Text Classification

+

Machine Learning Review 3

CS 287

Review: 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

Review: Gradients for Softmax Regression

For multiclass logistic regression:

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

Therefore for parameters θ,

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

Review: Gradients for Softmax Regression

For multiclass logistic regression:

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

Therefore for parameters θ,

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

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.

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

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

end while

return θ

end procedure

Quiz: Softmax Regression

Given bag-of-word features

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

and two training data points:

Class 1: `The movie was terrible`

Class 2: `The movie rocked`

Assume that we start with parameters $\mathbf{W} = 0$ and $\mathbf{b} = 0$, and we train with learning rate $\eta = 1$ and $\lambda = 0$. What is the loss and the parameters after one pass through the data in order?

Answer: Softmax Regression (1)

First iteration,

$$\hat{\mathbf{y}}_1 = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}$$

$$L(\mathbf{y}_1, \hat{\mathbf{y}}_1) = -\log 0.5$$

$$\mathbf{W} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 & 0 & 0 \\ -0.5 & -0.5 & -0.5 & -0.5 & 0 & 0 \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} 0.5 & -0.5 \end{bmatrix}$$

Answer: Softmax Regression (2)

Second iteration,

$$\hat{\mathbf{y}}_1 = \text{softmax}([1.5 \ \ -1.5]) \approx \begin{bmatrix} 0.95 & 0.05 \end{bmatrix}$$

$$L(\mathbf{y}_2, \hat{\mathbf{y}}_2) = -\log 0.05$$

$$\mathbf{W} \approx \begin{bmatrix} -0.45 & -0.45 & 0.5 & 0.5 & -0.95 & 0 \\ 0.45 & 0.45 & -0.5 & -0.5 & 0.95 & 0 \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} -0.45 & 0.45 \end{bmatrix}$$

Today's Class

So far

- Naive Bayes (Multinomial)
- Multiclass Logistic Regression (SGD)

Today

- Multiclass Hinge-loss
- More about optimization

Contents

Other Loss Functions

What if we just try to directly find $\mathbf{W}$ and $\mathbf{b}$?

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

- No longer a probabilistic interpretation.
- Just try to find parameters that fit training data.

0/1 Loss

Just count the number of training examples we classify correctly,

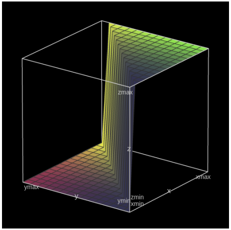
$$\mathcal{L}(\theta) = \sum_{i=1}^{n} L_{0/1}(\mathbf{y}, \hat{\mathbf{y}}) = \mathbf{1}(\arg\max_{c'} \hat{y}_{c'} \neq c)$$

0/1 Loss

Just count the number of training examples we classify correctly,

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} L_{0/1}(\mathbf{y}, \hat{\mathbf{y}}) = \mathbf{1}(\arg\max_{c'} \hat{y}_{c'} \neq c)$$

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



$$L_{0/1}([x \ y]) = \mathbf{1}(x > y)$$

Hinge Loss

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} L_{hinge}(\mathbf{y}, \hat{\mathbf{y}})$$

$$L_{hinge}(\mathbf{y}, \hat{\mathbf{y}}) = \max\{0, 1 - (\hat{y}_c - \hat{y}_{c'})\}$$

Where

- Let c be defined as true class $y_{i,c} = 1$

$$c' = \underset{i \in \mathcal{C} \setminus \{c\}}{\arg\max}\, \hat{y}_i$$

Hinge Loss

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} L_{hinge}(\mathbf{y}, \hat{\mathbf{y}})$$

$$L_{hinge}(\mathbf{y}, \hat{\mathbf{y}}) = \max\{0, 1 - (\hat{y}_c - \hat{y}_{c'})\}$$

Where

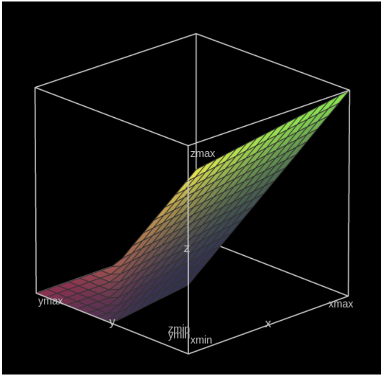
- Let c be defined as true class $y_{i,c} = 1$

$$c' = \arg\max_{i \in \mathcal{C} \setminus \{c\}} \hat{y}_i$$

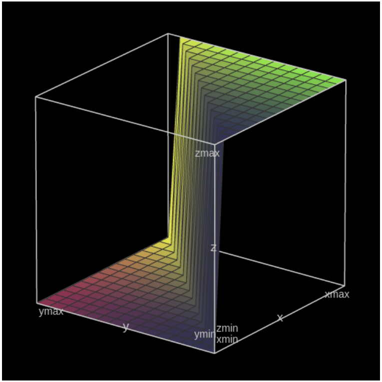
Minimizing hinge loss is an upper-bound for 0/1.

$$L_{hinge}(\mathbf{y}, \hat{\mathbf{y}}) \geq L_{0/1}(\mathbf{y}, \hat{\mathbf{y}})$$

Hinge Loss



$$\text{hinge}(\hat{\mathbf{y}}) = \mathbf{1}(\max\{0, 1 - (y - x)\})$$



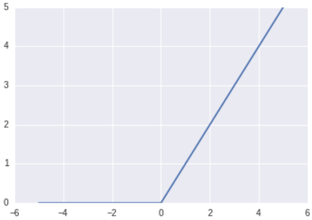
$$\arg\max([x \; y]) = \mathbf{1}(x > y)$$

Important Case: Hinge-loss for Binary

$$L_{hinge}([0\ 1], [x\ y]) = \max\{0, 1 - (y - x)\} = \text{ReLU}(1 - (y - x))$$

Neural network name (Rectified linear unit):

$$\text{ReLU}(t) = \max\{0, t\}$$



Hinge-Loss Properties

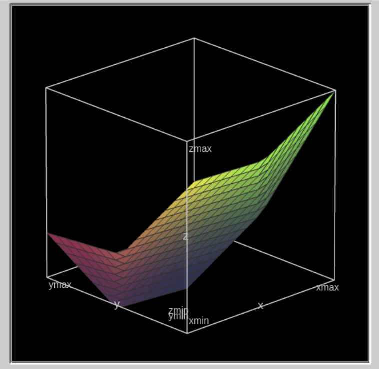
Complete objective:

$$\begin{aligned}\mathcal{L}_{hinge}(\theta) &= \sum_{i=1}^{n} \max\{0, 1 - (\hat{y}_c - \hat{y}_{c'})\} \\ &= \sum_{i=1}^{n} \max\{0, 1 - (\hat{y}_c - \max_{c' \in \mathcal{C} \setminus \{c\}} \hat{y}_{c'})\}\end{aligned}$$

- Apply convexity rules: Linear $\hat{\mathbf{y}}$ is convex, max of convex functions is convex, linear + convex is convex, sum of convex functions is convex (Boyd and Vandenberghe, 2004 p. 72-74)
- However, non-differentiable because of max.

Piecewise Linear Objective

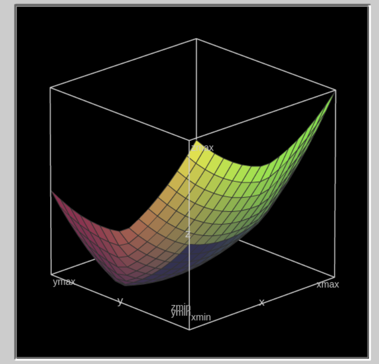
$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \max\{0, 1 - (\hat{y}_c - \hat{y}_{c'})\}$$



$$10 * \max\{0, 1 - (y - x)\} + 5 * \max\{0, 1 - (x - y)\}$$

Objective with Regularization

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \max\{0, 1 - (\hat{y}_c - \hat{y}_{c'})\} + \lambda ||\theta||^2$$



$$10 * \max\{0, 1 - (y - x)\} + 5 * \max\{0, 1 - (x - y)\} + 5 * ||\theta||^2$$

Contents

(Sub)Gradient Rule

- Technically ReLU is non-differentiable.
- Only an issue at 0, generally for "ties".
- We informally use subgradients,

$$\frac{d\,\text{ReLU}(x)}{dx} = \begin{cases} 1 & x > 0 \\ 0 & x < 0 \\ 1 \text{ or } 0 & o.w \end{cases}$$

Generally,

$$\frac{d \max_{v'}(f(x, v'))}{dx} = f'(x, \hat{v}) \text{ for any } \hat{v} \in \arg\max_{v'} f(x, v')$$

Symbolic Gradients

- Let c be defined as true class
- Let c' be defined as the highest scoring non-true class

$$c' = \arg\max_{i \in \mathcal{C} \setminus \{c\}} \hat{y}_i$$

- Partials of $L(y, \hat{y})$

$$\frac{\partial L(y, k\hat{y})}{\partial \hat{y}_j} = \begin{cases} 0 & \hat{y}_c - \hat{y}_{c'} > 1 \\ 1 & j = c' \\ -1 & j = c \\ 0 & o.w. \end{cases}$$

Intuition: If wrong or close to wrong, improve correct and lower closest incorrect.

Notes: Hinge Loss: Regularization

- Many different names,
  - Margin Classifier
  - Multiclass Hinge
  - Linear SVM
- Important to use regularization.

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

- Can be much more efficient to train than LR. (No partition).

Results: Longer Reviews

Our results	RT-2k	IMDB	Subj.
MNB-uni	83.45	83.55	**92.58**
MNB-bi	85.85	86.59	**93.56**
SVM-uni	86.25	86.95	90.84
SVM-bi	87.40	**89.16**	91.74
NBSVM-uni	87.80	88.29	92.40
NBSVM-bi	**89.45**	**91.22**	**93.18**
BoW (bnc)	85.45	87.8	87.77
BoW (bΔt'c)	85.8	88.23	85.65
LDA	66.7	67.42	66.65
Full+BoW	87.85	88.33	88.45
Full+Unlab'd+BoW	**88.9**	88.89	88.13

IMDB (longer movie review), Subj (longer subjectivity)

- NBSVM is hinge-loss interpolated with Naive Bayes.

Contents

Optimization Methods

Goal: Minimize function $\mathcal{L} : \mathbb{R}^{|\theta|} \mapsto \mathbb{R}$

First-order Methods

$$\mathcal{L}(\theta + \delta) \approx \mathcal{L}(\theta) + L'(\theta)^\top \dot{\delta}$$

- Require computing $L(\theta)$ and gradient $L'(\theta)$.

Second-order Methods

$$\mathcal{L}(\theta + \delta) \approx \mathcal{L}(\theta) + L'(\theta)^\top \delta + 1/2 \delta^\top \mathbf{H} \delta^\top$$

- Require computing $L(\theta)$ and gradient $L'(\theta)$ and Hessian $\mathbf{H}$.

Stochastic Methods

- Require computing $\mathbb{E}(L(\theta))$ and expected gradient.

Gradient Descent

while training criterion is not met **do**

$k \leftarrow 0$

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

for $i = 1 \text{ to } n$ **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

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

end for

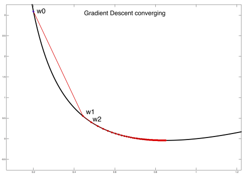
$\theta_{k+1} \leftarrow \theta_k - \eta_k \hat{\mathbf{g}}$

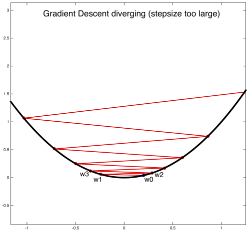
$k \leftarrow k + 1$

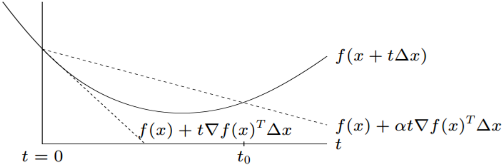
end while

return θ

Choosing the Learning Rate



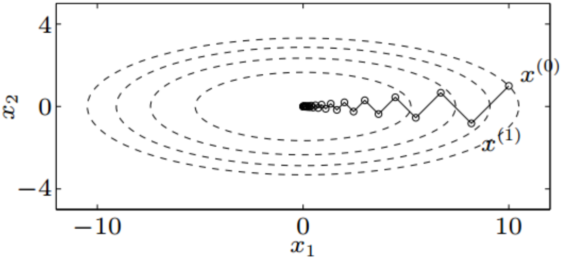




Gradient Descent with Momentum

Standard Gradient Descent (figure from Boyd):

$$\theta_{k+1} \leftarrow \theta_k - \eta_k \hat{\mathbf{g}}$$



Momentum terms:

$$\theta_{k+1} \leftarrow \theta_k - \eta_k \hat{\mathbf{g}} + \mu_k(\theta_k - \theta_{k-1})$$

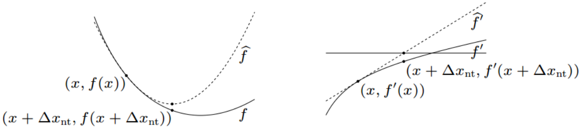
- Also known as: Heavy-ball method

Second-Order

Requires compute Hessian $\hat{\mathbf{H}}$

Second-order update becomes:

$$\theta_{k+1} \leftarrow \theta_k - \eta_k \hat{\mathbf{H}}^{-1} \hat{\mathbf{g}}$$



- Used for strictly convex functions (although there are variants)
- Also known as: Newton's Method

Quasi-Newton Methods

Construct an approximate Hessian from first-order information

- BFGS
  - construct approx. Hessian directly
  - $O(|\theta|^2)$ space
- L-BFGS;
  - limited-memory BFGS, only save last m gradients
  - can often set $m < 20$ or smaller

Fast implementations available, specific details are beyond scope of course.

Stochastic Methods

- Minimize function $L(\theta)$
- Require computing $\mathbb{E}(L(\theta))$ and gradient
- Typically, we by sampling a subset of the data. computing a gradient, and updating
- Other first-order optimizers (like momentum) can be used.

Gradient-Based Optimization: Minibatch SGD

while training criterion is not met **do**

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

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

  for $i = 1 \text{ to } m$ **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

Tricks On Using SGD (Bottou 2012)

- Can be crucial to experiment with learning rate.
- Often useful to use development set for stopping.
- Shuffle data first and run over it.
- Also describes "averaged" versions which work well in practice.

Optimization in NLP

- For convex batch-methods:
  - L-BFGS is easy to use and effective.
  - Nice for verifying results.
  - Sometimes even *m*-times the parameters is a lot though.
- For both convex, and especially, non-convex problems:
  - SGD and variants are dominant.
  - Trade-off of speed vs. exact optimization.
  - Also see notes on AdaGrad, another popular method.