

The ML Cookbook

Numbered formulas of applied machine learning, each with its source

Alexander Jung

Version 0.1.0

The formulas below are taken from the entries of the *Dictionary of Applied Machine Learning* (<https://dictionaryofml.org>, DOI 10.5281/zenodo.21569296), whose notation they use. Under each formula: the entries it comes from, one source pinpointed to the smallest locatable unit, and, for an identity, a check in numpy that the build ran before typesetting. Cite a line as “ML Cookbook, eq. (n)”.

Empirical risk minimization and linear regression

Empirical risk minimization

$$\hat{h} \in \arg \min_{h \in \mathcal{H}} \frac{1}{m} \sum_{r=1}^m L(\mathbf{z}^{(r)}, h) \quad (1)$$

Entries: `erm`. Source: Shalev-Shwartz and Ben-David (2014), Sect. 2.2 [1]. The learned hypothesis minimizes the average loss over the training set $\mathcal{D} = \{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ within the hypothesis space $\mathcal{H}$.

Linear model

$$\mathcal{H}^{(d)} := \{h : \mathbb{R}^d \rightarrow \mathbb{R} \mid h(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} \text{ for some } \mathbf{w} \in \mathbb{R}^d\} \quad (2)$$

Entries: `linmodel`. Source: Hastie et al. (2009), Sect. 3.2 [2].

Least squares objective of linear regression

$$\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{m} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2, \quad \mathbf{X} = (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)})^\top, \quad \mathbf{y} = (y^{(1)}, \dots, y^{(m)})^\top \quad (3)$$

Entries: `linreg`, `featuremtx`. Source: Hastie et al. (2009), Sect. 3.2 [2]. Hastie et al. write the residual sum of squares without the factor $1/m$, which does not change the minimizer.

Gradient of the least squares objective

$$\nabla f(\mathbf{w}) = -\frac{2}{m} \sum_{r=1}^m (y^{(r)} - \mathbf{w}^\top \mathbf{x}^{(r)}) \mathbf{x}^{(r)} = \frac{2}{m} \mathbf{X}^\top (\mathbf{X}\mathbf{w} - \mathbf{y}) \quad (4)$$

Entries: `gradient`, `condnr`. Source: Hastie et al. (2009), Sect. 3.2, Eq. (3.4) [2].

```
m, d = 50, 4
X, y = rng.standard_normal((m, d)), rng.standard_normal(m)
w = rng.standard_normal(d)
f = lambda w: np.sum((y - X @ w) ** 2) / m
grad = (2 / m) * X.T @ (X @ w - y)
h = 1e-6
numeric = np.array([(f(w + h * e) - f(w - h * e)) / (2 * h) for e in np.eye(d)])
assert np.allclose(grad, numeric, atol=1e-6)
```

Normal equations

$$\mathbf{X}^\top \mathbf{X} \hat{\mathbf{w}} = \mathbf{X}^\top \mathbf{y} \quad (5)$$

Entries: `normalequations`, `linreg`. Source: Hastie et al. (2009), Sect. 3.2 [2]. A unique solution exists exactly when $\mathbf{X}$ has full column rank.

```

m, d = 50, 4
X, y = rng.standard_normal((m, d)), rng.standard_normal(m)
w_hat = np.linalg.lstsq(X, y, rcond=None)[0]
assert np.allclose(X.T @ X @ w_hat, X.T @ y)

```

Least squares predictions as a projection

$$\mathbf{X}\hat{\mathbf{w}} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} \quad (6)$$

Entries: `featuremtx`, `linreg`. Source: Hastie et al. (2009), Sect. 3.2, Eq. (3.7) [2]. The right-hand side is the orthogonal projection of $\mathbf{y}$ onto the column space of $\mathbf{X}$; it is unchanged by any invertible transformation $\mathbf{X} \mapsto \mathbf{X}\mathbf{A}^\top$ of the features.

```

m, d = 50, 4
X, y = rng.standard_normal((m, d)), rng.standard_normal(m)
P = X @ np.linalg.inv(X.T @ X) @ X.T
assert np.allclose(P @ y, X @ np.linalg.lstsq(X, y, rcond=None)[0])
A = rng.standard_normal((d, d)) # an invertible feature map
assert np.allclose((X @ A.T) @ np.linalg.lstsq(X @ A.T, y, rcond=None)[0], P @ y)

```

Ridge regression in closed form

$$\hat{\mathbf{w}}^{(\alpha)} = (\mathbf{X}^\top \mathbf{X} + \alpha m \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y} \quad (7)$$

Entries: `ridgeregression`. Source: Hastie et al. (2009), Sect. 3.4.1, Eq. (3.44) [2]. The minimizer of $\frac{1}{m} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \alpha \|\mathbf{w}\|_2^2$; Hastie et al. write $\alpha \mathbf{I}$ for the objective without the factor $1/m$.

```

m, d, alpha = 50, 4, 0.3
X, y = rng.standard_normal((m, d)), rng.standard_normal(m)
w = np.linalg.solve(X.T @ X + alpha * m * np.eye(d), X.T @ y)
grad = (2 / m) * X.T @ (X @ w - y) + 2 * alpha * w # zero at the minimizer
assert np.allclose(grad, 0)

```

Kernel ridge regression in closed form

$$\hat{\boldsymbol{\beta}} = (\mathbf{K} + \alpha m \mathbf{I})^{-1} \mathbf{y}, \quad \hat{h}(\mathbf{x}) = \sum_{r=1}^m \hat{\beta}_r k(\mathbf{x}^{(r)}, \mathbf{x}) \quad (8)$$

Entries: `kernelridgeregression`. Source: Bishop (2006), Sect. 6.1, Eqs. (6.8) and (6.9) [3]. $\mathbf{K}$ is the Gram matrix with entries $k(\mathbf{x}^{(r)}, \mathbf{x}^{(r')})$; Bishop writes $\alpha \mathbf{I}$ for the unaveraged objective.

```

m, alpha = 30, 0.2
X, y = rng.standard_normal((m, 3)), rng.standard_normal(m)
K = np.exp(-np.sum((X[:, None] - X[None, :]) ** 2, -1)) # a Gaussian kernel
beta = np.linalg.solve(K + alpha * m * np.eye(m), y)
grad = (2 / m) * K @ (K @ beta - y) + 2 * alpha * K @ beta
assert np.allclose(grad, 0)

```

Conditioning and gradient descent

Gradient descent step

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla f(\mathbf{w}^{(t)}) \quad (9)$$

Entries: `gd`, `gradstep`. Source: Boyd and Vandenberghe (2004), Sect. 9.3 [4].

Gradient descent with exact line search on a strongly convex objective

$$f(\mathbf{w}^{(T)}) - f^* \leq \left(1 - \frac{m'}{M'}\right)^T (f(\mathbf{w}^{(0)}) - f^*), \quad m' \mathbf{I} \preceq \nabla^2 f \preceq M' \mathbf{I} \quad (10)$$

Entries: `gd`, `featuremtx`, `condnr`. Source: Boyd and Vandenberghe (2004), Sect. 9.3.1 [4]. The number of steps that reaches a given gap grows in proportion to M'/m' , the condition number of the Hessian.

Condition number of the normal equations

$$\kappa(\mathbf{X}^\top \mathbf{X}) = \kappa(\mathbf{X})^2, \quad \kappa(\mathbf{X}) = \frac{\lambda_1}{\lambda_{\text{rank}(\mathbf{X})}} \quad (11)$$

Entries: `featuremtx`, `condnr`. Source: Golub and Loan (2013), Sect. 5.3.2 [5]. $\lambda_1 \geq \lambda_2 \geq \dots$ are the singular values of $\mathbf{X}$. The least squares objective has Hessian $(2/m)\mathbf{X}^\top \mathbf{X}$, so M'/m' in the previous line equals $\kappa(\mathbf{X})^2$.

```
X = rng.standard_normal((50, 4)) @ np.diag([1, 3, 9, 27])
assert np.isclose(np.linalg.cond(X.T @ X), np.linalg.cond(X) ** 2)
s = np.linalg.svd(X, compute_uv=False)
assert np.isclose(np.linalg.cond(X), s[0] / s[-1])
```

Exact rate of gradient descent on a quadratic, slowest start

$$f(\mathbf{w}^{(t+1)}) - f^* = \left(\frac{q-1}{q+1}\right)^2 (f(\mathbf{w}^{(t)}) - f^*), \quad q = \kappa(\mathbf{X})^2 \quad (12)$$

Entries: `condnr`, `featuremtx`. Source: Boyd and Vandenberghe (2004), Sect. 9.3.2 [4]. Exact line search, started where the error has, along each eigenvector of the Hessian, a component inversely proportional to the eigenvalue. Verified on real data in `pythondemos/condnr.py`: 731 steps against 730 predicted.

```
X = rng.standard_normal((50, 2)) @ np.diag([1.0, 5.0])
y = rng.standard_normal(50)
f = lambda w: np.sum((X @ w - y) ** 2) / 50
w_hat = np.linalg.lstsq(X, y, rcond=None)[0]
mu, V = np.linalg.eigh(X.T @ X)
w = w_hat + V @ (1 / mu) # the slowest start
q = np.linalg.cond(X) ** 2
gaps = []
for _ in range(5):
    gaps.append(f(w) - f(w_hat))
    g = X.T @ (X @ w - y); Xg = X @ g
    w = w - (Xg @ (X @ w - y)) / (Xg @ Xg) * g # exact line search
ratios = np.array(gaps[1:]) / np.array(gaps[:-1])
assert np.allclose(ratios, ((q - 1) / (q + 1)) ** 2)
```

Sensitivity of the least squares solution

$$\|\tilde{\mathbf{w}} - \hat{\mathbf{w}}\| \lesssim \epsilon (\kappa(\mathbf{X}) + \kappa(\mathbf{X})^2) \|\hat{\mathbf{w}}\| \quad (13)$$

Entries: `featuremtx`. Source: Golub and Loan (2013), Sect. 5.3.6 [5]. $\mathbf{X}$ of full column rank; the perturbation of $\mathbf{X}$ has largest singular value at most $\epsilon\lambda_1$ and that of $\mathbf{y}$ Euclidean norm at most $\epsilon\|\mathbf{y}\|$. First order in ϵ , and a heuristic form of Theorem 5.3.1 there, valid unless $\mathbf{y}$ is nearly orthogonal to the column space of $\mathbf{X}$.

Loss functions for binary classification

Hinge loss

$$L((\mathbf{x}, y), h) := \max\{0, 1 - y \cdot h(\mathbf{x})\}, \quad y \in \{-1, +1\} \quad (14)$$

Entries: `hingeloss`. Source: Shalev-Shwartz and Ben-David (2014), Sect. 12.3 [1]. A convex upper bound on the 0/1 loss.

Logistic loss

$$L((\mathbf{x}, y), h) := \ln(1 + \exp(-y \cdot h(\mathbf{x}))), \quad y \in \{-1, +1\} \quad (15)$$

Entries: `logloss`, `logreg`. Source: Bishop (2006), Sect. 4.3.2, Eq. (4.90) [3]. Bishop writes the same loss as the cross-entropy error for labels in $\{0, 1\}$.

Margin losses are compared as functions of the margin

$$L((\mathbf{x}, y), h) = \phi(y \cdot h(\mathbf{x})), \quad \phi \text{ nonincreasing} \quad (16)$$

Entries: `accuracy`, `hingeloss`, `logloss`. Source: Hastie et al. (2009), Sect. 10.6 [2]. The 0/1, hinge and logistic losses are all of this form; a surrogate should penalize negative margins (confident wrong classification) more heavily than it rewards positive ones.

Singular values and covariance

Singular value decomposition

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{U}^\top, \quad \mathbf{V}^\top \mathbf{V} = \mathbf{I}, \quad \mathbf{U}^\top \mathbf{U} = \mathbf{I}, \quad \Lambda_{j,j} = \lambda_j \geq 0 \quad (17)$$

Entries: `svd`. Source: Golub and Loan (2013), Sect. 2.4 [5].

Rank from the singular values

$$\text{rank}(\mathbf{A}) = |\{j : \lambda_j > 0\}| \quad (18)$$

Entries: `featuremtx`, `svd`. Source: Golub and Loan (2013), Cor. 2.4.6 [5].

```
X = rng.standard_normal((50, 3)) @ rng.standard_normal((3, 6)) # rank 3
s = np.linalg.svd(X, compute_uv=False)
assert np.sum(s > 1e-10 * s[0]) == np.linalg.matrix_rank(X) == 3
```

Best rank- k approximation

$$\hat{\mathbf{A}} := \sum_{j=1}^k \lambda_j \mathbf{v}^{(j)} (\mathbf{u}^{(j)})^\top \in \arg \min_{\text{rank}(\mathbf{B}) \leq k} \|\mathbf{A} - \mathbf{B}\|_F \quad (19)$$

Entries: `svd`. Source: Eckart and Young (1936) [6].

```
A = rng.standard_normal((20, 12)); k = 4
V, s, Ut = np.linalg.svd(A, full_matrices=False)
A_k = (V[:, :k] * s[:k]) @ Ut[:k, :]
best = np.sqrt(np.sum(s[k:] ** 2)) # the error of the SVD truncation
for _ in range(20): # no other rank-k matrix does better
    B = rng.standard_normal((20, k)) @ rng.standard_normal((k, 12))
    assert np.linalg.norm(A - B) >= best - 1e-12
assert np.isclose(np.linalg.norm(A - A_k), best)
```

Sample covariance matrix

$$\hat{\mathbf{C}} = \frac{1}{m} \sum_{r=1}^m (\mathbf{x}^{(r)} - \hat{\boldsymbol{\mu}})(\mathbf{x}^{(r)} - \hat{\boldsymbol{\mu}})^\top \quad (20)$$

Entries: `samplecovmtx`, `featuremtx`. Source: Bishop (2006), Sect. 12.1.1, Eq. (12.3) [3]. For features with zero sample mean, $\hat{\mathbf{C}} = \mathbf{X}^\top \mathbf{X} / m$, whose eigenvalues are the squared singular values of $\mathbf{X}$ divided by m .

```
X = rng.standard_normal((50, 4)) @ rng.standard_normal((4, 4))
Xc = X - X.mean(0)
S = Xc.T @ Xc / 50
assert np.allclose(S, np.cov(X, rowvar=False, bias=True))
assert np.allclose(np.sort(np.linalg.eigvalsh(S)), np.sort(np.linalg.svd(Xc, compute_uv=False) ** 2 / 50))
```

Principal component analysis: the dropped eigenvalues

$$\min_{\mathbf{R} \in \mathbb{R}^{d \times d'}} \frac{1}{m} \sum_{r=1}^m \left\| \mathbf{x}^{(r)} - \mathbf{R} \mathbf{W} \mathbf{x}^{(r)} \right\|_2^2 = \sum_{j=d'+1}^d \lambda_j \quad (21)$$

Entries: `pca`, `featuremtx`. Source: Bishop (2006), Sect. 12.1.2, Eq. (12.46) [3]. $\lambda_1 \geq \dots \geq \lambda_d$ here are the eigenvalues of the sample covariance matrix; the minimum keeps the d' leading eigenvectors.

```
X = rng.standard_normal((200, 5)) @ rng.standard_normal((5, 5)); X -= X.mean(0)
S = X.T @ X / 200
lam, U = np.linalg.eigh(S); lam, U = lam[::-1], U[:, ::-1]
k = 2; W = U[:, :k].T; R = U[:, :k]
err = np.mean(np.sum((X - X @ W.T @ R.T) ** 2, 1))
assert np.isclose(err, lam[k:].sum())
```

Concentration inequalities

Markov's inequality

$$\mathbb{P}(x \geq a) \leq \frac{\mathbb{E}\{x\}}{a} \quad \text{for a nonnegative random variable } x \text{ and any } a > 0 \quad (22)$$

Entries: `markovsinequality`. Source: Folland (1999), p. 50 [7].

```
a = np.linspace(0.5, 20, 100)
tail = np.exp(-a)
assert np.all(tail <= 1 / a) # P(x >= a) for an exponential x, E{x} = 1
```

Hoeffding's inequality

$$\mathbb{P}(|S_n - \mathbb{E}\{S_n\}| \geq t) \leq 2 \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right) \quad \text{for all } t > 0 \quad (23)$$

Entries: `hoeffdingsinequality`. Source: Hoeffding (1963) [8]. $S_n = \sum_{i=1}^n x_i$ for independent random variables x_i with $a_i \leq x_i \leq b_i$.

References

1. S. Shalev-Shwartz and S. Ben-David (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge Univ. Press, New York, NY, USA.
2. T. Hastie, R. Tibshirani and J. Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed., Springer Science+Business Media, New York, NY, USA.
3. C. M. Bishop (2006). *Pattern Recognition and Machine Learning*. Springer Science+Business Media, New York, NY, USA.
4. S. Boyd and L. Vandenberghe (2004). *Convex Optimization*. Cambridge Univ. Press, Cambridge, U.K.
5. G. H. Golub and C. F. Van Loan (2013). *Matrix Computations*. 4th ed., The Johns Hopkins Univ. Press, Baltimore, MD, USA.
6. C. Eckart and G. Young (1936). *The approximation of one matrix by another of lower rank*. Psychometrika, 1(3), pp. 211–218.
7. Gerald B. Folland (1999). *Real Analysis: Modern Techniques and Their Applications*. 2nd ed., Wiley, New York, NY, USA.
8. W. Hoeffding (1963). *Probability Inequalities for Sums of Bounded Random Variables*. J. Amer. Statistical Assoc., 58(301), pp. 13–30.