Announcements

• Project proposal due tonight!
Stochastic Gradient Descent

- Have a bunch of iid data of the form:
  \[ \{ (x_i, y_i) \}_{i=1}^{n} \]
  \[ x_i \in \mathbb{R}^d \quad y_i \in \mathbb{R} \]

- Learning a model’s parameters:
  Each \( \ell_i(w) \) is convex.

\[
\frac{1}{n} \sum_{i=1}^{n} \ell_i(w)
\]
Stochastic Gradient Descent

- Have a bunch of iid data of the form:
  \[ \{(x_i, y_i)\}_{i=1}^{n} \quad x_i \in \mathbb{R}^d \quad y_i \in \mathbb{R} \]

- Learning a model’s parameters:
  Each \( \ell_i(w) \) is convex.

Gradient Descent:
\[
 w_{t+1} = w_t - \eta \nabla_w \left( \frac{1}{n} \sum_{i=1}^{n} \ell_i(w) \right) \bigg|_{w=w_t}
\]
Stochastic Gradient Descent

- Have a bunch of iid data of the form:
  \[ \{(x_i, y_i)\}_{i=1}^{n} \quad x_i \in \mathbb{R}^d \quad y_i \in \mathbb{R} \]

- Learning a model's parameters:
  Each \( \ell_i(w) \) is convex.

Gradient Descent:
\[
w_{t+1} = w_t - \eta \nabla_w \left( \frac{1}{n} \sum_{i=1}^{n} \ell_i(w) \right) \bigg|_{w=w_t}
\]

Stochastic Gradient Descent:
\[
w_{t+1} = w_t - \eta \nabla_w \ell_{I_t}(w) \bigg|_{w=w_t} \quad I_t \text{ drawn uniform at random from } \{1, \ldots, n\}
\]

\[
\mathbb{E}[\nabla \ell_{I_t}(w)] = \nabla f(w_t)
\]

\[
\mathbb{E} [f(w_t)] - f(w_0) \leq \frac{c}{\sqrt{t}}
\]
Stochastic Gradient Descent: A Learning perspective

Machine Learning – CSE546
Kevin Jamieson
University of Washington

October 24, 2017
Learning Problems as Expectations

- Minimizing loss in training data:
  - Given dataset:
    - Sampled iid from some distribution $p(x)$ on features:
  - Loss function, e.g., hinge loss, logistic loss,…
  - We often minimize loss in training data:
    \[
    \ell_D(w) = \frac{1}{N} \sum_{j=1}^{N} \ell(w, x^j)
    \]

- However, we should really minimize expected loss on all data:
  \[
  \ell(w) = E_x [\ell(w, x)] = \int p(x) \ell(w, x) dx
  \]

- So, we are approximating the integral by the average on the training data
Gradient descent in Terms of Expectations

- “True” objective function:
  \[ E_x [\ell(w, x)] \]

- Taking the gradient:
  \[ \nabla E_x [\ell(w, x)] = \mathbb{E}_x [\nabla \ell(w, x)] \]

- “True” gradient descent rule:
  \[ w_{t+1} = w_t - 2 \mathbb{E}_x [\nabla \ell(w, x)] \]

- How do we estimate expected gradient?
  \[ w_{t+1} = w_t - 2 \nabla \ell_i (w, x_i) \]
SGD: Stochastic Gradient Descent

- “True” gradient: \[ \nabla \ell(w) = E_x [\nabla \ell(w, x)] \]

- One iid sample estimate:

- How many iid samples do we have?

  \[ n \text{ iid samples (not infinite)} \]

  So we cannot get infinite stream of iid samples

See [Hardt, Recht, Singer 2016] for resolution based on stability
Perceptron

\( \ell_c(w, x_i) = (y_i - w^T x_i)^2 \)

Machine Learning – CSE546
Kevin Jamieson
University of Washington

October 24, 2017
Click prediction for ads is a streaming data task:
- User enters query, and ad must be selected
  - Observe $x^j$, and must predict $y^j$
- User either clicks or doesn’t click on ad
  - Label $y^j$ is revealed afterwards
    - Google gets a reward if user clicks on ad
- Update model for next time
Online classification

New point arrives at time $k$
The Perceptron Algorithm [Rosenblatt '58, '62]

- Classification setting: $y$ in \{-1,+1\}
- Linear model
  - Prediction: $y_k = \text{sign} \left( x_n^T w + b \right)$

Training:
- Initialize weight vector: $w_0 = 0, b_0 = 0$
- At each time step $k$:
  - Observe features: $x_k$
  - Make prediction: $y^*_{kn} = \text{sign} \left( x_k^T w_n + b_n \right)$
  - Observe true class: $y_h$
  - Update model:
    - If prediction is not equal to truth
      $$
      \begin{pmatrix}
      w_{n+1} \\
      b_{n+1}
      \end{pmatrix} = \begin{pmatrix}
      w_n \\
      b_n
      \end{pmatrix} + y_n \begin{pmatrix}
      x_k \\
      1
      \end{pmatrix}
      $$
The Perceptron Algorithm

Classification setting: \( y \) in \{-1,+1\}

Linear model

- Prediction: \( \text{sign}(w^T x_i + b) \)

Training:

- Initialize weight vector: \( w_0 = 0, b_0 = 0 \)
- At each time step:
  - Observe features: \( x_k \)
  - Make prediction: \( \text{sign}(x_k^T w_k + b_k) \)
  - Observe true class: \( y_k \)

Update model:

- If prediction is not equal to truth

\[
\begin{bmatrix}
w_{k+1} \\
b_{k+1}
\end{bmatrix} = \begin{bmatrix}
w_k \\
b_k
\end{bmatrix} + y_k \begin{bmatrix} x_k \\ 1 \end{bmatrix}
\]
"the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence."

Linear Separability

- Perceptron guaranteed to converge if
  - Data linearly separable:
Perceptron Analysis: Linearly Separable Case

- Theorem [Block, Novikoff]:
  - Given a sequence of labeled examples: 
    \((x_i, y_i)\) \(i = 1, 2, \ldots\)
  - Each feature vector has bounded norm: 
    \(\|x_i\|_2 \leq R\)
  - If dataset is linearly separable:
    \(\exists \mathbf{w}, b_0 \quad \text{sign} (\mathbf{w}^T x_i + b_0) = y_i \quad \forall i\)

- Then the number of mistakes made by the online perceptron on any such sequence is bounded by
  \[ \frac{R^2}{\gamma^2} \]
  \(\gamma = \text{"margin"}
  \text{Gap between classes} \)
Beyond Linearly Separable Case

- Perceptron algorithm is super cool!
  - No assumption about data distribution!
    - Could be generated by an oblivious adversary, no need to be iid
  - Makes a fixed number of mistakes, and it’s done for ever!
    - Even if you see infinite data
Beyond Linearly Separable Case

- Perceptron algorithm is super cool!
  - No assumption about data distribution!
    - Could be generated by an oblivious adversary, no need to be iid
  - Makes a fixed number of mistakes, and it’s done for ever!
    - Even if you see infinite data

- Perceptron is useless in practice!
  - Real world not linearly separable
  - If data not separable, cycles forever and hard to detect
  - Even if separable may not give good generalization accuracy (small margin)
What is the Perceptron Doing???

- When we discussed logistic regression:
  - Started from maximizing conditional log-likelihood

- When we discussed the Perceptron:
  - Started from description of an algorithm

- What is the Perceptron optimizing????

\[(x_i, y_i) \text{ arrives w/ loss } \max \theta_0 - (x_i^T w + b) y_i \]

\[w_{k+1} = w_k - \Delta L_k(w_k) \text{ where } L_k = \max \theta_0 - y_i (x_i^T w + b) \]
Linear classifiers – Which line is better?
Pick the one with the largest margin!
Pick the one with the largest margin!

Distance of $x_0$ from hyperplane $x^T w + b$:

$$\frac{1}{||w||_2} (x_0^T w + b)$$

$$= \frac{1}{||w||_2} (x_0^T w + y_0)$$

If $w$ classifier $x_0$ as $y_0$
\[
\langle x, y \rangle = \sum_{i=1}^{d} x_i y_i = x^T y
\]

Pick the one with the largest margin!

Distance of \( x_0 \) from hyperplane \( x^T w + b \):

\[
\frac{1}{||w||_2} (x_0^T w + b)
\]

Optimal Hyperplane

\[
\max_{w,b} \gamma \\
\text{subject to } \frac{1}{||w||_2} y_i (x_i^T w + b) \geq \gamma \quad \forall i
\]
Pick the one with the largest margin!

Distance of $x_0$ from hyperplane $x^T w + b$:

$$
\frac{1}{\|w\|_2} (x_0^T w + b)
$$

Optimal Hyperplane

$$
\max_{w,b} \gamma
$$

subject to

$$
\frac{1}{\|w\|_2} y_i(x_i^T w + b) \geq \gamma \quad \forall i
$$

Optimal Hyperplane (reparameterized)

$$
\min_{w,b} \|w\|_2^2
$$

subject to

$$
y_i(x_i^T w + b) \geq 1 \quad \forall i
$$
Pick the one with the largest margin!

\[ x^T w + b = 0 \]

- Solve efficiently by many methods, e.g.,
  - quadratic programming (QP)
    - Well-studied solution algorithms
  - Stochastic gradient descent
  - Coordinate descent (in the dual)

**Optimal Hyperplane (reparameterized)**

\[
\begin{align*}
\min_{w,b} & \; \|w\|^2_w \\
\text{subject to} & \; y_i(x_i^T w + b) \geq 1 \quad \forall i
\end{align*}
\]
What if the data is still not linearly separable?

If data is linearly separable

\[ \min_{w,b} \|w\|_2^2 \]

\[ y_i(x_i^T w + b) \geq 1 \quad \forall i \]
What if the data is still not linearly separable?

- If data is linearly separable
  \[
  \min_{w,b} \frac{1}{||w||_2^2} \quad \text{subject to} \quad y_i(x_i^T w + b) \geq 1 \quad \forall i
  \]

- If data is not linearly separable, some points don’t satisfy margin constraint:
  \[
  \min_{w,b} ||w||_2^2 \\
  y_i(x_i^T w + b) \geq 1 - \xi_i \quad \forall i \\
  \xi_i \geq 0, \sum_{j=1}^{n} \xi_j \leq \nu
  \]
What if the data is still not linearly separable?

- If data is linearly separable
  
  \[
  \min_{w,b} \|w\|^2_2 \\
  y_i(x_i^T w + b) \geq 1 \quad \forall i
  \]

- If data is not linearly separable, some points don’t satisfy margin constraint:
  
  \[
  \min_{w,b} \|w\|^2_2 \\
  y_i(x_i^T w + b) \geq 1 - \xi_i \quad \forall i \\
  \xi_i \geq 0, \sum_{j=1}^{n} \xi_j \leq \nu
  \]

- What are “support vectors?”
SVM as penalization method

- Original quadratic program with linear constraints:

\[
\min_{w, b} \|w\|_2^2 \\
y_i(x_i^T w + b) \geq 1 - \xi_i \quad \forall i \\
\xi_i \geq 0, \sum_{j=1}^{n} \xi_j \leq \nu
\]
SVM as penalization method

- Original quadratic program with linear constraints:
  \[
  \min_{w,b} \|w\|^2_2 + c\nu \\
  y_i(x_i^T w + b) \geq 1 - \xi_i \quad \forall i \\
  \xi_i \geq 0, \sum_{j=1}^n \xi_j \leq \nu
  \]

- Using same constrained convex optimization trick as for lasso:
  
  For any \(\nu \geq 0\) there exists a \(\lambda \geq 0\) such that the solution
  the following solution is equivalent:

  \[
  \sum_{i=1}^n \max\{0, 1 - y_i(b + x_i^T w)\} + \lambda\|w\|^2_2
  \]
Machine Learning Problems

- Have a bunch of iid data of the form:
  \[ \{(x_i, y_i)\}_{i=1}^{n} \quad x_i \in \mathbb{R}^d \quad y_i \in \mathbb{R} \]

- Learning a model’s parameters:
  Each \( \ell_i(w) \) is convex.

\[
\sum_{i=1}^{n} \ell_i(w)
\]

Hinge Loss: \( \ell_i(w) = \max\{0, 1 - y_i x_i^T w\} \)

Logistic Loss: \( \ell_i(w) = \log(1 + \exp(-y_i x_i^T w)) \)

Squared error Loss: \( \ell_i(w) = (y_i - x_i^T w)^2 \)

How do we solve for \( w \)? The last two lectures!
SVMs vs logistic regression

- We often want probabilities/confidences, logistic wins here?
SVMs vs logistic regression

- We often want probabilities/confidences, logistic wins here?
- No! Perform isotonic regression or non-parametric bootstrap for probability calibration. Predictor gives some score, how do we transform that score to a probability?
SVMs vs logistic regression

- We often want probabilities/confidences, logistic wins here?
- No! Perform isotonic regression or non-parametric bootstrap for probability calibration. Predictor gives some score, how do we transform that score to a probability?

- For classification loss, logistic and svm are comparable
- **Multiclass setting:**
  - Softmax naturally generalizes logistic regression
  - SVMs have
- What about good old least squares?
What about multiple classes?

$\text{1 vs all}$

$\text{L}_i^c : \text{class } c \text{ vs } i \neq c$

$\text{1 vs 1}$

$\left( \frac{K}{2} \right) \text{ classifiers}$