}[Y \mid X = x].$$

the population minimiser of squared loss is the **conditional mean** — the regression function. this is why the whole **bias-variance decomposition** is a squared-loss story (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009).

1.2 absolute error targets the median

for L_{abs} , the risk $\mathbb{E}|Y - c|$ has derivative $\mathbb{P}(Y < c) - \mathbb{P}(Y > c)$, which vanishes when both tails carry equal mass: $c^* = \text{median}(Y \mid X = x)$. more generally the **pinball loss** $L_\tau(r) = r(\tau - \mathbb{I}[r < 0])$ targets the τ -quantile — the basis of quantile regression.

1.3 huber buys robustness with a dial

huber loss is squared error inside the band $|r| \leq \delta$ and absolute error outside, spliced so the derivative is continuous. its gradient is *clipped* at $\pm\delta$: a wild outlier pulls the fit with bounded force, whereas under squared loss the pull grows linearly with the outlier's distance. the minimiser interpolates between mean ($\delta \rightarrow \infty$) and median ($\delta \rightarrow 0$).

the derivative of the loss is the 'influence' of a point on the fit; unbounded derivative = unbounded influence = one bad sensor owns your model

2 classification losses: everything is a function of the margin

encode labels $y \in \{-1, +1\}$ and let $f(x) \in \mathbb{R}$ be a real-valued score. all the standard classification losses depend on the pair (y, f) only through the **margin** $m = yf(x)$: positive margin means correctly classified, magnitude means confidence.

loss	$\phi(m)$	used by
0-1 (the truth)	$\mathbb{I}[m \leq 0]$	what you actually pay
hinge	$\max(0, 1 - m)$	support vector machines
squared hinge	$\max(0, 1 - m)^2$	l2-svm variants
logistic	$\log(1 + e^{-m})$	logistic regression, neural nets
exponential	e^{-m}	adaboost
squared	$(1 - m)^2$	least-squares classification

the 0-1 loss is the one you care about, and it is computationally hopeless: piecewise constant, gradient zero almost everywhere, and empirical-risk minimisation under it is np-hard for general linear classifiers. every other row is a **surrogate**: a convex upper bound (after suitable scaling) on the step function that gradients can grip.

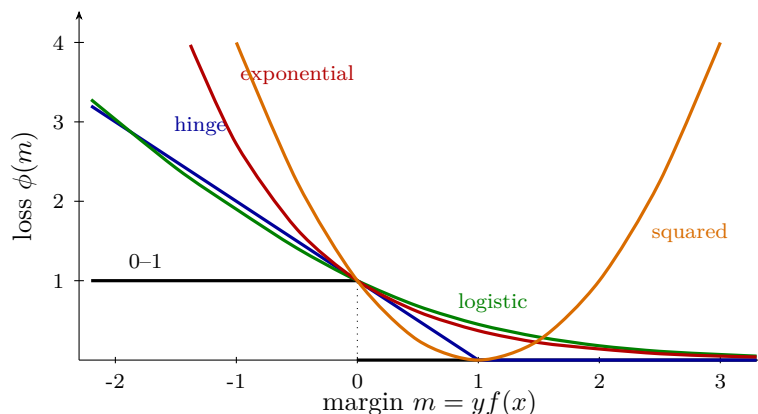


Figure 1: the margin-loss zoo: 0–1 truth and its convex surrogates. logistic is scaled by $1/\log 2$ so it passes through $(0, 1)$. note squared loss turning back up for $m > 1$ — it penalises being too right.

2.1 what each surrogate’s minimiser is

fix x , write $p = \mathbb{P}(Y = +1 \mid X = x)$, and minimise the conditional surrogate risk $p\phi(f) + (1-p)\phi(-f)$ over $f \in \mathbb{R}$. setting the derivative to zero for each loss:

loss	population minimiser $f^*(x)$	reading
exponential	$\frac{1}{2} \log \frac{p}{1-p}$	half the log-odds
logistic	$\log \frac{p}{1-p}$	the log-odds (hence “logistic”)
squared	$2p - 1$	$\mathbb{E}[Y \mid x]$ for ± 1 labels
hinge	$\text{sign}(p - \frac{1}{2})$	the bayes <i>decision</i> , nothing more

the exponential-loss computation is worth doing once because it explains adaboost. differentiate $p e^{-f} + (1-p) e^f$, set to zero: $-p e^{-f} + (1-p) e^f = 0$, so $e^{2f} = p/(1-p)$, giving $f^* = \frac{1}{2} \log \frac{p}{1-p}$ — boosting’s additive fit converges to half the log-odds, which is why $\text{sign}(f)$ is the bayes classifier. this is the argument of ch. 10 of (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009).

two structural observations:

- **hinge is the odd one out:** its minimiser is already thresholded. an svm estimates the *boundary* directly and discards p ; logistic regression estimates p and gets the boundary for free. if you need calibrated probabilities from an svm you must bolt them on afterwards (platt scaling — see [performance metrics](#) on calibration).
- **squared loss is self-sabotaging for classification:** beyond $m = 1$ it *increases* — it punishes examples for being classified too confidently, dragging the boundary toward well-classified points.

2.2 classification calibration

when does minimising a surrogate actually solve the original problem? call ϕ **classification-calibrated** if driving the ϕ -risk to its minimum forces the 0–1 risk to the bayes risk. for convex ϕ the characterisation is clean: it is calibrated if and only if it is differentiable at 0 with $\phi'(0) < 0$.¹ hinge, logistic, exponential, and squared all pass; this is the license that lets svms and boosting optimise the “wrong” objective and still get the right answer asymptotically. what differs between them is *everything else* — rates, robustness, and what function of p you can recover.

¹bartlett, jordan & mcauliffe (2006), *convexity, classification, and risk bounds*, jasa 101(473). the general (non-convex) condition is that the minimiser of the conditional ϕ -risk has the same sign as $p - \frac{1}{2}$ whenever $p \neq \frac{1}{2}$.

references

Deisenroth, Marc Peter and Faisal, A. Aldo and Ong, Cheng Soon (2020). *Mathematics for Machine Learning*, Cambridge University Press. Goodfellow, Ian (2016). *Deep Learning*, MIT Press. Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome (2009). *The Elements of Statistical Learning*, Springer.

3 the mle correspondence

every loss is secretly a negative log-likelihood: minimising $\sum_i L(y_i, f(x_i))$ is maximum likelihood under the noise model $p(y | x) \propto e^{-L(y, f(x))}$, and adding a regulariser is map estimation under the matching prior. the dictionary:

loss (minimise)	noise model (maximise likelihood)
squared error	gaussian, fixed variance
absolute error	laplace
huber	gaussian centre, laplace tails
logistic / binary cross-entropy	bernoulli via sigmoid
multiclass cross-entropy	categorical via softmax
pinball (quantile)	asymmetric laplace

the correspondence runs both directions and is the honest way to choose: if your residuals are heavy-tailed, squared loss is a *wrong likelihood*, not a neutral default. the erm–mle bridge is developed carefully in (Deisenroth, Marc Peter and Faisal, A. Aldo and Ong, Cheng Soon, 2020); the deep-learning convention of “cross-entropy + softmax” is exactly categorical mle, with the pleasant gradient $\hat{p} - y$ that neither saturates nor explodes for confident wrong answers (Goodfellow, Ian, 2016).

exponential loss has no clean likelihood — one reason statisticians read adaboost as fitting an additive model rather than doing inference

4 robustness: read the tails

rank the losses by what one adversarial point can do, worst first:

- **exponential**: a single large negative margin costs $e^{|m|}$ — adaboost concentrates exponentially on its noisiest, most mislabelled examples, the mechanism behind its label-noise fragility (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009).
- **squared** (regression): influence linear in the residual, unbounded. one 40-sigma sensor glitch relocates your mean.
- **logistic / hinge**: asymptotically *linear* in the negative margin — the gentlest convex penalties; a mislabelled point costs $O(|m|)$, not $e^{|m|}$.
- **absolute / huber**: bounded influence; the robust-statistics choices.
- **0–1**: perfectly robust (an outlier costs at most 1) and perfectly unoptimisable. the entire catalogue is the art of approaching its robustness while keeping a usable gradient.

5 the minimisers, verified numerically

both tables above make sharp predictions; check them by brute force — a constant fit under each regression loss on skewed data with outliers, and the conditional-risk minimiser of each margin loss at $p = 0.8$.

```
1 import numpy as np
2
3 rng = np.random.default_rng(2)
4
5 # skewed, outlier-ridden sample: lognormal body + a few wild points
6 y = np.concatenate([rng.lognormal(0.0, 0.75, 500), [25.0, 30.0, 40.0]])
7
8 def huber(r, delta=1.0):
```

```

9     a = np.abs(r)
10     return np.where(a <= delta, 0.5 * r**2, delta * (a - 0.5 * delta))
11
12 grid = np.linspace(0.0, 6.0, 6001)      # candidate constants c
13 sq = grid[np.argmin([np.mean((y - c)**2) for c in grid])]
14 ab = grid[np.argmin([np.mean(np.abs(y - c)) for c in grid])]
15 hu = grid[np.argmin([np.mean(huber(y - c)) for c in grid])]
16
17 print(f"sample mean = {y.mean():.3f}    sample median = {np.median(y):.3f}")
18 print(f"argmin squared loss c* = {sq:.3f}    (the mean)")
19 print(f"argmin absolute loss c* = {ab:.3f}    (the median)")
20 print(f"argmin huber loss c* = {hu:.3f}    (in between, outlier-resistant)")
21
22 # classification: minimiser of each margin loss as a function of p = P(Y=1|x)
23 p = 0.8
24 fs = np.linspace(-4, 4, 8001)
25 exp_f = fs[np.argmin(p * np.exp(-fs) + (1 - p) * np.exp(fs))]
26 log_f = fs[np.argmin(p * np.log1p(np.exp(-fs)) + (1 - p) *
    ↪ np.log1p(np.exp(fs)))]
27 sqm_f = fs[np.argmin(p * (1 - fs)**2 + (1 - p) * (1 + fs)**2)]
28 print(f"\np = {p}: logit/2 = {0.5*np.log(p/(1-p)):.3f}, logit =
    ↪ {np.log(p/(1-p)):.3f}, 2p-1 = {2*p-1:.3f}")
29 print(f"empirical minimisers: exponential = {exp_f:.3f}, "
30       f"logistic = {log_f:.3f}, squared = {sqm_f:.3f}")

```

```

sample mean    = 1.476    sample median = 0.956
argmin squared loss c* = 1.476    (the mean)
argmin absolute loss c* = 0.956    (the median)
argmin huber loss c* = 1.097    (in between, outlier-resistant)

```

```

p = 0.8: logit/2 = 0.693, logit = 1.386, 2p-1 = 0.600
empirical minimisers: exponential = 0.693, logistic = 1.386, squared = 0.600

```

the three wild points (25, 30, 40) drag the squared-loss fit half a unit above the median; huber concedes just over a quarter of that. and the margin-loss minimisers land on $\frac{1}{2} \logit(0.8) = 0.693$, $\logit(0.8) = 1.386$, and $2p - 1 = 0.6$ to three decimals — the table is not a metaphor.

6 choosing a loss

- **regression, trusted gaussian-ish noise:** squared. you want the mean and the closed forms.
- **regression, heavy tails or sensor glitches:** huber (tune δ to a robust scale estimate) or absolute if you genuinely want the median.
- **classification, need probabilities:** logistic / cross-entropy. calibrated in expectation, sane gradients, mle pedigree.
- **classification, need only the boundary, want sparse support:** hinge — the loss that makes **kernel methods** practical, since its solutions depend on few support vectors.
- **never:** squared loss for classification (self-sabotage beyond $m = 1$); exponential loss with label noise.

and remember the loss you train with is not the metric you report — the mapping between the two lives in **performance metrics**.

7 see also

- **the bias-variance decomposition** — an identity that belongs to squared loss alone
- **performance metrics** — evaluation-time counterparts of these training-time objects
- **kernel methods** — hinge loss plus a kernel is the svm
- **the no free lunch theorem** — no loss is the right loss for every problem