

MSE 546: Advanced Machine Learning

Sirisha Rambhatla

University of Waterloo

Lecture
Logistic Regression

Outline

- 1 Machine Learning
- 2 Classification
- 3 Logistic regression
- 4 Reading

Outline

- 1 Machine Learning
- 2 Classification
- 3 Logistic regression
- 4 Reading

Recap: What do we need for Machine Learning?

Recap: Key ingredients in machine learning

- Data
collected from past observation (we often call them *training data*)
- Modeling
designed to capture the patterns in the data
 - The model does not have to be true — “All models are wrong, but some are useful” by George Box.
- Prediction
apply the model to forecast what is going to happen in future

Recap: What is in machine learning?

Different flavors of learning problems

- Supervised learning
Aims to predict (as in previous examples)
- Unsupervised learning
Aims to discover hidden and latent patterns and explore data
- Decision making (e.g. reinforcement learning)
Aims to act optimally under uncertainty
- Many other paradigms

Outline

- 1 Machine Learning
- 2 Classification**
- 3 Logistic regression
- 4 Reading

Classification

The setup:

- input (feature vector): $\mathbf{x} \in \mathbb{R}^D$
- output (label): $y \in [C] = \{1, 2, \dots, C\}$
- goal: learn a mapping $f : \mathbb{R}^D \rightarrow [C]$

This lecture: **binary classification using Logistic Regression**

- Number of classes: $C = 2$
- Labels: $\{-1, +1\}$ (cat or dog, fraud or not, price up or down...)

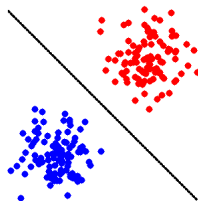
Deriving classification algorithms

Step 1. Pick a set of models \mathcal{F} .

Predict a label using the *Sign* of $w^\top x$:

$$\text{sign}(w^\top x) = \begin{cases} +1 & \text{if } w^\top x > 0 \\ -1 & \text{if } w^\top x \leq 0 \end{cases}$$

(Sometimes use sgn for sign .)



Step 1: The models

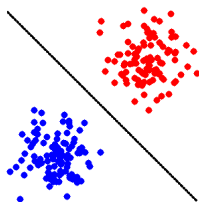
The set of **(separating) hyperplanes**:

$$\mathcal{F} = \{f(x) = \text{sgn}(w^\top x) \mid w \in \mathbb{R}^D\}$$

Good choice for *linearly separable* data, i.e., $\exists w$ s.t.

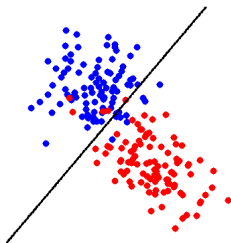
$$\text{sgn}(w^\top x_n) = y_n \quad \text{or} \quad y_n w^\top x_n > 0$$

for all $n \in [N]$.



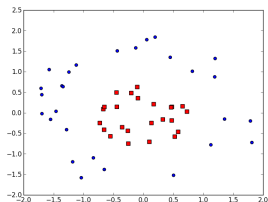
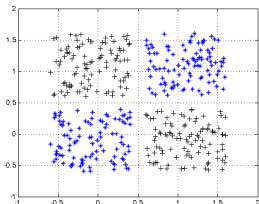
Step 1: The models

Still makes sense for “almost” linearly separable data



The models

For clearly not linearly separable data,



We can apply a nonlinear transformation to the features!

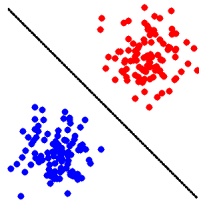
(Beyond the scope for now)

Step 2: Define a loss

Step 2. Define error/loss $L(y', y)$.

- *Loss function defines how errors will be counted.*
- For classification, more convenient to look at the loss **as a function of $yw^\top x$** :

$$yw^\top x = \begin{cases} \geq 0 & \text{if correct} \\ -1 & \text{if wrong} \end{cases}$$



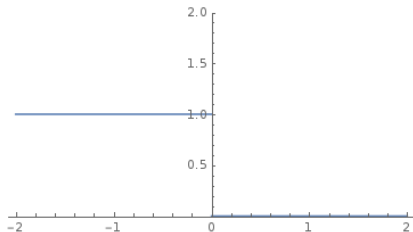
An Example for Step 2 (Define a loss): 0-1 Loss

Step 2 (Example): Most natural one for classification is the **0-1 loss**:

$$L(y', y) = \mathbb{I}[y' \neq y]$$

That is, with $z = y\mathbf{w}^\top \mathbf{x}$, this can also be represented as

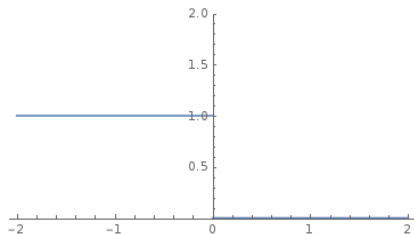
$$\ell_{0-1}(z) = \mathbb{I}[z \leq 0]$$



the loss for hyperplane \mathbf{w} on an example (\mathbf{x}, y) is $\ell_{0-1}(y\mathbf{w}^\top \mathbf{x})$

Problem: Minimizing 0-1 loss is hard

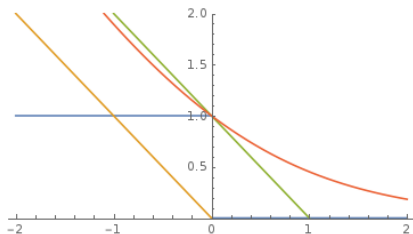
However, 0-1 loss is *not convex*.



Even worse, minimizing 0-1 loss is *NP-hard in general*.

Solution: Convex Surrogate Losses!

Solution: find a **convex surrogate loss**



- **perceptron loss** $\ell_{\text{perceptron}}(z) = \max\{0, -z\}$ (used in Perceptron)
- **hinge loss** $\ell_{\text{hinge}}(z) = \max\{0, 1 - z\}$ (used in SVM and many others)
- **logistic loss** $\ell_{\text{logistic}}(z) = \log(1 + \exp(-z))$ (used in logistic regression; the base of \log doesn't matter, in this case it is 2) (in red)

ML becomes convex optimization!

Step 3. Find ERM:

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w} \in \mathbb{R}^D} \sum_{n=1}^N \ell(y_n \mathbf{w}^\top \mathbf{x}_n) = \operatorname{argmin}_{\mathbf{w} \in \mathbb{R}^D} \frac{1}{N} \sum_{n=1}^N \ell(y_n \mathbf{w}^\top \mathbf{x}_n)$$

where $\ell(\cdot)$ can be a convex loss such as **logistic loss**

- *no closed-form* in general
- can apply general convex optimization methods

Outline

- 1 Machine Learning
- 2 Classification
- 3 Logistic regression**
- 4 Reading

A simple view

In one sentence: find the minimizer of

$$\begin{aligned} F(\mathbf{w}) &= \frac{1}{N} \sum_{n=1}^N \ell_{\text{logistic}}(y_n \mathbf{w}^\top \mathbf{x}_n) \\ &= \frac{1}{N} \sum_{n=1}^N \ln(1 + e^{-y_n \mathbf{w}^\top \mathbf{x}_n}) \end{aligned}$$

But why logistic loss? and why “regression”?

Predicting probability

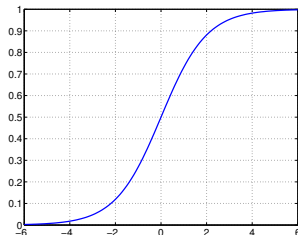
Instead of predicting a discrete label, can we *predict the probability of each label?* i.e. **regress** the probabilities

One way: **sigmoid function + linear model**

$$\mathbb{P}(y = +1 \mid \mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^\top \mathbf{x})$$

where σ is the sigmoid function:

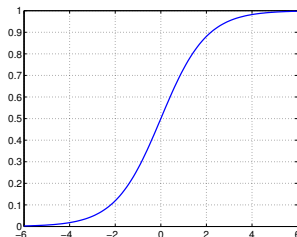
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



Properties

Properties of sigmoid $\sigma(z) = \frac{1}{1+e^{-z}}$

- between 0 and 1 (as probability)
- $\sigma(\mathbf{w}^\top \mathbf{x}) \geq 0.5 \Leftrightarrow \mathbf{w}^\top \mathbf{x} \geq 0$, consistent with predicting the label with $\text{sgn}(\mathbf{w}^\top \mathbf{x})$
- larger $\mathbf{w}^\top \mathbf{x} \Rightarrow$ larger $\sigma(\mathbf{w}^\top \mathbf{x}) \Rightarrow$ higher *confidence* in label 1
- $\sigma(z) + \sigma(-z) = 1$ for all z



The probability of label -1 is naturally

$$1 - \mathbb{P}(y = +1 \mid \mathbf{x}; \mathbf{w}) = 1 - \sigma(\mathbf{w}^\top \mathbf{x}) = \sigma(-\mathbf{w}^\top \mathbf{x})$$

and thus

$$\mathbb{P}(y \mid \mathbf{x}; \mathbf{w}) = \sigma(y\mathbf{w}^\top \mathbf{x}) = \frac{1}{1 + e^{-y\mathbf{w}^\top \mathbf{x}}}$$

How to regress with discrete labels?

What we observe are labels, not probabilities.

Take a **probabilistic view**

- assume data is generated in this way by some w
- perform Maximum Likelihood Estimation (MLE)

Specifically, what is the probability of seeing label y_1, \dots, y_n given x_1, \dots, x_n , as a function of some w ?

$$P(w) = \prod_{n=1}^N \mathbb{P}(y_n \mid x_n; w)$$

MLE: find w^* that **maximizes the probability** $P(w)$

The MLE solution

$$\begin{aligned}\mathbf{w}^* &= \operatorname{argmax}_{\mathbf{w}} P(\mathbf{w}) = \operatorname{argmax}_{\mathbf{w}} \prod_{n=1}^N \mathbb{P}(y_n \mid \mathbf{x}_n; \mathbf{w}) \\ &= \operatorname{argmax}_{\mathbf{w}} \sum_{n=1}^N \ln \mathbb{P}(y_n \mid \mathbf{x}_n; \mathbf{w}) = \operatorname{argmin}_{\mathbf{w}} \sum_{n=1}^N -\ln \mathbb{P}(y_n \mid \mathbf{x}_n; \mathbf{w}) \\ &= \operatorname{argmin}_{\mathbf{w}} \sum_{n=1}^N \ln(1 + e^{-y_n \mathbf{w}^\top \mathbf{x}_n}) = \operatorname{argmin}_{\mathbf{w}} \sum_{n=1}^N \ell_{\text{logistic}}(y_n \mathbf{w}^\top \mathbf{x}_n) \\ &= \operatorname{argmin}_{\mathbf{w}} F(\mathbf{w})\end{aligned}$$

i.e. *minimizing logistic loss is exactly doing MLE for the sigmoid model*

Outline

- 1 Machine Learning
- 2 Classification
- 3 Logistic regression
- 4 Reading**

Reading

PiML1

- Chapter 10 – 10.1, 10.2 (intro and 10.2.1, 10.2.3, 10.2.4)