

Text Classification
+
Machine Learning Review 2

CS 287

Quiz: Naive Bayes

Given bag-of-word features

$$\mathcal{F} = \{\text{The, movie, was, terrible, rocked, A}\}$$

and two training data points:

Class 1: The movie was terrible

Class 2: The movie rocked

What is the conditional distribution $P(Y|X)$ of the new example “terrible movie” with $\alpha = 0$?

What about “A terrible movie” with $\alpha = 1$?

Answer: Naive Bayes (1)

terrible movie

Prior is simple: $p(\mathbf{y}) = [1/2 \quad 1/2]$

Construct count matrix:

$$\mathbf{F} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

With $\alpha = \epsilon$,

$$p(\mathbf{x}|\mathbf{y}) = \begin{bmatrix} 1/4 & 1/4 & 1/4 & 1/4 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \end{bmatrix}$$

$$p(\mathbf{y}|\mathbf{x}) \propto [1/2 \times 1/2 \times 1/2 \quad 1/2 \times 0 \times 1/3] = [1 \quad 0]$$

Answer: Naive Bayes (2)

A terrible movie

With $\alpha = 1$,

$$\bar{\mathbf{F}} = \begin{bmatrix} 2 & 2 & 2 & 2 & 1 & 1 \\ 2 & 2 & 1 & 1 & 2 & 1 \end{bmatrix}$$

$$p(\mathbf{x}|\mathbf{y}) = \begin{bmatrix} 1/5 & 1/5 & 1/5 & 1/5 & 1/10 & 1/10 \\ 2/9 & 2/9 & 1/9 & 1/9 & 2/9 & 1/9 \end{bmatrix}$$

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}) &\propto \begin{bmatrix} 1/2 \times 1/10 \times 1/5 \times 1/5 & 1/2 \times 1/9 \times 1/9 \times 2/9 \\ 0.593 & 0.407 \end{bmatrix} \\ &\approx \begin{bmatrix} 0.593 & 0.407 \end{bmatrix} \end{aligned}$$

Contents

Classification Review

Discriminative Models

Loss

Optimization

Review: Multiclass Sentiment

- ▶ ★★★

I visited The Abbey on several occasions on a visit to Cambridge and found it to be a solid, reliable and friendly place for a meal.

- ▶ **

However, the food leaves something to be desired. A very obvious menu and average execution

- ▶ ★★★★★

Fun, friendly neighborhood bar. Good drinks, good food, not too pricey. Great atmosphere!

Review: Sparse Bag-of-Words Features

Representation is counts of input words,

- ▶ \mathcal{F} ; the vocabulary of the language.
- ▶ $\mathbf{x} = \sum_i \delta(f_i)$

Example: Movie review input,

A sentimental mess

$$\mathbf{x} = \delta(\text{word:A}) + \delta(\text{word:sentimental}) + \delta(\text{word:mess})$$

$$\mathbf{x}^\top = \begin{bmatrix} 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix} \begin{array}{l} \text{word:A} \\ \vdots \\ \text{word:mess} \\ \text{word:sentimental} \end{array}$$

Review: Output Class Notation

- ▶ $\mathcal{C} = \{1, \dots, d_{\text{out}}\}$; possible output classes
- ▶ $c \in \mathcal{C}$; always one true output class
- ▶ $\mathbf{y} = \delta(c) \in \mathbb{R}^{1 \times d_{\text{in}}}$; true one-hot output representation

Review: Multiclass Classification

Examples: Yelp stars, etc.

- ▶ $d_{\text{out}} = 5$; for examples
- ▶ In our notation, one star, two star...

$$\begin{aligned} \star c = 1 \quad \mathbf{y} &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix} \text{ vs.} \\ \star\star c = 2 \quad \mathbf{y} &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix} \dots \end{aligned}$$

Examples: Word Prediction (Unit 3)

- ▶ $d_{\text{out}} > 100,000$;
- ▶ In our notation, \mathcal{C} is vocabulary and each c is a word.

$$\begin{aligned} \textit{the} c = 1 \quad \mathbf{y} &= \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 \end{bmatrix} \text{ vs.} \\ \textit{dog} c = 2 \quad \mathbf{y} &= \begin{bmatrix} 0 & 1 & 0 & 0 & \dots & 0 \end{bmatrix} \dots \end{aligned}$$

Review Linear Models for Classification

Linear model,

$$\hat{\mathbf{y}} = f(\mathbf{x}\mathbf{W} + \mathbf{b})$$

- ▶ $\mathbf{W} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$, $\mathbf{b} \in \mathbb{R}^{1 \times d_{\text{out}}}$; model parameters
- ▶ $f : \mathbb{R}^{d_{\text{out}}} \mapsto \mathbb{R}^{d_{\text{out}}}$; activation function
- ▶ Sometimes $\mathbf{z} = \mathbf{x}\mathbf{W} + \mathbf{b}$ informally “score” vector.
- ▶ Note \mathbf{z} and $\hat{\mathbf{y}}$ are not one-hot.

Class prediction,

$$\hat{c} = \arg \max_{i \in \mathcal{C}} \hat{y}_i = \arg \max_{i \in \mathcal{C}} (\mathbf{x}\mathbf{W} + \mathbf{b})_i$$

Probabilistic Linear Models

Can estimate a linear model probabilistically,

- ▶ Let output be a random variable Y , with sample space \mathcal{C} .
- ▶ Representation be a random vector X .
- ▶ (Simplified frequentist representation)
- ▶ Interested in estimating parameters θ ,

$$P(Y|X; \theta)$$

Informally we use $p(\mathbf{y} = \delta(c)|\mathbf{x})$ for $P(Y = c|X = \mathbf{x})$.

Contents

Classification Review

Discriminative Models

Loss

Optimization

Discriminative Model. Conditional Log-Likelihood as Loss

- ▶ $(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n)$; supervised data
- ▶ Parameters maximize conditional likelihood of training data.

$$\mathcal{L}(\theta) = - \sum_{i=1}^n \log p(\mathbf{y}_i | \mathbf{x}_i; \theta)$$

For linear models $\theta = (\mathbf{W}, \mathbf{b})$

- ▶ (Contrast with generative models like NB, $p(\mathbf{x}_i, \mathbf{y}_i)$)
- ▶ Do this by minimizing negative log-likelihood (NLL).

$$\arg \min_{\theta} \mathcal{L}(\theta)$$

The Softmax

Alternative parametrization of probabilistic model.

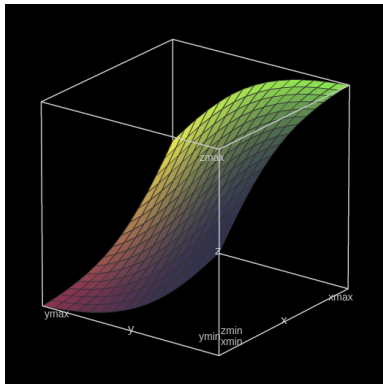
Use a softmax to force a distribution,

$$\text{softmax}(\mathbf{z}) = \frac{\exp(\mathbf{z})}{\sum_c \exp(z_c)}$$

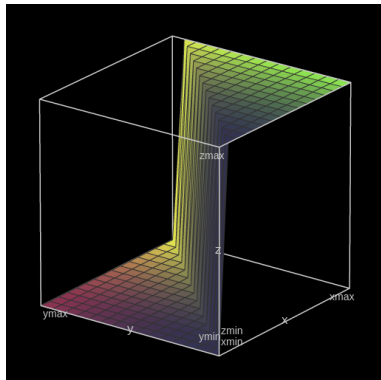
$$\log \text{softmax}(\mathbf{z}) = \mathbf{z} - \log \sum_c \exp(z_c)$$

- ▶ Exercise: Confirm always gives a distribution.
- ▶ Denominator known as *partition* function (we'll see many times).

Why is it called the softmax?



$$\text{softmax}([x \ y]) = \frac{\exp(x)}{\exp(x) + \exp(y)}$$



$$\text{arg max}([x \ y]) = \mathbf{1}(x > y)$$

Multiclass Logistic Regression

- ▶ Direct estimation of conditional $p(\mathbf{y} = c | \mathbf{x}; \theta)$

$$\hat{\mathbf{y}} = p(\mathbf{y} = c | \mathbf{x}; \theta) = \text{softmax}(\mathbf{x}\mathbf{W} + \mathbf{b})$$

- ▶ $\mathbf{W} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$, $\mathbf{b} \in \mathbb{R}^{1 \times d_{\text{out}}}$; model parameters
- ▶ “Regression” of the distribution.
- ▶ Classification still done as,

$$\hat{c} = \arg \max_{c \in \mathcal{C}} (\mathbf{x}\mathbf{W} + \mathbf{b})_c$$

Example: Multiclass Logistic Regression

3 classes, 2 features,

$$\mathbf{W} = \begin{bmatrix} 1 & 2 & -1 \\ -8 & 2 & 3 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

$$\mathbf{z} = \mathbf{xW} + \mathbf{b} = \begin{bmatrix} 1 & 2 & -1 \end{bmatrix}$$

Log-partition function,

$$\log \sum_c \exp(z_c) = \log(\exp(1) + \exp(2) + \exp(-1)) \approx 2.349$$

$$\log \text{softmax}(\mathbf{z}) \approx \begin{bmatrix} 1 - 2.349 & 2 - 2.349 & -1 - 2.349 \end{bmatrix}$$

$$p(\mathbf{y}|\mathbf{x}) = \begin{bmatrix} 0.259 & 0.705 & 0.035 \end{bmatrix}$$

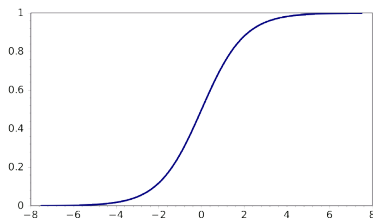
Important Case: Logistic Regression

For binary classification:

$$\begin{aligned}\text{softmax}([z_1 \ z_2]) &= \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)} \\ &= \frac{1}{1 + \exp(-(z_1 - z_2))} = \sigma(z_1 - z_2)\end{aligned}$$

Logistic sigmoid function:

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$



Multiclass Logistic Regression: A Model with Many Names

- ▶ Multinomial Logistic Regression
- ▶ Log-Linear Model (particularly in NLP)

$$\log \text{softmax}(\mathbf{z}) = \mathbf{z} - \log \sum_c \exp(z_c)$$

- ▶ Softmax Regression
- ▶ Max-Entropy (MaxEnt)

Contents

Classification Review

Discriminative Models

Loss

Optimization

Regression with Squared Loss

Define the squared error of two points \mathbf{p} and \mathbf{q} ,

$$\text{squared-error}(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_2^2$$

Can use to compare true with prediction,

$$L_{\text{squared-error}}(\mathbf{y}, \hat{\mathbf{y}}) = \|\mathbf{y} - \hat{\mathbf{y}}\|_2^2$$

Common case, when $y_c = 1$ for class c ,

$$L_{\text{squared-error}}(\mathbf{y}, \hat{\mathbf{y}}) = \|\delta(c) - \hat{\mathbf{y}}\|_2^2 \text{ where } y_c = 1$$

Not a good metric for comparing distributions.

Cross-Entropy Loss

Define the cross-entropy of two distributions \mathbf{p} and \mathbf{q} ,

$$H(\mathbf{p}, \mathbf{q}) = - \sum_c p_{c'} \log q_{c'}$$

Can use to compare true with prediction,

$$L_{\text{cross-entropy}}(\mathbf{y}, \hat{\mathbf{y}}) = - \sum_c y_{c'} \log \hat{y}_{c'}$$

Common case, when $y_c = 1$ for class c ,

$$L_{\text{cross-entropy}}(\mathbf{y}, \hat{\mathbf{y}}) = - \log \hat{y}_c \text{ where } y_c = 1$$

$$\hat{\mathbf{y}} = p(\mathbf{y}_i | \mathbf{x}_i; \theta)$$

Loss Minimization

$$L_{\text{cross-entropy}}(\mathbf{y}, \hat{\mathbf{y}}) = -\log \hat{y}_c \text{ where } y_c = 1$$

Equivalent to probabilistic objective:

$$\mathcal{L}(\theta) = -\sum_{i=1}^n \log p(\mathbf{y}_i | \mathbf{x}_i; \theta) = \sum_{i=1}^n L_{\text{cross-entropy}}(\mathbf{y}_i, \hat{\mathbf{y}}_i)$$

And the distribution is parameterized as a softmax,

$$\begin{aligned} L_{\text{cross-entropy}}(\mathbf{y}, \hat{\mathbf{y}}) &= -\log \hat{y}_c \\ &= -\log \text{softmax}(\mathbf{x}\mathbf{W} + \mathbf{b})_c \\ &= -z_c + \log \sum_{c' \in \mathcal{C}} \exp(z_{c'}) \end{aligned}$$

However, this has **no closed form** for minimization.

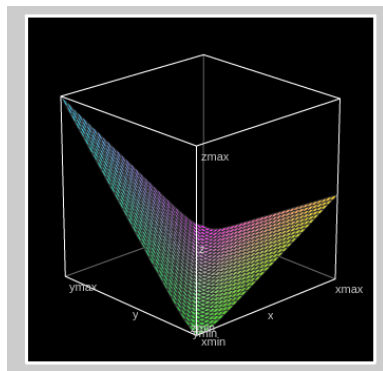
Objective

Complete objective:

$$\mathcal{L}(\theta) = - \sum_{i=1}^n (\mathbf{x}_i \mathbf{W} + \mathbf{b})_{c_i} + \log \sum_{c'} \exp(\mathbf{x}_i \mathbf{W} + \mathbf{b})_{c'}$$

- ▶ First term is linear in \mathbf{W} (convex)
- ▶ Second term is log-sum-exp of a linear functions of \mathbf{W} (convex).
- ▶ Objective is convex, but not strictly convex.
- ▶ Exercise(Hard) (Boyd and Vandenberghe, 2004 p. 72-74)

Objective



Loss for 2 classes, only bias, parameters x, y

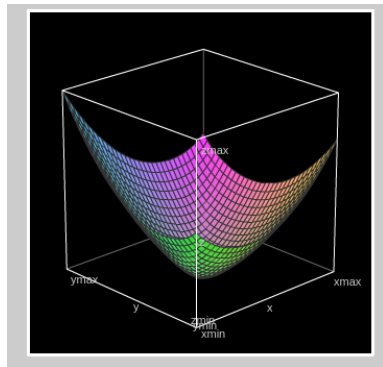
$$L([x \ y]) = -10 \log \text{softmax}([x \ y])_1 - 5 \log \text{softmax}([x \ y])_2$$

ℓ_2 Regularization

$$\mathcal{L}(\theta) = - \sum_{i=1}^n L(\mathbf{y}_i, \hat{\mathbf{y}}_i) + \frac{\lambda}{2} \|\theta\|_2^2$$

- ▶ Prevents overfitting on the training data
- ▶ LR will naturally send $\|W\| \rightarrow \infty$ (Murphy, p.252)
- ▶ Objective becomes strictly convex.

Regularized Optimization



Loss for 2 classes, only bias, parameters x, y , $\lambda = 2$

$$L([x \ y]) = -10 \log \text{softmax}([x \ y])_1 - 5 \log \text{softmax}([x \ y])_2 + \|[x \ y]\|_2^2$$

Contents

Classification Review

Discriminative Models

Loss

Optimization

Review: Calculus!

$$\frac{d(wx)}{dx} = w$$

$$\frac{d \log u(x)}{dx} = \frac{u'}{u}$$

$$\frac{d(u/v)}{dx} = \frac{u'v - uv'}{v^2}$$

$$\frac{d \exp u(x)}{dx} = u' \exp u$$

- ▶ For function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ Jacobian is

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f(\mathbf{x})_1}{\partial x_1} & \cdots & \frac{\partial f(\mathbf{x})_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f(\mathbf{x})_m}{\partial x_1} & \cdots & \frac{\partial f(\mathbf{x})_m}{\partial x_n} \end{bmatrix}$$

- ▶ In-Class: Compute Jacobian of (1) L , (2) softmax, (3) $\mathbf{z} = \mathbf{x}\mathbf{W} + \mathbf{b}$

Symbolic Gradients

- ▶ Partial of $L(\mathbf{y}, \hat{\mathbf{y}})$ for all $j \in \{1, \dots, d_{\text{out}}\}$ and $y_c = 1$

$$\frac{\partial L(\mathbf{y}, \hat{\mathbf{y}})}{\partial \hat{y}_j} = \begin{cases} -\frac{1}{\hat{y}_j} & j = c \\ 0 & \text{o.w.} \end{cases}$$

- ▶ Partial of $\hat{y}_j = \text{softmax}(\mathbf{z})$

$$\frac{\partial \hat{y}_j}{\partial z_i} = \begin{cases} \hat{y}_i(1 - \hat{y}_i) & i = j \\ -\hat{y}_i \hat{y}_j & i \neq j \end{cases}$$

- ▶ Partial of $\mathbf{z} = \mathbf{x}\mathbf{W} + \mathbf{b}$

$$\frac{\partial z_i}{\partial b_{i'}} = \mathbf{1}(i = i') \quad \frac{\partial z_i}{\partial W_{f,i'}} = x_f \mathbf{1}(i = i')$$

Symbolic Gradients

- ▶ Partial of $L(\mathbf{y}, \hat{\mathbf{y}})$ for all $j \in \{1, \dots, d_{\text{out}}\}$ and $y_c = 1$

$$\frac{\partial L(\mathbf{y}, \hat{\mathbf{y}})}{\partial \hat{y}_j} = \begin{cases} -\frac{1}{\hat{y}_j} & j = c \\ 0 & \text{o.w.} \end{cases}$$

- ▶ Partial of $\hat{\mathbf{y}} = \text{softmax}(\mathbf{z})$

$$\frac{\partial \hat{y}_j}{\partial z_i} = \begin{cases} \hat{y}_i(1 - \hat{y}_i) & i = j \\ -\hat{y}_i \hat{y}_j & i \neq j \end{cases}$$

- ▶ Partial of $\mathbf{z} = \mathbf{x}\mathbf{W} + \mathbf{b}$

$$\frac{\partial z_i}{\partial b_{i'}} = \mathbf{1}(i = i') \quad \frac{\partial z_i}{\partial W_{f,i'}} = x_f \mathbf{1}(i = i')$$

Symbolic Gradients

- ▶ Partials of $L(\mathbf{y}, \hat{\mathbf{y}})$ for all $j \in \{1, \dots, d_{\text{out}}\}$ and $y_c = 1$

$$\frac{\partial L(\mathbf{y}, \hat{\mathbf{y}})}{\partial \hat{y}_j} = \begin{cases} -\frac{1}{\hat{y}_j} & j = c \\ 0 & \text{o.w.} \end{cases}$$

- ▶ Partials of $\hat{\mathbf{y}} = \text{softmax}(\mathbf{z})$

$$\frac{\partial \hat{y}_j}{\partial z_i} = \begin{cases} \hat{y}_i(1 - \hat{y}_i) & i = j \\ -\hat{y}_i \hat{y}_j & i \neq j \end{cases}$$

- ▶ Partials of $\mathbf{z} = \mathbf{x}\mathbf{W} + \mathbf{b}$

$$\frac{\partial z_i}{\partial b_{i'}} = \mathbf{1}(i = i') \quad \frac{\partial z_i}{\partial W_{f,i'}} = x_f \mathbf{1}(i = i')$$

Review: Chain Rule

Assume we have a function and a loss:

$$f : \mathbb{R}^n \rightarrow \mathbb{R}^m \quad L : \mathbb{R}^m \rightarrow \mathbb{R}$$

Then for $i \in \{1, \dots, n\}$

$$\frac{\partial L(f(\mathbf{x}))}{\partial x_i} = \sum_{j=1}^m \frac{\partial f(\mathbf{x})_j}{\partial x_i} \frac{\partial L(f(\mathbf{x}))}{\partial f(\mathbf{x})_j}$$

Chain Rule

For multiclass logistic regression:

$$\frac{\partial L(\mathbf{y}, \hat{\mathbf{y}})}{\partial z_i} = \sum_j \frac{\partial \hat{y}_j}{\partial z_i} \frac{\mathbf{1}(j = c)}{\hat{y}_j} = \begin{cases} -(1 - \hat{y}_i) & i = c \\ \hat{y}_i & \text{ow.} \end{cases}$$

Therefore for parameters,

$$\frac{\partial L}{\partial b_i} = \frac{\partial L}{\partial z_i} \quad \frac{\partial L}{\partial W_{f,i}} = x_f \frac{\partial L}{\partial z_i}$$

Intuition:

- ▶ Nothing happens on correct classification.
- ▶ Weight of true features increases based on prob not given.
- ▶ Weight of false features decreases based on prob given.

Chain Rule

For multiclass logistic regression:

$$\frac{\partial L(\mathbf{y}, \hat{\mathbf{y}})}{\partial z_i} = \sum_j \frac{\partial \hat{y}_j}{\partial z_i} \frac{\mathbf{1}(j = c)}{\hat{y}_j} = \begin{cases} -(1 - \hat{y}_i) & i = c \\ \hat{y}_i & \text{ow.} \end{cases}$$

Therefore for parameters,

$$\frac{\partial L}{\partial b_i} = \frac{\partial L}{\partial z_i} \quad \frac{\partial L}{\partial W_{f,i}} = x_f \frac{\partial L}{\partial z_i}$$

Intuition:

- ▶ Nothing happens on correct classification.
- ▶ Weight of true features increases based on prob not given.
- ▶ Weight of false features decreases based on prob given.

Gradients-based minimization

Consider one example (\mathbf{x}, \mathbf{y}) , we compute “forward” and then “backward” ,

1. Compute scores $\mathbf{z} = \mathbf{x}\mathbf{W} + \mathbf{b}$
2. Compute softmax of scores, $\hat{\mathbf{y}} = \text{softmax}(\mathbf{z})$
3. Compute loss of scores, $L(\mathbf{y}, \hat{\mathbf{y}})$
4. Compute gradient $\frac{\partial L(\mathbf{y}, \hat{\mathbf{y}})}{\partial \hat{y}_j}$.
5. Compute gradient $\frac{\partial L(\mathbf{y}, \hat{\mathbf{y}})}{\partial z_i}$.
6. Compute gradient of \mathbf{b} and \mathbf{W} ,

Gradients-based minimization

Consider one example (\mathbf{x}, \mathbf{y}) , we compute “forward” and then “backward” ,

1. Compute scores $\mathbf{z} = \mathbf{x}\mathbf{W} + \mathbf{b}$
2. Compute softmax of scores, $\hat{\mathbf{y}} = \text{softmax}(\mathbf{z})$
3. Compute loss of scores, $L(\mathbf{y}, \hat{\mathbf{y}})$
4. Compute gradient $\frac{\partial L(\mathbf{y}, \hat{\mathbf{y}})}{\partial \hat{y}_j}$.
5. Compute gradient $\frac{\partial L(\mathbf{y}, \hat{\mathbf{y}})}{\partial z_i}$.
6. Compute gradient of \mathbf{b} and \mathbf{W} ,

Optimization

In next class we will talk more about gradient-based optimization.

- ▶ Here we simply describe an algorithm.

Gradient-Based Optimization: SGD

procedure SGD

while training criterion is not met **do**

 Sample a training example $\mathbf{x}_i, \mathbf{y}_i$

 Compute the loss $L(\hat{\mathbf{y}}_i, \mathbf{y}_i; \theta)$

 Compute gradients $\hat{\mathbf{g}}$ of $L(\hat{\mathbf{y}}_i, \mathbf{y}_i; \theta)$ with respect to θ

$\theta \leftarrow \theta - \eta_k \hat{\mathbf{g}}$

end while

return θ

end procedure

Gradient-Based Optimization: Mini-Batch SGD

while training criterion is not met **do**

Sample a minibatch of m examples $(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_m, \mathbf{y}_m)$

$\hat{\mathbf{g}} \leftarrow 0$

for $i = 1$ to m **do**

Compute the loss $L(\hat{\mathbf{y}}_i, \mathbf{y}_i; \theta)$

Compute gradients \mathbf{g}' of $L(\hat{\mathbf{y}}_i, \mathbf{y}_i; \theta)$ with respect to θ

$\hat{\mathbf{g}} \leftarrow \hat{\mathbf{g}} + \frac{1}{m} \mathbf{g}'$

end for

$\theta \leftarrow \theta - \eta_k \hat{\mathbf{g}}$

end while

return θ

Optimization with Regularization

The gradient with respect to the regularizer has a simple form for ℓ_2 :

$$\frac{\partial \|\theta\|}{\partial W_{f,i}} = \lambda W_{f,i}$$

But note that we are doing mini-batches, one solution is to split update:

$$\theta \leftarrow \theta - \eta_k(\hat{\mathbf{g}} + \lambda/n\theta)$$

$$\theta \leftarrow (1 - \eta_k\lambda/n)\theta - \eta_k(\hat{\mathbf{g}})$$

Often called “weight decay”.

Pros and Cons of Logistic Regression (Murphy, p 268)

Cons

- ▶ Harder to fit versus naive Bayes.
- ▶ Must fit all classes together.
- ▶ Not a good fit for semi-supervised/missing data cases

Pros

- ▶ Better calibrated probability estimates
- ▶ Natural handling of feature input
 - ▶ Features likely not multinomials
- ▶ (For us) extend naturally to neural networks

Softmax Notes: Calculating Log-Sum-Exp

- ▶ Calculating $\log \sum_{c' \in \mathcal{C}} \exp(\hat{y}_{c'})$ directly numerical issues.
- ▶ Instead $\log \sum_{c' \in \mathcal{C}} \exp(\hat{y}_{c'} - M) + M$ where $M = \max_{c' \in \mathcal{C}} \hat{y}_{c'}$