Linear Classifiers 2

Christian Hardmeier

2018-04-10

CiML chapter 5 in a hurry

- Do read chapter 5 if you haven't done so already.
- If you have any numerical features
  that are not always between −1 and 1,
  don't forget to try feature scaling.

Recap: The perceptron

- Given a linearly separable training set,
  the perceptron algorithm efficiently finds
  a separating hyperplane.
- It starts with an initial guess,
  predicts each training example in turn and
  updates its guess whenever it makes an error.
- Perceptron update:

$$\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$$
$$b \leftarrow b + y$$

- This week: A more general framework
  for training linear classifiers.

Learning as optimisation

- The goal of the training procedure is to find a set of parameters that
  - performs well on the training set, and
  - also works well for unseen test data (generalisation).
- We can define an objective function addressing these two problems and optimise it directly.

Objective function

$$f(\mathbf{w}, b; \mathcal{T}) = \text{loss}(\mathbf{w}, b; \mathcal{T}) + \lambda R(\mathbf{w}, b)$$

$\text{loss}(\mathbf{w}, b; \mathcal{T})$	measures training loss given parameters
$R(\mathbf{w}, b)$	controls model complexity
λ	controls the balance between the two

Optimisation problem

$$\begin{aligned}\hat{\mathbf{w}}, \hat{b} &= \arg\min_{\mathbf{w}, b} f(\mathbf{w}, b; \mathcal{T}) \\ &= \arg\min_{\mathbf{w}, b} \text{loss}(\mathbf{w}, b; \mathcal{T}) + \lambda R(\mathbf{w}, b)\end{aligned}$$

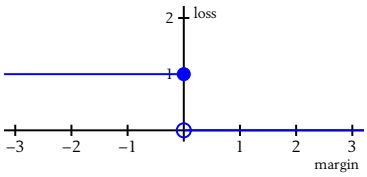
Find the parameters $\hat{\mathbf{w}}, \hat{b}$ for which the objective function f achieves its minimum on the training set.

Note

- When we introduced the perceptron, we started with an algorithm and proceeded to demonstrate that it had some desirable properties.
- We now start by stating what properties our model should have and proceed to derive a method for achieving this.

0-1 loss for classification

The simplest loss function for classification is 0-1 loss.



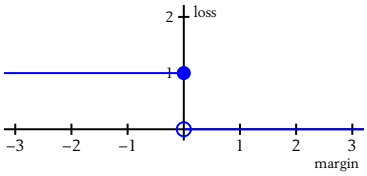
$$\text{loss}(\mathbf{w}, b; \mathcal{T}) = \begin{cases} 1 & \text{if } y\hat{y} \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

Why do we use 0-1 loss?

- "Default" loss function: Makes no assumptions about the problem
- Error rate is the average 0-1 loss of a data set. Accuracy is 1 – Error rate.

Problems of optimising 0-1 loss

- Optimising 0-1 loss is essentially a discrete optimisation problem.
- In general, it is computationally intractable.



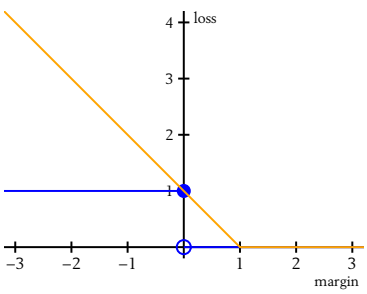
So what do we want our loss functions to look like?

- A good loss function should be an upper bound on the 0-1 loss.
  - This makes it likely that we reduce 0-1 loss when we minimise the surrogate loss.
- It should be continuous (no jumps).
  - If a loss function is discontinuous, you can't use gradient-based optimisation techniques.
  - Optimisation becomes much harder.
- Ideally, it should be convex.
  - Convexity guarantees that there is a single global minimum.
  - If your loss isn't convex, you may end up with a suboptimal minimum.

Surrogate loss functions

Hinge loss

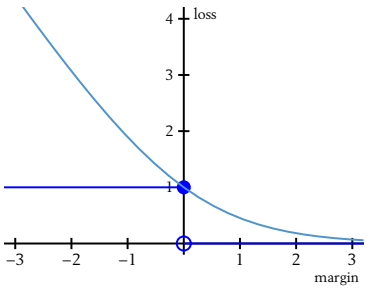
$$\text{loss}(\mathbf{w}, b; \mathcal{T}) = \max(0, 1 - y\hat{y})$$



Surrogate loss functions

Logistic loss

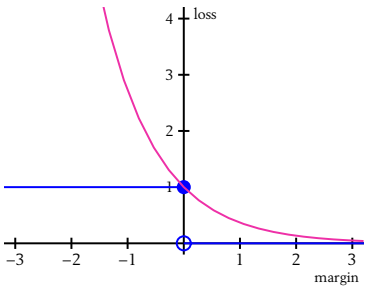
$$\text{loss}(\mathbf{w}, b; \mathcal{T}) = \frac{1}{\log 2} \log(1 + \exp(-y\hat{y}))$$



Surrogate loss functions

Exponential loss

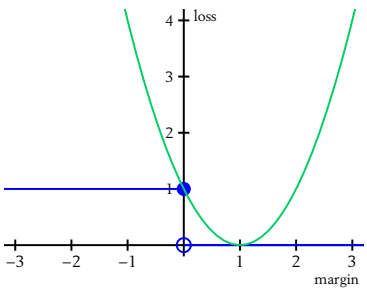
$$\text{loss}(\mathbf{w}, b; \mathcal{T}) = \exp(-y\hat{y})$$



Surrogate loss functions

Squared loss

$$\text{loss}(\mathbf{w}, b; \mathcal{T}) = (y - \hat{y})^2$$



Properties of different loss functions

- Recall that the margin is the distance of a point from the decision boundary.
- The margin is positive if the point is correctly classified, negative if not.
- Discuss the properties of your loss function:
  - How does it behave for correctly classified instances?
  - How does it behave for incorrectly classified instances?
  - How does it compare to 0-1 loss?
  - How does it compare to the other alternatives?

Loss minimisation and linear separability

- In the perceptron algorithm, linear separability was a prerequisite for convergence.
- The loss minimisation framework doesn't *require* separability.
- If the training set isn't separable, the training loss may never reach zero.
- Optimisation still finds the best approximation.

Overfitting

extremely bad .	never once predictable .
shrewd but pointless .	warm and exotic .
a non-mystery mystery .	a true pleasure .
what an embarrassment .	sexy and romantic .
a noble failure .	psychologically savvy .
woefully pretentious .	unflinchingly bleak and desperate
a relative letdown .	delightfully rendered
warmed-over hash .	fun and nimble .

Words occurring only once will be very strongly associated with the positive or negative class.

Regularisation

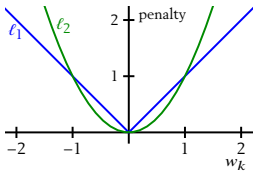
Recall our objective function:

$$f(\mathbf{w}, b; \mathcal{T}) = \text{loss}(\mathbf{w}, b; \mathcal{T}) + \lambda R(\mathbf{w}, b)$$

- The regularisation term $R(\mathbf{w}, b)$ counteracts overfitting by *constraining the capacity of the model*.
- It creates an incentive to create a *simpler* model, even at the cost of underfitting the training set.
- The variable λ controls the strength of this incentive.

ℓ_1 and ℓ_2 regularisation

- The most common regularisation methods impose a penalty on the length of the weight vectors.
- Feature weights need more evidence to grow.
- Most important variants:
  - ℓ_1 regularisation: $R(\mathbf{w}, b) = ||\mathbf{w}||_1 = \sum_k |w_k|$.
  - ℓ_2 regularisation: $R(\mathbf{w}, b) = ||\mathbf{w}||_2^2 = \sum_k w_k^2$.
- Bias is often left unregularised.



Is regularisation necessary?

Let $\mathcal{T} = \{\langle\langle 0, 0\rangle, -1\rangle, \langle\langle 1, 0\rangle, -1\rangle, \langle\langle 0, 1\rangle, -1\rangle, \langle\langle 1, 1\rangle, 1\rangle\}$.

Let L be the unregularised exponential loss:

$$L = \sum_{i=1}^{|\mathcal{T}|} \exp(-y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b))$$
$$= \exp b + \exp(w_1 + b) + \exp(w_2 + b) + \exp(-w_1 - w_2 - b)$$

Now

$$\frac{\partial L}{\partial w_1} = \exp(w_1 + b) - \exp(-w_1 - w_2 - b)$$

Is regularisation necessary?

Let $\mathcal{T} = \{\langle\langle 0, 0, 1\rangle, -1\rangle, \langle\langle 1, 0, 0\rangle, -1\rangle, \langle\langle 0, 1, 0\rangle, -1\rangle, \langle\langle 1, 1, 0\rangle, 1\rangle\}$.

Let L be the unregularised exponential loss:

$$\begin{aligned} L &= \sum_{i=1}^{|\mathcal{T}|} \exp(-y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b)) \\ &= \exp(w_3 + b) + \exp(w_1 + b) + \exp(w_2 + b) + \exp(-w_1 - w_2 - b) \end{aligned}$$

Now

$$\frac{\partial L}{\partial w_3} = \exp(w_3 + b) > 0$$

Is regularisation necessary?

Let $\mathcal{T} = \{\langle\langle 0, 0, 1\rangle, -1\rangle, \langle\langle 1, 0, 0\rangle, -1\rangle, \langle\langle 0, 1, 0\rangle, -1\rangle, \langle\langle 1, 1, 0\rangle, 1\rangle\}$.

Let L be the ℓ_2-regularised exponential loss:

$$\begin{aligned} L &= \sum_{i=1}^{|\mathcal{T}|} \exp(-y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b)) + ||\mathbf{w}||^2 \\ &= \exp(w_3 + b) + \exp(w_1 + b) + \exp(w_2 + b) + \exp(-w_1 - w_2 - b) \\ &\quad + w_1^2 + w_2^2 + w_3^2 \end{aligned}$$

Now

$$\frac{\partial L}{\partial w_3} = \exp(w_3 + b) + 2w_3$$

When does this happen?

- In an NLP task with lexical features (bag of words or similar), some features will always be very rare.
- The problem arises when a rare feature only occurs in one class of examples (positive or negative).
- Such features are often pure noise, but will affect unregularised loss and make the optimisation problem diverge.
- You need regularisation to deal with them.

How to perform the optimisation

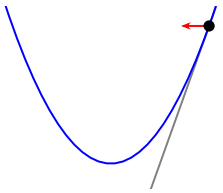
- Here is our optimisation problem:

$$\begin{aligned}\hat{\mathbf{w}}, \hat{b} &= \arg\min_{\mathbf{w},b} f(\mathbf{w}, b; \mathcal{T}) \\ &= \arg\min_{\mathbf{w},b} \text{loss}(\mathbf{w}, b; \mathcal{T}) + \lambda R(\mathbf{w}, b)\end{aligned}$$

- Most optimisation problems in machine learning are solved with iterative numerical methods.
- Gradient descent is one of the simplest and most common.

Gradient descent: Principle

- Start with a guess.
- Find the direction of steepest descent at that point by computing the gradient of the objective function, $\nabla f(\mathbf{w}, b; \mathcal{T})$.
- Take a step in that direction.
- Iterate until you find a minimum.



Gradient descent: Step

- In each iteration, the gradient descent algorithm makes an adjustment to the current guess:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla f(\mathbf{w}, b; \mathcal{T})$$

- The step size depends on the magnitude of the gradient and on the learning rate η.

How to compute gradients

- The gradient is the vector of the partial derivatives of the objective function (regularised training loss) with respect to all input variables (weights and bias).
- It can be calculated for the loss function and the regulariser separately.

ℓ_2 regulariser

$$R(\mathbf{w}, b) = ||\mathbf{w}||_2^2 = \sum_k w_k^2$$

Partial derivatives:

$$\frac{\partial R}{\partial w_k} = 2w_k$$

ℓ_1 regulariser

$$R(\mathbf{w}, b) = ||\mathbf{w}||_1 = \sum_k |w_k|$$

Partial derivatives:

$$\frac{\partial R}{\partial w_k} = \begin{cases} \text{sign}(w_k) & \text{if } w_k \neq 0 \\ \text{undefined} & \text{if } w_k = 0 \end{cases}$$

Hinge loss

$$L(\mathbf{w}, b; \mathcal{T}) = \sum_{i=1}^{|\mathcal{T}|} \begin{cases} 1 - y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b) & \text{if } y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b) < 1 \\ 0 & \text{otherwise} \end{cases}$$

Partial derivatives:

$$\frac{\partial L}{\partial w_k} = \sum_{i=1}^{|\mathcal{T}|} \begin{cases} -y^{(i)} x_k^{(i)} & \text{if } y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b) < 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{|\mathcal{T}|} \begin{cases} -y^{(i)} & \text{if } y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b) < 1 \\ 0 & \text{otherwise} \end{cases}$$

Logistic loss

$$L(\mathbf{w}, b; \mathcal{T}) = \frac{1}{\log 2} \sum_{i=1}^{|\mathcal{T}|} \log\left(1 + \exp\left(-y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b)\right)\right)$$

Partial derivatives:

$$\frac{\partial L}{\partial w_k} = \frac{1}{\log 2} \sum_{i=1}^{|\mathcal{T}|} -y^{(i)} x_k^{(i)} \frac{\exp(-y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b))}{1 + \exp(-y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b))}$$

Next up

- Assignment 1: due 13/4.
- Assignment 2: already on-line (due 27/4).
- Thu 12/4: no meeting.
- Tue 17/4, 14–16: Lab session in Chomsky.
  Please study the assignment before coming to the lab.
- Thu 19/4, 14–16: Lecture in 16-0042.
  Reading: CiML chapter 9