



Regression and Classification Losses

Losses and training | MSE, MAE, Huber, log-cosh, quantile (pinball), BCE, cross-entropy with label smoothing, focal and hinge: formulas, gradients, robustness to outliers, class weighting and which one to pick (PyTorch 2.x names)

#Losses #Regression #Classification #CrossEntropy #FocalLoss

What a loss decides

A courier fined for late deliveries		A model trained with a loss
Fine grows with the square of the delay	≈	MSE: one very late parcel dominates the bill
Fine grows linearly with the delay	≈	MAE: every minute costs the same, outliers matter less
Squared up to 1 h, linear beyond	≈	Huber: precise when close, calm about disasters
Late costs 9× more than early	≈	Pinball with $\tau = 0.9$: the courier plans for the 90th percentile
Fine only if not early by a safe margin	≈	Hinge: zero penalty once correct with margin
WHERE THE ANALOGY BREAKS A courier sees the whole fine; a network only sees the gradient of the loss at its current prediction, so the slope of the penalty, not its size, drives learning.		

A loss is the penalty schedule the model is fined with. Change the schedule and the same model learns a different answer

Before Choose and debug a supervised loss, can you explain...

	Not yet	Partly	Yes
MUST KNOW Derivatives and the chain rule the gradient of a loss w.r.t. a prediction; Learn: maths: derivatives	☐	☐	☐
MUST KNOW Entropy, cross-entropy, KL why $-\log p$ is the cost of a probability; this sheet does not re-derive them; Learn: maths: information theory	☐	☐	☐
MUST KNOW Softmax and sigmoid logits $\rightarrow$ probabilities; Learn: deep-learning: neurons and MLP	☐	☐	☐
HELPFUL Evaluation metrics the loss is not the metric you report; Learn: machine-learning: evaluation metrics	☐	☐	☐
HELPFUL Imbalanced data and calibration resampling and thresholds live there; Learn: machine-learning: imbalanced data	☐	☐	☐

What this sheet assumes (each in one line)

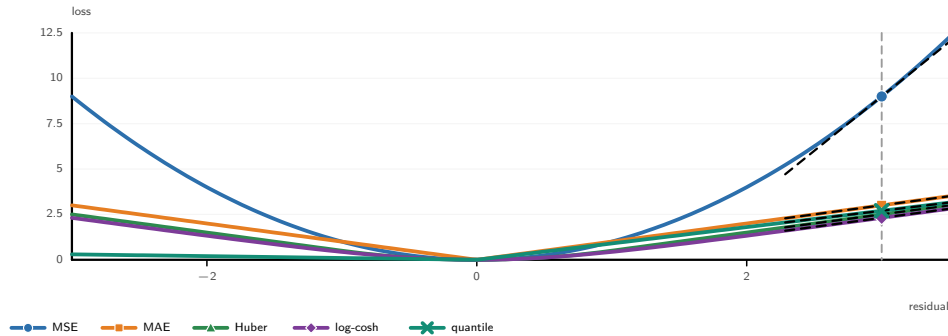
A loss sets **what the optimum is** (mean, median, quantile, calibrated probability, margin) and **how hard each example pulls** (its gradient). Pick it from the decision you will make with the prediction, then check the gradient behaves on your outliers and rare classes.

Regression losses

Loss	Formula	Gradient $dL/d\hat{y}$	Optimum
MSE	r^2	$-2r$	conditional mean
MAE	$ r $	$-\text{sign}(r)$	conditional median
Huber (δ)	$\frac{1}{2}r^2$ if $ r \leq \delta$; $\delta(r - \frac{1}{2}\delta)$ else	$-r$, capped at $\pm\delta$	between mean and median
Log-cosh	$\log \cosh r$	$-\tanh(r)$, in $(-1, 1)$	close to Huber with $\delta \approx 1$
Pinball (τ)	$\max(\tau r, (\tau - 1)r)$	$-\tau$ if $r > 0$, $1 - \tau$ if $r < 0$	τ -quantile
Poisson NLL	$\hat{y} - y \log \hat{y}$	$1 - y/\hat{y}$	mean of a count ($\hat{y} > 0$)

where y = true target (target unit); $\hat{y}$ = model prediction (target unit); r = residual $y - \hat{y}$ (target unit); δ = Huber threshold where quadratic turns linear (PyTorch default 1.0) (target unit); τ = target quantile, e.g. 0.9 for the 90th percentile (0-1)
Per-example losses for a residual $r = y - \hat{y}$. The optimum column is what the model converges to when the loss is averaged over noisy targets

Against the residual $r = y - \hat{y}$



loss	$L(r)$	dL/dr	L at 3	dL/dr at 3
MSE	r^2	$2r$	9	6
MAE	$ r $	$\text{sign}(r)$ (0 at $r = 0$)	3	1
Huber	$\frac{1}{2}r^2$ if $ r \leq \delta$, else $\delta(r - \frac{1}{2}\delta)$	r if $ r \leq \delta$, else $\delta \cdot \text{sign}(r)$	2.5	1
log-cosh	$\ln \cosh(r)$	$\tanh(r)$	2.31	0.995
quantile	$\tau \cdot r$ if $r \geq 0$, else $(\tau - 1) \cdot r$	τ if $r > 0$, else $\tau - 1$	2.7	0.9

where $r = y - \hat{y}$: target minus prediction ($dL/d\hat{y} = -dL/dr$); $\delta = 1$: Huber switch from quadratic to linear; $\tau = 0.9$: quantile level (under-prediction costs τ , over-prediction $1 - \tau$); the dashed segment at each point is the tangent (slope = gradient)

Regression losses against the residual, with their values and gradients at $r = 3$: MSE's gradient keeps growing, Huber's, MAE's and log-cosh's stay bounded; pinball with $\tau = 0.9$ is tilted so under-prediction costs $9 \times$ more

PROBLEM A batch of 4 house prices (in 100 kEUR) has one data-entry outlier. Compare MSE, MAE and Huber ($\delta = 1$) and the gradient each example sends back.

- GIVEN**
- True $y = [3, 5, 2, 20]$; predicted $\hat{y} = [2.5, 5, 3, 6]$ (100 kEUR)
 - Residuals $r = y - \hat{y} = [0.5, 0, -1, 14]$
 - Gradient magnitude per example: MSE $2|r|$, MAE 1 (0 at $r = 0$), Huber $\min(|r|, \delta)$

1 MSE

$$(0.25 + 0 + 1 + 196) / 4 = 49.31$$

Why: the outlier is $196 / 197.25 = 99.4\%$ of the loss

2 MAE

$$(0.5 + 0 + 1 + 14) / 4 = 3.875$$

Why: outlier is $14 / 15.5 = 90\%$ of the loss, but its gradient is only 1

3 Huber $\delta = 1$

$$0.125 + 0 + 0.5 + 1 \cdot (14 - 0.5) = 14.125; / 4 = 3.53$$

Why: quadratic for $|r| \leq 1$, linear beyond

4 Gradients of the outlier

MSE 28, MAE 1, Huber 1

Why: the gradient, not the loss value, moves the weights

5 Gradients of the normal points

MSE 1 and 2; Huber 0.5 and 1

Why: Huber keeps MSE's precision near zero

ANSWER Under MSE the outlier pulls $28 / (28 + 1 + 2) = 90\%$ of the update; under Huber $1 / (1 + 0.5 + 1) = 40\%$: the fit follows the three good houses

CHECK If the 20 were a real luxury house you want to predict, MSE is right and Huber under-fits it: robustness is a choice about the data, not a free win

One outlier, three losses. Same batch, same predictions: how much of the update the outlier drives

💡 Quantile regression in practice

Train three heads with $\tau = 0.1, 0.5, 0.9$ for an 80 % prediction interval; LightGBM `objective='quantile', alpha=0.9, scikit-learn HistGradientBoostingRegressor(loss='quantile', quantile=0.9)`. Check coverage on held-out data: about 80 % of targets should fall inside.

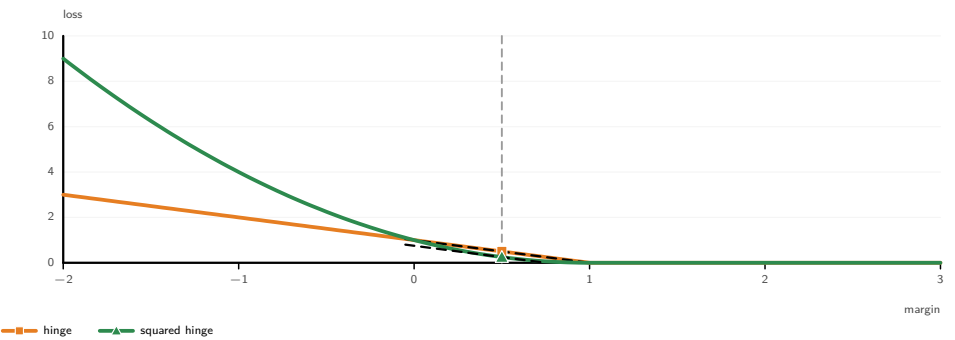
🕒 Classification losses

Loss	Formula	Gradient w.r.t. logits	PyTorch
BCE	$-[y \log p + (1 - y) \log(1 - p)]$, $p = \sigma(z)$	$p - y$	<code>BCEWithLogitsLoss(pos_weight=)</code>
Cross-entropy	$-\log p_y$, $p = \text{softmax}(z)$	$p - \text{onehot}(y)$	<code>CrossEntropyLoss(weight=)</code>
CE + label smoothing	$-\sum_k q_k \log p_k$, $q_k = (1 - \varepsilon)[k = y] + \varepsilon/K$	$p - q$	<code>label_smoothing=0.1</code>
Focal	$-\alpha(1 - p_t)^\gamma \log p_t$	CE gradient $\times$ modulating factor	<code>torchvision sigmoid_focal_loss</code>
Hinge	$\max(0, 1 - ys)$, $y \in \{-1, +1\}$	$-y$ if $ys < 1$, else 0	<code>MultiMarginLoss (multi-class)</code>
Squared hinge	$\max(0, 1 - ys)^2$	$-2y(1 - ys)$ if $ys < 1$	<code>LinearSVC</code> default

where $z = \text{logit}(s)$, the raw scores (-); p , p_k , p_y = predicted probability; of class k ; of the true class (0-1); p_t = probability given to the true class (p if $y = 1$, $1 - p$ if $y = 0$) (0-1); y = label: 0/1 for BCE, class index for CE, -1/+1 for hinge (-); q_k = smoothed target for class k (0-1); ε , K = smoothing amount (typ. 0.1); number of classes (-, count); γ , α = focusing exponent (typ. 2) and class balance weight (typ. 0.25) (-); s = classifier score (signed margin) (-)

Classification losses. All take logits in PyTorch; feeding probabilities to a loss that expects logits is the commonest bug

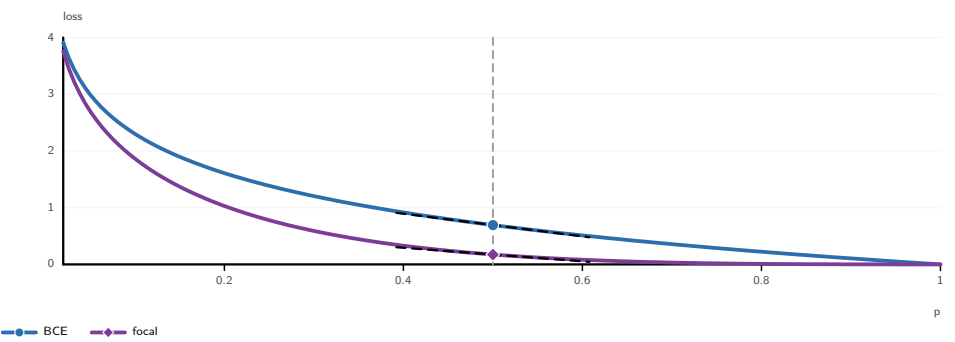
Against the margin $m = y \cdot f(x)$



loss	$L(m)$	dL/dm	L at 0.5	dL/dm at 0.5
hinge	$\max(0, 1 - m)$	-1 if $m < 1$, else 0	0.5	-1
squared hinge	$\max(0, 1 - m)^2$	$-2 \cdot \max(0, 1 - m)$	0.25	-1

where $m = y \cdot f(x)$: label $y \in \{-1, +1\}$ times the raw score $f(x)$; $m \geq 1$ is correct with margin; the dashed segment at each point is the tangent (slope = gradient)

Against the probability p of the true class



loss	$L(p)$	dL/dp	L at 0.5	dL/dp at 0.5
BCE	$-\ln p$	$-1 / p$	0.693	-2
focal	$-(1 - p)^\gamma \cdot \ln p$	$\gamma(1 - p)^{\gamma-1} \cdot \ln p - (1 - p)^{\gamma-1} / p$	0.173	-1.19

where p : predicted probability of the true class (the loss pushes it towards 1); $\gamma = 2$: focal focusing ($\gamma = 0$ gives BCE); the dashed segment at each point is the tangent (slope = gradient)

Classification losses against the margin (or the true-class probability for focal), values and gradients at margin 0.5: hinge is exactly 0 beyond margin 1, BCE never quite reaches 0, focal with $\gamma = 2$ flattens the easy region

Quantity	Plain CE	Label smoothing $\epsilon = 0.1$
softmax p	[0.665, 0.245, 0.090]	same p
target	[1, 0, 0]	$q = [0.933, 0.033, 0.033]$
loss	$-\ln 0.665 = 0.408$	$0.9 \times 0.408 + 0.1 \times \text{mean}(0.408, 1.408, 2.408) = 0.508$
gradient p - target	[-0.335, 0.245, 0.090]	[-0.268, 0.211, 0.057]
gradient zero when	$p_0 = 1$ (logit gap $\rightarrow \infty$)	$p = q$ (finite logit gap)

Worked by hand: logits $z = [2, 1, 0]$, true class 0, $K = 3$. Smoothing raises the loss a little and stops the gradient once p matches q , not at $p = 1$

📄 Focal loss in numbers ($\gamma = 2$)

Easy example, $p_t = 0.9$: $CE = -\ln 0.9 = 0.105$, factor $(1 - 0.9)^2 = 0.01 \rightarrow 0.00105$ (100× smaller). Hard example, $p_t = 0.1$: $CE = 2.303$, factor $0.81 \rightarrow 1.865$ (1.2× smaller). Dense detectors see ~100,000 easy background anchors per image; focal loss stops them drowning the few objects.

⚙️ Class weighting, robustness and choosing

Item	Formula	Example ($N = 1,000$, $K = 2$, $950 / 50$)
weight of class c	$w_c = \frac{N}{K n_c}$	$w_0 = 1000/1900 = 0.53$; $w_1 = 1000/100 = 10$
BCE positive weight	$pos_weight = n_{neg}/n_{pos}$	$950 / 50 = 19$

where N = number of training examples (count); K = number of classes (count); n_c , n_{pos} , n_{neg} = examples in class c; positives; negatives (count)

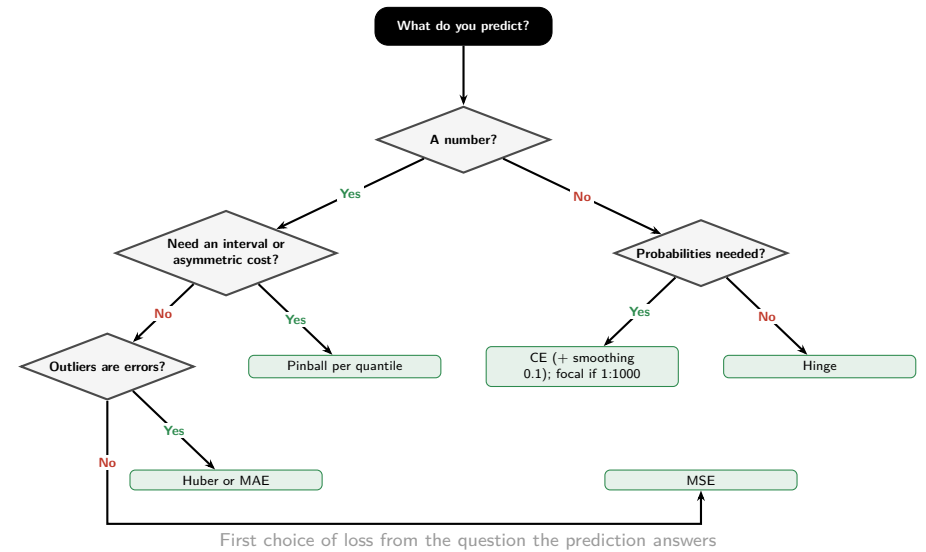
Balanced class weights (scikit-learn `class_weight='balanced'`; pass as `CrossEntropyLoss(weight=w)`). Resampling and threshold moving are in machine-learning: imbalanced data

⚠️ Weights distort probabilities

Class weights and focal loss shift predicted probabilities away from true frequencies: a 0.5 output no longer means 50 %. If you need calibrated probabilities (risk scores, LLM judges), train with plain CE and move the threshold, or recalibrate on held-out data.

	MSE	Huber / log-cosh	Pinball	Recommended Cross-entropy	Focal	Hinge
Target	real value	real value	real value	class	class	class
Outliers	sensitive	robust	robust	✗	✗	✗
Optimum	mean	≈ mean near, median far	quantile τ	calibrated probabilities	re-weighted	margin, no probabilities
Typical use	clean regression, forecasting the mean	noisy sensors, box regression (smooth L1)	intervals, stock and capacity planning	default classifier, LLM next token	extreme imbalance, dense detection	linear SVM, some ranking

Which loss for which goal (starting point; then check on your metric)



BCEWithLogitsLoss independent sigmoid per output; multi-label Tell: Can two labels be true at once?	vs	CrossEntropyLoss one softmax over classes; exactly one label
Huber threshold δ in target units Tell: Is the loss scaled by $1/\beta$?	vs	Smooth L1 Huber divided by β (PyTorch)
Label smoothing softens targets for all examples Tell: Does the change depend on how easy the example is?	vs	Focal loss down-weights easy examples
Class weights fixed per class Tell: Is the multiplier the same for every member of a class?	vs	Sample weights per example (importance, recency)

Loss pairs that get mixed up, and the one question that separates them

🚩 Myths, recap and self-test

YOU MAY HEAR

"Use MSE for everything continuous"

BUSTED

✓ ACTUALLY

MSE predicts the mean and lets outliers drive training

For skewed targets (prices, durations) predict $\log(y)$ with MSE, or use Huber, or a quantile loss if decisions are asymmetric.

YOU MAY HEAR

"Softmax then CrossEntropyLoss"

BUSTED

✓ ACTUALLY

PyTorch's CrossEntropyLoss already applies log-softmax

Applying softmax twice squashes gradients and training stalls; pass raw logits. Same for BCEWithLogitsLoss vs sigmoid + BCELoss (also less stable).

YOU MAY HEAR

"Focal loss is always better on imbalance"

BUSTED

✓ ACTUALLY

It helps with millions of easy negatives, rarely on ordinary 1:10 tabular data

Try class weights or threshold moving first; focal adds two hyperparameters and uncalibrated outputs.

YOU MAY HEAR

"Lower validation loss means a better model"

BUSTED

✓ ACTUALLY

Only for the loss's own goal

A weighted or smoothed loss can fall while accuracy or calibration gets worse; always track the business metric next to the loss.

Loss beliefs that cost accuracy

Remember

01

The loss picks the optimum

mean (MSE), median (MAE), quantile (pinball), probability (CE)

02

The gradient picks who pulls

bounded gradients (Huber, MAE) tame outliers

03

Pass logits, not probabilities

CE and BCEWithLogits apply the softmax/sigmoid themselves

04

Label smoothing 0.1 is a cheap default

for large classifiers; drop it if you need calibrated probabilities

05

Imbalance: weights first, focal for extreme cases

and remember both distort probabilities

Q1 What does an MSE-trained model predict for noisy targets? And MAE?

A The conditional mean; MAE gives the conditional median.

Q2 Huber gradient for a residual of 14 with $\delta = 1$?

A Magnitude 1 (capped at δ), against 28 for MSE.

Q3 Pinball loss with $\tau = 0.9$: cost of under-predicting by 1 vs over-predicting by 1?

A 0.9 vs 0.1, so the model leans high and learns the 90th percentile.

Q4 Gradient of softmax cross-entropy w.r.t. the logits?

A $p - \text{onehot}(y)$: predicted minus target probabilities.

Q5 Smoothed target for the true class with $\epsilon = 0.1$ and $K = 10$?

A $(1 - 0.1) + 0.1/10 = 0.91$; every other class gets 0.01.

Q6 Why does focal loss use $(1 - p_t)^\gamma$?

A To shrink the loss of easy, already-correct examples (p_t near 1) so rare hard ones dominate.

Q7 When is hinge loss exactly zero?

A When $y \cdot s \geq 1$: correct side with a margin of at least 1.

Q8 Balanced weight of a class with 50 of 1,000 examples, 2 classes?

A $1000 / (2 \times 50) = 10$.

Q9 Multi-label (several tags per image): which loss?

A BCEWithLogitsLoss, one sigmoid per tag; softmax CE assumes exactly one class.

Q10 Why log-cosh over Huber?

A It is smooth everywhere (twice differentiable), useful for second-order methods such as XGBoost custom objectives.