Linear classification

COMS 4771 Fall 2019

Overview

- Logistic regression model
- Linear classifiers
- Gradient descent and SGD
- Multinomial logistic regression model

Logistic regression model I

- Suppose $\boldsymbol{x}$ is given by d real-valued features, so $\boldsymbol{x} \in \mathbb{R}^d$, while $y \in \{-1, +1\}$.
- *Logistic regression model* for $(\boldsymbol{X}, Y)$:
  - $Y \mid \boldsymbol{X} = \boldsymbol{x}$ is Bernoulli (but taking values in $\{-1, +1\}$ rather than $\{0, 1\}$), with "heads probability" parameter

$$\frac{1}{1 + \exp(-\boldsymbol{x}^\top \boldsymbol{w})}.$$

  - $\boldsymbol{w} \in \mathbb{R}^d$ is parameter vector of interest
  - $\boldsymbol{w}$ not involved in marginal distribution of $\boldsymbol{X}$ (which we don't care much about)

Logistic regression model II

- *Sigmoid function* $\sigma(t) := 1/(1 + e^{-t})$
  - Useful property: $1 - \sigma(t) = \sigma(-t)$
  - $\Pr(Y = +1 \mid \boldsymbol{X} = \boldsymbol{x}) = \sigma(\boldsymbol{x}^\top \boldsymbol{w})$
  - $\Pr(Y = -1 \mid \boldsymbol{X} = \boldsymbol{x}) = 1 - \sigma(\boldsymbol{x}^\top \boldsymbol{w}) = \sigma(-\boldsymbol{x}^\top \boldsymbol{w})$
- Convenient formula: for each $y \in \{-1, +1\}$,

$$\Pr(Y = y \mid \boldsymbol{X} = \boldsymbol{x}) = \sigma(y \boldsymbol{x}^\top \boldsymbol{w}).$$

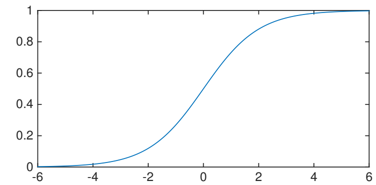


Figure 1: Sigmoid function

Log-odds in logistic regression model

- Log-odds in the model is given by a linear function:

$$\ln \frac{\Pr(Y = +1 \mid \boldsymbol{X} = \boldsymbol{x})}{\Pr(Y = -1 \mid \boldsymbol{X} = \boldsymbol{x})} = \boldsymbol{x}^{\mathsf{T}}\boldsymbol{w}.$$

- Just like in linear regression, common to use feature expansion!
  - E.g., affine feature expansion $\varphi(\boldsymbol{x}) = (1, \boldsymbol{x}) \in \mathbb{R}^{d+1}$

Optimal classifier in logistic regression model

- Recall that *Bayes classifier* is

$$f^\star(x) = \begin{cases} +1 & \text{if } \Pr(Y = +1 \mid X = x) > 1/2 \\ -1 & \text{otherwise.} \end{cases}$$

- If distribution of $(\boldsymbol{X}, Y)$ comes from logistic regression model with parameter $\boldsymbol{w}$, then Bayes classifier is

$$f^\star(\boldsymbol{x}) = \begin{cases} +1 & \text{if } \boldsymbol{x}^{\mathsf{T}}\boldsymbol{w} > 0 \\ -1 & \text{otherwise.} \end{cases}$$
$$= \operatorname{sign}(\boldsymbol{x}^{\mathsf{T}}\boldsymbol{w}).$$

- This is a *linear classifier*
- Many other statistical models for classification data lead to a linear (or affine) classifier, e.g., Naive Bayes

Linear classifiers, operationally

- Compute linear combination of features, then check if above threshold (zero)

$$\operatorname{sign}(\boldsymbol{x}^{\mathsf{T}}\boldsymbol{w}) = \begin{cases} +1 & \text{if } \sum_{i=1}^{d} w_i x_i > 0 \\ -1 & \text{otherwise.} \end{cases}$$

- With affine feature expansion, threshold can be non-zero

if $0.335 \cdot x_1 \; + \; 2.5 \cdot x_2 \; + \; \cdots \; + \; 6.35 \cdot x_{10^6} \; > \; 4.3$ **then**
    return spam
else
    return not spam
end if

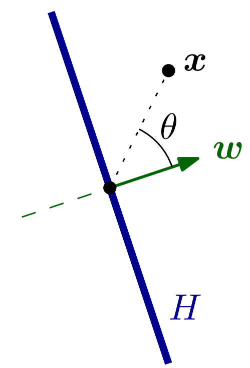
Figure 2: Example of an affine classifier

Geometry of linear classifiers I

- *Hyperplane* specified by *normal vector* $\boldsymbol{w} \in \mathbb{R}^d$:
  - $H = \{\boldsymbol{x} \in \mathbb{R}^d : \boldsymbol{x}^{\mathsf{T}}\boldsymbol{w} = 0\}$
  - This is the *decision boundary* of a linear classifier
  - Angle θ between $\boldsymbol{x}$ and $\boldsymbol{w}$ has

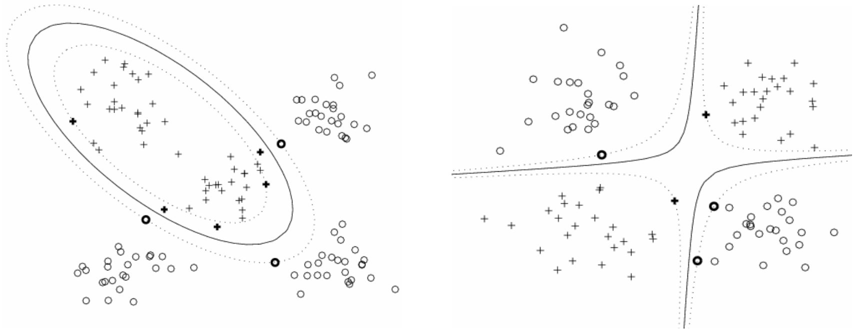
$$\cos(\theta) = \frac{\boldsymbol{x}^{\mathsf{T}}\boldsymbol{w}}{\|\boldsymbol{x}\|_2 \|\boldsymbol{w}\|_2}$$

  - Distance to hyperplane given by $\|\boldsymbol{x}\|_2 \cdot \cos(\theta)$
  - $\boldsymbol{x}$ is on same side of H as $\boldsymbol{w}$ iff $\boldsymbol{x}^{\mathsf{T}}\boldsymbol{w} > 0$



Geometry of linear classifiers II

- With feature expansion, can obtain other types of decision boundaries



MLE for logistic regression

- Treat training examples as iid, same distribution as test example
- Log-likelihood of $\boldsymbol{w}$ given data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n) \in \mathbb{R}^d \times \{-1, +1\}$:

$$-\sum_{i=1}^{n} \ln(1 + \exp(-y_i \boldsymbol{x}_i^{\mathsf{T}} \boldsymbol{w})) + \{\text{terms not involving } \boldsymbol{w}\}$$

- No "closed form" expression for maximizer
- (Later, we'll discuss algorithms for finding approximate maximizers using iterative methods like gradient descent.)
- What about ERM perspective?

Zero-one loss and ERM for linear classifiers

- Recall: error rate of classifier f can also be written as risk:

$$\mathcal{R}(f) = \mathbb{E}[\mathbb{1}_{\{f(X) \neq Y\}}] = \Pr(f(X) \neq Y),$$

  where loss function is zero-one loss.
- For classification, we are ultimately interested in classifiers with small error rate
  - I.e., small (zero-one loss) risk
- Just like for linear regression, can apply plug-in principle to derive *empirical risk minimization (ERM)*, but now for linear classifiers.
  - Find $\boldsymbol{w} \in \mathbb{R}^d$ to minimize

$$\widehat{\mathcal{R}}(\boldsymbol{w}) := \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{\operatorname{sign}(\boldsymbol{x}_i^{\mathsf{T}} \boldsymbol{w}) \neq y_i\}}.$$

- Very different from MLE for logistic regression

ERM for linear classifiers

- **Theorem**: In IID model, ERM solution $\hat{\boldsymbol{w}}$ satisfies

$$\mathbb{E}[\mathcal{R}(\hat{\boldsymbol{w}})] \leq \min_{\boldsymbol{w} \in \mathbb{R}^d} \mathcal{R}(\boldsymbol{w}) + O\left(\sqrt{\frac{d}{n}}\right).$$

- Unfortunately, solving this optimization problem, even for linear classifiers, is computationally intractable.
  - (Sharp contrast to ERM optimization problem for linear regression!)

Linearly separable data I

- Training data is *linearly separable* if there exists a linear classifier with training error rate zero.
  - (Special case where ERM optimization problem is tractable.)
  - There exists $\boldsymbol{w} \in \mathbb{R}^d$ such that $\operatorname{sign}(\boldsymbol{x}_i^\top \boldsymbol{w}) = y_i$ for all $i = 1, \ldots, n$.
  - Equivalent:

$$y_i \boldsymbol{x}_i^\top \boldsymbol{w} > 0 \quad \text{for all } i = 1, \ldots, n$$

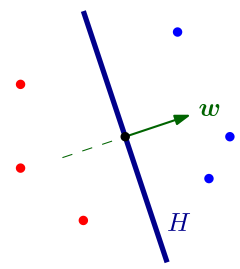


Figure 4: Linearly separable

Linearly separable data II

- Training data is *linearly separable* if there exists a linear classifier with training error rate zero.
  - (Special case where ERM optimization problem is tractable.)
  - There exists $\boldsymbol{w} \in \mathbb{R}^d$ such that $\operatorname{sign}(\boldsymbol{x}_i^\top \boldsymbol{w}) = y_i$ for all $i = 1, \ldots, n$.
  - Equivalent:

$$y_i \boldsymbol{x}_i^\top \boldsymbol{w} > 0 \quad \text{for all } i = 1, \ldots, n$$

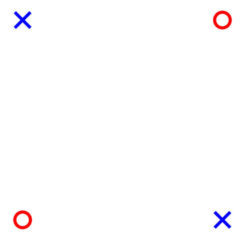


Figure 5: Not linearly separable

Finding a linear separator I

- Suppose training data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n) \in \mathbb{R}^d \times \{-1, +1\}$ is linearly separable.
- How to find a linear separator (assuming one exists)?
- Method 1: solve linear feasibility problem

Finding a linear separator II

- Method 2: (approximately) solve logistic regression MLE

Surrogate loss functions I

- Often, a linear separator will not exist.
- Regard each term in negative log-likelihood as a "loss"

$$\ell_{\log}(s) := \ln(1 + \exp(-s))$$

- C.f. Zero-one loss:

$$\ell_{\text{zo}}(s) := \mathbb{1}_{\{s \le 0\}}$$

- $\ell_{\log}$ (up to scaling) is upper-bound on ℓ_{zo}: a *surrogate loss*:

$$\ell_{\text{zo}}(s) \le \frac{1}{\ln 2}\,\ell_{\log}(s) = \ell_{\log_2}(s).$$

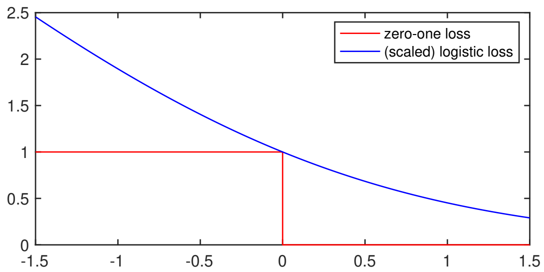


Figure 6: Logistic loss vs zero-one loss

Surrogate loss functions II

- Another example: squared loss
  - $\ell_{\text{sq}}(s) = (1 - s)^2$

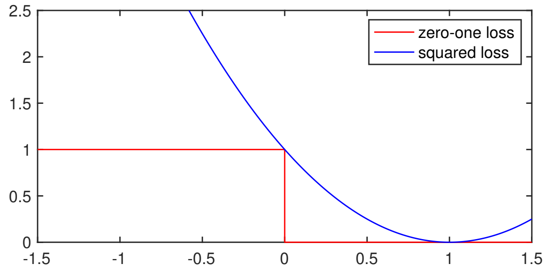


Figure 7: Squared loss vs zero-one loss

Surrogate loss functions III

- Modified squared loss:
  - $\ell_{\text{msq}}(s) := \max\{0, 1 - s\}^2$.

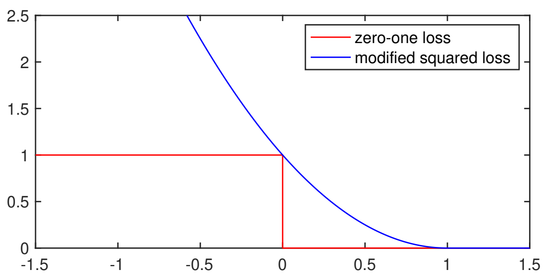


Figure 8: Modified squared loss vs zero-one loss

Gradient descent for logistic regression

- No "closed form" expression for logistic regression MLE
- Instead, use iterative algorithms like gradient descent.
- Gradient of empirical risk: by linearity of derivative operator,

$$\nabla \widehat{\mathcal{R}}_{\ell_{\log}}(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} \nabla\, \ell_{\log}(y_i \boldsymbol{x}_i^{\mathsf{T}} \boldsymbol{w}).$$

- Algorithm: start with some $\boldsymbol{w}^{(0)} \in \mathbb{R}^d$ and $\eta > 0$.
  - For $t = 1, 2, \ldots$:

$$\boldsymbol{w}^{(t)} := \boldsymbol{w}^{(t-1)} - \eta \nabla \widehat{\mathcal{R}}_{\ell_{\log}}(\boldsymbol{w}^{(t-1)})$$

Gradient of logistic loss

- Gradient of logistic loss on i-th training example: using chain rule,

$$\begin{aligned}\nabla \ell_{\log}(y_i \boldsymbol{x}_i^{\mathsf{T}} \boldsymbol{w}) &= \ell'_{\log}(y_i \boldsymbol{x}_i^{\mathsf{T}} \boldsymbol{w}) y_i \boldsymbol{x}_i \\ &= -\left(1 - \frac{1}{1+\exp(-y_i \boldsymbol{x}_i^{\mathsf{T}} \boldsymbol{w})}\right) y_i \boldsymbol{x}_i \\ &= -\left(1 - \Pr_{\boldsymbol{w}}(Y = y_i \mid \boldsymbol{X} = \boldsymbol{x}_i)\right) y_i \boldsymbol{x}_i\end{aligned}$$

Here, $\Pr_{\boldsymbol{w}}$ is probability distribution of $(\boldsymbol{X}, Y)$ from logistic regression model with parameter $\boldsymbol{w}$.

Interpretation of gradient descent for logistic regression

- Interpretation of gradient descent:

$$\nabla \ell_{\log}(y_i \boldsymbol{x}_i^{\mathsf{T}} \boldsymbol{w}) = -\left(1 - \Pr_{\boldsymbol{w}}(Y = y_i \mid \boldsymbol{X} = \boldsymbol{x}_i)\right) y_i \boldsymbol{x}_i$$

  - Trying to make $\Pr_{\boldsymbol{w}}(Y = y_i \mid \boldsymbol{X} = \boldsymbol{x}_i)$ as close to 1 as possible.
  - (Achieved by making $\boldsymbol{w}$ infinitely far in direction of $y_i \boldsymbol{x}_i$.)
  - How much of $y_i \boldsymbol{x}_i$ to add to $\boldsymbol{w}$ is scaled by how far the $\Pr_{\boldsymbol{w}}(\cdots)$ currently is from 1.

Behavior of gradient descent for logistic regression

- Analysis of gradient descent for logistic regression MLE much more complicated than for linear regression
  - Solution could be at infinity!
- **Theorem**: for appropriate choice of step size $\eta > 0$,

$$\widehat{\mathcal{R}}(\boldsymbol{w}^{(t)}) \to \inf_{\boldsymbol{w} \in \mathbb{R}^d} \widehat{\mathcal{R}}(\boldsymbol{w})$$

as $t \to \infty$ (even if the infimum is never attained).

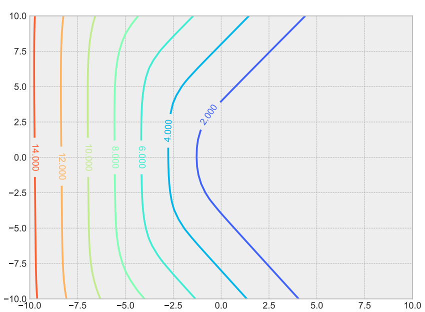


Figure 9: Empirical logistic loss risk

Stochastic gradient method I

- Every iteration of gradient descent takes $\Theta(nd)$ time.
  - Pass through all training examples to make a single update.
  - If n is enormous, too expensive to make many passes.
- Alternative: *Stochastic gradient descent (SGD)*
  - Use one or a few training examples to estimate the gradient.
  - Gradient at $\boldsymbol{w}^{(t)}$:

$$\frac{1}{n} \sum_{j=1}^{n} \nabla \ell(y_j \boldsymbol{x}_j^{\mathsf{T}} \boldsymbol{w}^{(t)}).$$

    (Called *full batch gradient*.)
  - Pick term J uniformly at random:

$$\nabla \ell(y_J \boldsymbol{x}_J^{\mathsf{T}} \boldsymbol{w}^{(t)}).$$

  - What is expected value of this random vector?

$$\mathbb{E}\left[\nabla \ell(y_J \boldsymbol{x}_J^{\mathsf{T}} \boldsymbol{w}^{(t)})\right] =$$

Stochastic gradient method II

- Minibatch
  - To reduce variance of estimate, use several random examples $J_1, \ldots, J_B$ and average—called *minibatch gradient*.

$$\frac{1}{B} \sum_{b=1}^{B} \nabla \ell(y_{J_b} \boldsymbol{x}_{J_b}^{\mathsf{T}} \boldsymbol{w}^{(t)}).$$

  - Rule of thumb: larger batch size $B \to$ larger step size η.
- Alternative: instead of picking example uniformly at random, shuffle order of training examples, and take next example in this order.
  - Verify that expected value is same!
  - Seems to reduce variance as well, but not fully understood.

Example: SGD for logistic regression

- Logistic regression MLE for data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n) \in \mathbb{R}^d \times \{-1, +1\}$.
- Start with $\boldsymbol{w}^{(0)} \in \mathbb{R}^d$, $\eta > 0$, $t = 1$
- For epoch $p = 1, 2, \ldots$:
  - For each training example $(\boldsymbol{x}, y)$ in a random order:

$$\boldsymbol{w}^{(t)} := \boldsymbol{w}^{(t-1)} + \eta \left(1 - \frac{1}{1 + \exp(-y\boldsymbol{x}^{\mathsf{T}} \boldsymbol{w}^{(t-1)})}\right) y\boldsymbol{x}; \quad t := t + 1.$$

- (If $\boldsymbol{w}^{(0)} = \boldsymbol{0}$, then final solution is in span of $\boldsymbol{x}_i$.)

Multinomial logistic regression

- How to handle $K > 2$ classes?
- *Multinomial logistic regression model*
  - $Y \mid \boldsymbol{X} = \boldsymbol{x}$ has a categorical distribution over $\{1, \ldots, K\}$

$$\Pr(Y = k \mid \boldsymbol{X} = \boldsymbol{x}) = \operatorname{softmax}(\boldsymbol{W}\boldsymbol{x})_k$$

  - *Softmax function* (vector-valued function):

$$\operatorname{softmax}(\boldsymbol{v})_k := \frac{\exp(v_k)}{\sum_{l=1}^{K} \exp(v_l)}$$

  - $\boldsymbol{W} = [\boldsymbol{w}_1 | \cdots | \boldsymbol{w}_K]^{\mathsf{T}} \in \mathbb{R}^{K \times d}$ is parameter matrix of interest

MLE for multinomial logistic regression

- Treat training examples as iid, same distribution as test example
- Encode label y_i as a *one-hot* vector $\tilde{\boldsymbol{y}}_i \in \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_K\}$
  - $\tilde{\boldsymbol{y}}_i = (\tilde{y}_{i,1}, \ldots, \tilde{y}_{i,K})$, where $\tilde{y}_{i,k} = \mathbb{1}_{\{y_i = k\}}$
- MLE equivalent to minimizing empirical risk with *cross-entropy loss*

$$\frac{1}{n} \sum_{i=1}^{n} \ell_{\text{ce}}(\tilde{\boldsymbol{y}}_i, \operatorname{softmax}(\boldsymbol{W}\boldsymbol{x}_i))$$

where $\ell_{\text{ce}}(\boldsymbol{p}, \boldsymbol{q}) = -\sum_{k=1}^{K} p_k \ln q_k$.