CSE 512: Guided project: Due Dec. 7

1 Word embeddings

Open the ipython notebook `word_embedding_intro_release.ipynb`. Include in your writeup

- a frequency histogram, and a histogram of the co-occurances of the words in the dataset
- the list of the 10 most frequent words, and the 10 least frequent words
- the list of the 10 most cooccuring word pairs
- the 10 closest words to the city where you were born. (If you were born in a town which is not in the vocabulary, pick the closest city which *is* in the vocabulary.
- the 10 closest words to an object that is close to you right now.
- your 2-D PCA word embeddings, and some comments on any interesting geometric structure you may see

2 Duality theory and SVMs

- Remember the soft-margin SVM problem from forever ago?

$$\begin{array}{ll} \underset{\theta,s}{\text{minimize}} & \frac{1}{2}\|\theta\|_2^2 + C\sum_{i=1}^m \max\{s_i, 0\} \\ \text{subject to} & y_i x_i^T\theta + s_i = 1,\ i = 1,...,m \end{array} \tag{1}$$

  We're going to use this method to classify our parts of speech.

- **Subgradients and subdifferentials.** The hinge loss function

$$g(s) = \sum_{i=1}^m \max\{s_i, 0\}$$

  is unfortunately not differentiable whenever there exists one $s_i = 0$. But, it is convex. So, we can still take descent steps with respect to the objective by looking for *subgradients*. Specifically, for the scalar function

$$g_i(s) = \max\{s_i, 0\},$$

  we describe the *subdifferential* of g_i at s as

$$\partial g_i(s) = \mathbf{conv}\left\{\lim_{\epsilon\to 0^+} g_i'(s+\epsilon), \lim_{\epsilon\to 0^-} g_i'(s+\epsilon)\right\} = \begin{cases}\{1\} & s_i > 0\\ \{0\} & s_i < 0\\ [0,1] & s_i = 0.\end{cases}$$

  Since subdifferentials are linear (not obvious, but can be proved), we can summarize the subdifferential of the hinge function via

$$\partial(C\cdot g(s)) = C\sum_{i=1}^m \partial g_i(s)$$

  where we describe the transformation of sets as

$$C\mathcal{S} = \{Cx : x\in\mathcal{S}\} \text{ and } \mathcal{S}_1 + \mathcal{S}_2 = \{x+y : x\in\mathcal{S}_1,\ y\in\mathcal{S}_2\}.$$

- The subgradient method for minimizing $f(x)$ for some convex but not differentiable function f goes as

$$x^{(t+1)} = x^{(t)} - \alpha^{(t)} g^{(t)}, \quad g^{(t)} \in \partial f(x^{(t)}).$$

Here, $g^{(t)}$ is any element from $\partial f(x^{(t)})$. When you have more than one choice, you can pick any element, but you should design a rule that is agnostic to your knowledge of where the true solution is. When $\alpha^{(t)}$ is diminishing but not summable, e.g.

$$\alpha^{(t+1)} < \alpha^{(t)}, \qquad \lim_{t\to+\infty} \sum_{i=1}^{t} \alpha^{(i)} = \infty$$

then the subgradient method is known to converge. In particular, picking $\alpha^{(t)} = 1/(Lt)$, the method converges at a rate of $f(x^{(t)}) - f^* = O(1/\sqrt{(t)})$.

Q1 Show that for a *constant* step size, subgradient descent doesn't converge. Show this by arguing that minimizing the function

$$\underset{x}{\text{minimize}} \quad \frac{1}{2}(x-c)^2 + \max\{x, 0\}$$

cannot reach its optimum for *all* choices of c, using a step size that is agnostic to c.

Specifically, do this by picking a step size reference point $0 < \bar{s} < 2/L$, a starting point $x^{(0)} \neq x^*$, and some value of c, and show via simulation that $x^{(k)}$ doesn't converge for a step size of $\bar{s}$, $\bar{s}/10$, and $\bar{s}/100$.

- **Projections.** Consider the projection problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \frac{1}{2}\|x - \hat{x}\|_2^2 \\ \text{subject to} & Ax = b \end{array} \tag{2}$$

Q2 Find the Lagrange dual of (2) and use the solution of the dual to show that, assuming that AA^T is invertible, then the solution to (2) is

$$x = A^T(AA^T)^{-1}(b - A\hat{x}) + \hat{x}.$$

- **Projected subgradient descent.** For a problem of form

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & Ax = b \end{array}$$

the projected subgradient descent method runs the iterative scheme (from any initial point)

$$x^{(t+1)} = \mathbf{proj}_{\mathcal{H}}(x^{(t)} - \alpha g^{(t)})$$

where $g^{(t)}$ is a subgradient of f at $x^{(t)}$ and $\mathcal{H} = \{x : Ax = b\}$ and the projection of $\hat{x}$ on $\mathcal{H}$ is the solution to problem (2).

Q3 Use the pieces you have so far to propose an iteration scheme to solve (1) via projected subgradient descent. That is, given some $\theta^{(t)}$ and $s^{(t)}$, write out the equations used to compute $\theta^{(t+1)}$ and $s^{(t+1)}$, as explicitly as possible. (You will implement this code in the next section, so do it carefully here.)

- An equivalent formulation of (1) is the following

$$\begin{array}{ll} \underset{\theta, s}{\text{minimize}} & \frac{1}{2}\|\theta\|_2^2 + C\sum_{i=1}^{m} s_i \\ \text{subject to} & y_i x_i^T \theta + s_i \geq 1, \ i = 1, ..., m \\ & s_i \geq 0. \end{array} \tag{3}$$

Q4 Write down the Lagrangian saddle point problem of (3) and minimize it with respect to the primal variables to derive the following dual problem

$$\begin{array}{ll} \underset{u}{\text{maximize}} & -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j x_i^T x_j u_i u_j + \sum_{i=1}^{m} u_i \\ \text{subject to} & 0 \leq u \leq C. \end{array} \tag{4}$$

Q5 Show that if θ^*, s^* optimize (3) and u^* optimizes (4), then

1. $s_i^* > 1$ implies the ith training sample is not classified correctly,
2. $y_i x_i^T \theta > 1$ implies $u_i = 0$,
3. $y_i x_i^T \theta < 1$ implies $u_i = C$,
4. $0 < u_i < C$ implies $y_i x_i^T \theta = 1$.

Q6 Given u^*, propose a predictor, e.g. some function where $f(u^*, x) = x^T \theta^*$.

Q7 Propose how you would solve this using projected gradient descent, and how you would form a predictor. Remember that later, you will code this up, so include any detail here you would need to write this code.

For implementing the projection on the constraints, use the following convex optimization fact:

> The projection on a *box constraint*
>
> $$\mathcal{S} = \{x \in \mathbb{R}^n : a_i \le x_i \le b_i, \; i = 1, ..., n\}$$
>
> can be implemented elementwise as
>
> $$\mathbf{proj}_{\mathcal{S}}(\hat{x})_i = \begin{cases} a_i, & \text{if } \hat{x}_i < a_i \\ b_i, & \text{if } \hat{x}_i > b_i \\ \hat{x}_i, & \text{else.} \end{cases}$$

- **Kernel support vector machines** One way to think about kernel functions is to replace the linear prediction

$$y = \mathbf{sign}(x_i^T \theta)$$

with a lifted version of x_i, as $\phi(x_i)$, e.g.

$$y = \mathbf{sign}(\phi(x_i)^T \theta).$$

After finding the dual, this gives us an objective function of

$$g(u) = -\tfrac{1}{2} \sum_{i=1}^{m} (y_i \phi(x_i)^T u)^2 + \sum_{i=1}^{m} u_i = -\tfrac{1}{2} u^T K u + u^T \mathbf{1}$$

where $K_{ij} = y_i y_j \phi(x_i)^T \phi(x_j)$ defines the Kernel matrix. The Kernel trick is simply, instead of holding onto the representations $\phi(x)$, to only hold onto these inner product values. For example, the radial basis function (RBF) kernel computes

$$K_{ij} = K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|_2}{2\sigma^2}\right), \qquad \hat{K}_{ij} = K_{ij} y_i y_j$$

which can be precomputed.

$$\begin{array}{ll} \underset{u}{\text{maximize}} & -\tfrac{1}{2} \sum_{i=1}^{m} u^T \hat{K} u^2 + \sum_{i=1}^{m} u_i \\ \text{subject to} & 0 \le u \le C. \end{array} \tag{5}$$

Q8 Propose how, given u^*, you would offer a prediction on a new datasample, x. Remember that you do not have access to ϕ, but you do have access to the kernel function $\mathcal{K}$.

Q9 Propose how you would solve this using projected gradient descent. Remember that later, you will code this up, so include any detail here you would need to write this code.

3 Multiclass classification

Now open `svm_POS_release.ipynb` and go through the notebook. Report in your writeup

- loss, train and test misclassification plots for noun-vs-all, for primal projected subgradient method
- comparison of loss functions for primal and dual linear SVM
- evidence of cross validation to determine best value of C in the primal linear SVM and C and σ in Kernel SVM
- confusion matrix of final answer on train, validation, and test set.

4 Discussion

Conclude your report with some observations and thoughts on

- the computational complexity of each method
- the effectiveness of each method
- any tricks used to deal with imbalanced data, or with overfitting.

Would you recommend using word embedding for POS classification, using the method we proposed here? Using a different method? Not at all?