

Machine Learning

Textbook: Machine Learning Refined, by Watt, Borhani, and Katsaggelos, 2nd edition:
MLRefined.com

Previous Lecture

- ◆ The logistic sigmoid maps any real-valued score into the interval $(0, 1)$.
- ◆ Logistic regression models $P(Y = +1 \mid \mathbf{x}) = \sigma(f(\mathbf{x}; \mathbf{w}))$.
- ◆ A probability threshold of $1/2$ gives the same linear decision boundary $f(\mathbf{x}; \mathbf{w}) = 0$.
- ◆ The linear score equals the log-odds: $\log[p/(1-p)] = f(\mathbf{x}; \mathbf{w})$.
- ◆ Maximum likelihood with binary labels produces the smooth logistic loss $\log(1 + e^{-y f})$.
- ◆ Logistic loss depends on the signed margin and can be optimized with gradient methods.



§6.2, 6.7 - Cross-Entropy

Last time:

logistic regression produced

$$p = P(Y = 1 \mid \mathbf{x}) = \sigma(f).$$

Maximum likelihood gave a logarithmic loss.

Today:

write that loss in standard probability language,
understand why it is called **cross-entropy**,
and derive its gradient.

Use 0/1 Labels

For probability calculations,

encode the classes as

$$y \in \{0, 1\}.$$

Let

$$p = P(Y = 1 \mid \mathbf{x}) = \sigma(f).$$

Then

$$P(Y = 0 \mid \mathbf{x}) = 1 - p.$$

The model predicts an entire Bernoulli distribution:

Class	0	1
Predicted probability	$1 - p$	p

One Formula for Either Observed Class

For

$$y \in \{0, 1\},$$

the probability assigned to the observed label is

$$P(Y = y \mid \mathbf{x}) = p^y(1 - p)^{1-y}.$$

Check:

if $y = 1$:

$$p^1(1 - p)^0 = p.$$

If $y = 0$:

$$p^0(1 - p)^1 = 1 - p.$$

This is the Bernoulli likelihood for one observation.

Negative Log-Likelihood

Take the negative logarithm:

$$-\log[p^y(1 - p)^{1-y}].$$

Using

$$\log(a^b) = b \log a,$$

we obtain

$$\ell(y, p) = -[y \log p + (1 - y) \log(1 - p)].$$

This is the **binary cross-entropy loss**.

It is also called

binary log loss.

What Happens for Each Label?

If

$y = 1$:

$$\ell(1, p) = -\log p.$$

If

$y = 0$:

$$\ell(0, p) = -\log(1 - p).$$

So:

give the observed class high probability

$\Rightarrow$

small loss.

Give the observed class low probability

$\Rightarrow$

large loss.

Confidently Wrong Predictions Are Expensive

Suppose
 $y = 1$.

Then

p	0.99	0.90	0.50	0.10	0.01
$-\log p$	0.010	0.105	0.693	2.303	4.605

As
 $p \rightarrow 0$,

the loss satisfies
 $-\log p \rightarrow \infty$.

A confident wrong probability is penalized strongly.

Correct but Uncertain Versus Correct and Confident

Again let
 $y = 1$.

Prediction
 $p = 0.51$
is technically on the correct side of a 0.5 threshold.

Its loss:
 $-\log(0.51) \approx 0.673$.

Prediction
 $p = 0.99$
has the same predicted class.

But its loss:
 $-\log(0.99) \approx 0.010$.

Cross-entropy evaluates probabilities,
not just hard classifications.

Why Is It Called Cross-Entropy?

A probability distribution over the two classes can be written as
 $\mathbf{q} = (q_0, q_1)$.

Suppose the model predicts
 $\mathbf{p} = (p_0, p_1)$.

The cross-entropy of $\mathbf{q}$ relative to $\mathbf{p}$ is
 $H(\mathbf{q}, \mathbf{p}) = -\sum_k q_k \log p_k$.

For binary classification:
 $\mathbf{p} = (1 - p, p)$.

For an observed label y :

$$\mathbf{q} = (1 - y, y).$$

Therefore

$$H(\mathbf{q}, \mathbf{p}) = -[(1 - y) \log(1 - p) + y \log p] .$$

Exactly our binary log loss.

The Observed Distribution Is One-Hot

If

$$y = 1,$$

then

$$\mathbf{q} = (0, 1).$$

Cross-entropy:

$$\begin{aligned} H(\mathbf{q}, \mathbf{p}) &= -[0 \log(1 - p) + 1 \log p] \\ &= -\log p. \end{aligned}$$

If

$$y = 0,$$

then

$$\mathbf{q} = (1, 0).$$

Cross-entropy:

$$H(\mathbf{q}, \mathbf{p}) = -\log(1 - p).$$

Only the probability assigned to the observed class enters the loss.

Expected Cross-Entropy

Now suppose the true probability of class 1
at some fixed input is

$$q.$$

The model predicts

$$p.$$

The expected cross-entropy is

$$C(p) = -[q \log p + (1 - q) \log(1 - p)].$$

Question:

which predicted probability p
minimizes this expected loss?

Cross-Entropy Is Minimized by the True Probability

Differentiate:

$$C'(p) = -\frac{q}{p} + \frac{1-q}{1-p}.$$

Combine:

$$C'(p) = \frac{p-q}{p(1-p)}.$$

Set equal to zero:

$$p - q = 0.$$

Thus

$$p^* = q.$$

So expected cross-entropy is minimized
when the predicted probability equals the true probability.

This is why cross-entropy is appropriate
for learning probabilities,
not merely decision boundaries.

Verify It Is a Minimum

Differentiate again:

$$C''(p) = \frac{q}{p^2} + \frac{1-q}{(1-p)^2}.$$

For

$$0 < p < 1,$$

we have

$$C''(p) > 0.$$

Therefore

$$p = q$$

is the unique minimum

whenever

$$0 < q < 1.$$

The loss rewards calibrated probability estimates.

Return to Logistic Regression

The logistic model uses

$$p = \sigma(f),$$

where

$$f = \mathbf{\tilde{x}}^T \mathbf{w}.$$

Cross-entropy becomes

$$\begin{aligned} \ell(y, f) \\ = -[y \log \sigma(f) + (1 - y) \log(1 - \sigma(f))]. \end{aligned}$$

This looks complicated.

Its derivative is remarkably simple.

Differentiate With Respect to the Probability

Start with

$$\ell(y,p) = -[y \log p + (1 - y) \log(1 - p)].$$

Then

$$\begin{aligned} \frac{\partial \ell}{\partial p} &= -\frac{y}{p} + \frac{1-y}{1-p} \\ &= \frac{p-y}{p(1-p)}. \end{aligned}$$

Now use

$$p = \sigma(f).$$

From Lesson 1.15:

$$\frac{dp}{df} = p(1 - p).$$

The Cancellation

By the chain rule:

$$\begin{aligned} \frac{\partial \ell}{\partial f} &= \frac{\partial \ell}{\partial p} \frac{dp}{df} \\ &= \frac{p-y}{p(1-p)} \cdot p(1 - p). \end{aligned}$$

Therefore

$$\frac{\partial \ell}{\partial f} = p - y.$$

One of the most useful simplifications in machine learning:

$$\begin{aligned} &\textbf{sigmoid} + \textbf{cross-entropy} \\ \Rightarrow & \\ &\textbf{prediction minus label}. \end{aligned}$$

Gradient With Respect to the Parameters

Because

$$f = \tilde{\mathbf{x}}^T \mathbf{w},$$

we have

$$\nabla_{\mathbf{w}} f = \tilde{\mathbf{x}}.$$

Thus for one observation:

$$\nabla_{\mathbf{w}} \ell = (p - y) \tilde{\mathbf{x}}.$$

For P observations:

$$\nabla g(\mathbf{w}) = \frac{1}{P} \sum_{p=1}^P (p_p - y_p) \tilde{\mathbf{x}}_p.$$

Matrix form:

$$\nabla g(\mathbf{w}) = \frac{1}{P} X^T (\mathbf{p} - \mathbf{y}).$$

Compare with least squares:

$$\frac{2}{P} X^T (\hat{\mathbf{y}} - \mathbf{y}).$$

The structures are strikingly similar.

A Small Gradient Example

Take

$$x = (-1, 0, 1),$$

$$y = (0, 1, 1).$$

Use the augmented design matrix

$$X = \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}.$$

Start at

$$\mathbf{w} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Then every score is 0,

so every probability is

$$p_p = 0.5.$$

Thus

$$\mathbf{p} - \mathbf{y} = \begin{bmatrix} 0.5 \\ -0.5 \\ -0.5 \end{bmatrix}.$$

Compute the Gradient

Use

$$\nabla g = \frac{1}{3} X^T (\mathbf{p} - \mathbf{y}).$$

Then

$$\begin{aligned} & X^T (\mathbf{p} - \mathbf{y}) \\ &= \begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.5 \\ -0.5 \\ -0.5 \end{bmatrix} \\ &= \begin{bmatrix} -0.5 \\ -1 \end{bmatrix}. \end{aligned}$$

Therefore

$$\nabla g = \begin{bmatrix} -1/6 \\ -1/3 \end{bmatrix}.$$

Gradient descent moves in the opposite direction:
both intercept and slope increase.

That raises predicted probabilities
for the two observations labeled 1.

The Hessian

For logistic regression:

$$\nabla g = \frac{1}{P} X^T (\mathbf{p} - \mathbf{y}).$$

Differentiating again gives

$$H = \frac{1}{P} X^T R X,$$

where

$$R = \text{diag}(p_1(1 - p_1), \dots, p_P(1 - p_P)).$$

Each diagonal entry satisfies

$$p_p(1 - p_p) \geq 0.$$

Why Logistic Cross-Entropy Is Convex

For any vector $\mathbf{v}$:

$$\begin{aligned} & \mathbf{v}^T H \mathbf{v} \\ &= \frac{1}{P} \mathbf{v}^T X^T R X \mathbf{v} \\ &= \frac{1}{P} (X \mathbf{v})^T R (X \mathbf{v}) \\ &= \frac{1}{P} \sum_{p=1}^P p_p(1 - p_p) (\tilde{\mathbf{x}}_p^T \mathbf{v})^2 \\ &\geq 0. \end{aligned}$$

Therefore

$$H \succeq 0.$$

So logistic cross-entropy is a **convex** function
of the linear-model parameters.

There are no spurious local minima.

For perfectly separable data,
the finite minimum issue from Lesson 1.15 still remains.

Cross-Entropy and Logistic Loss Are the Same Loss

Lesson 1.15 used labels

$$s \in \{-1, +1\}$$

and logistic loss

$$\log(1 + e^{-sf}).$$

Today we use

$$y \in \{0, 1\}.$$

Relate the labels by

$$y = \frac{s+1}{2}.$$

If

$s = +1:$
 $y = 1,$
and cross-entropy is
 $-\log \sigma(f) = \log(1 + e^{-f}).$

If

$s = -1:$
 $y = 0,$
and cross-entropy is
 $-\log(1 - \sigma(f)) = -\log \sigma(-f)$
 $= \log(1 + e^f).$

Both cases are

$$\log(1 + e^{-sf}).$$

Different notation.

Same objective.

A Stable Formula From the Logit

Directly computing

$$-\log \sigma(f)$$

or

$$-\log(1 - \sigma(f))$$

can be numerically troublesome

when probabilities are extremely close to 0 or 1.

Using the score f directly:

$$\ell(y, f) = \log(1 + e^f) - yf.$$

Check:

if $y = 0,$
 $\ell = \log(1 + e^f).$

If $y = 1,$

$$\ell = \log(1 + e^f) - f = \log(1 + e^{-f}).$$

This form leads to stable software implementations.

Cross-Entropy in Python

Define the sigmoid:

```
def sigmoid(t):  
    return 1 / (1 + np.exp(-t))
```

A direct probability version:

```
def cross_entropy(y, p):  
    return -np.mean(  
        y*np.log(p)  
        + (1-y)*np.log(1-p)  
    )
```

Conceptually clear.

But:

probabilities extremely close to 0 or 1
can produce numerical problems.

Stable Cross-Entropy From Scores

Use

$$\ell(y,f) = \log(1 + e^f) - yf.$$

In NumPy:

```
def cross_entropy_from_scores(w, X, y):  
    scores = X @ w  
  
    losses = np.logaddexp(0, scores) \  
        - y*scores  
  
    return np.mean(losses)
```

Recall:

$\log(1 + e^f)$
computes
 $\log(1 + e^f)$
stably.

This avoids taking logarithms of rounded probabilities.

The Gradient in Python

```
def grad_cross_entropy(w, X, y):  
    scores = X @ w  
    p = sigmoid(scores)  
  
    return X.T @ (p - y) / len(y)
```

This code is almost a direct transcription of

$$\nabla g = \frac{1}{P} X^T (\mathbf{p} - \mathbf{y}).$$

No explicit loop over observations is required.

Matrix notation becomes executable code.

One Gradient-Descent Step in Python

For the small example:

```

X = np.array([[1., -1.],
              [1.,  0.],
              [1.,  1.]])

y = np.array([0., 1., 1.])

w = np.array([0., 0.])
alpha = 0.3

grad = grad_cross_entropy(w, X, y)
w = w - alpha*grad

print(grad)
print(w)

```

Output:

```

[-0.16666667 -0.33333333]
[ 0.05         0.10        ]

```

Exactly the hand calculation.

Why Accuracy Is Not Enough for Training Probabilities

Suppose one positive observation
 has true label
 $y = 1$.

Two models predict:

	Model A	Model B
p	0.51	0.99
Class at threshold .5	1	1
Accuracy on this point	correct	correct
Cross-entropy	.673	.010

Accuracy treats them identically.

Cross-entropy distinguishes
 weak and strong probability predictions.

That makes it suitable for fitting probabilistic classifiers.

Why Cross-Entropy Is Not the Same as Calibration

Minimizing expected cross-entropy
 encourages correct probabilities.

But a trained model may still be poorly calibrated
 because of:
 limited data;
 restricted model form;

regularization;

distribution shift;

imperfect optimization.

Cross-entropy is the training criterion.

Calibration is an empirical property
that must be checked on data.

The Main Chain of Ideas

Linear score:

$$f = \tilde{\mathbf{x}}^T \mathbf{w}.$$

Sigmoid:

$$p = \sigma(f).$$

Bernoulli probability:

$$p^y(1-p)^{1-y}.$$

Negative log-likelihood:

$$-[y \log p + (1-y) \log(1-p)].$$

Gradient:

$$(p-y)\tilde{\mathbf{x}}.$$

Everything follows from one probabilistic model.

What did we learn?

- ♦ Binary cross-entropy is the negative log-likelihood of a Bernoulli prediction.
- ♦ For $y \in \{0, 1\}$, $\ell(y, p) = -[y \log p + (1-y) \log(1-p)]$.
- ♦ Cross-entropy is minimized in expectation when the predicted probability equals the true probability.
- ♦ Combining the sigmoid with cross-entropy gives the simple score derivative $\partial \ell / \partial f = p - y$.
- ♦ For logistic regression, $\nabla g = (1/P) X^T (\mathbf{p} - \mathbf{y})$.
- ♦ Binary cross-entropy with 0/1 labels is exactly the same objective as logistic loss with $-1/+1$ labels.



Next

- Logistic regression is one smooth way to learn a linear classifier.
- Other classification losses emphasize the decision boundary and margin more directly.
- Next: perceptron and support-vector-machine ideas.
- **1.17 - Perceptrons and Support Vector Machines**