Lecture 2: (review) Binary Classification

Kai-Wei Chang
CS @ University of Virginia
kw@kwchang.net

Some slides are adapted from Vivek Skirmar and Dan Roth
Administration

- Remember to register your presentation group
  https://goo.gl/uSy5ta

- TA session: Tuesday 5pm-6pm CS414
  - TA will discuss the solution of Hw0
Previous Lecture

- Overview of structured prediction and representation learning

- Question?
This Lecture

- Supervised Learning
- Linear Classifiers
  - Perceptron Algorithm
  - Support Vector Machine
  - Logistic Regression
- Optimization in machine learning
The Badges game

- Naoki Abe
- Eric Baum

- Conference attendees to the ICML 1994 were given name badges labeled with + or −.
- What function was used to assign these labels?
## Training data

<table>
<thead>
<tr>
<th>+</th>
<th>-</th>
<th>+</th>
<th>-</th>
<th>+</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naoki Abe</td>
<td>Myriam Abramson</td>
<td>David W. Aha</td>
<td>Eric Allender</td>
<td>Dana Angluin</td>
<td>Chidanand Apte</td>
</tr>
<tr>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Kamal M. Ali</td>
<td>Eric</td>
<td>Aha</td>
<td>Minoru Asada</td>
<td>Lars Asker</td>
<td>Jose L. Balcazar</td>
</tr>
<tr>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Lars Asker</td>
<td>Apte</td>
<td>Asada</td>
<td>Aslam</td>
<td>Balcazar</td>
<td>Baroglio</td>
</tr>
<tr>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Carla E. Brodley</td>
<td>Peter Bartlett</td>
<td>Eric Baum</td>
<td>Welton Becket</td>
<td>Shai Ben-David</td>
<td>George Berg</td>
</tr>
<tr>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Nader Bshouty</td>
<td>Berkman</td>
<td>George</td>
<td>Allender</td>
<td>Berkman</td>
<td>Malini Bhandaru</td>
</tr>
<tr>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Wray Buntine</td>
<td>Wray</td>
<td>Ben-David</td>
<td>Bylander</td>
<td>Byrne</td>
<td>Bhandaru</td>
</tr>
<tr>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Andrey Burago</td>
<td>Bir</td>
<td>Bhanu</td>
<td>Claire</td>
<td>Case</td>
<td>Boyan</td>
</tr>
<tr>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Chris Darken</td>
<td>Avrim</td>
<td>Blum</td>
<td>Jason</td>
<td>Chan</td>
<td>Blum</td>
</tr>
<tr>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Philip Chan</td>
<td>Anselm</td>
<td>Blumer</td>
<td>Zhixiang</td>
<td>Chen</td>
<td>Darken</td>
</tr>
</tbody>
</table>
Raw test data

Gerald F. DeJong
Chris Drummond
Yolanda Gil
Attilio Giordana
Jiarong Hong
J. R. Quinlan

Priscilla Rasmussen
Dan Roth
Yoram Singer
Lyle H. Ungar
Why we need machine learning?

- There is no (or limited numbers of) human expert for some problems
  - E.g.: Identify DNA binding sites, predicting disease progression, predicting protein folding structure
Why we need machine learning?

- There is no (or limited numbers of) human expert for some problems
- Humans can perform a task, but can’t describe how they do it
- E.g.: Object recognition
Why we need machine learning?

- There is no (or limited numbers of) human expert for some problems
- Humans can perform a task, but can’t describe how they do it
- The desired function is hard to be written down in a closed form
  - E.g.,: predict stock price
Supervised learning

Input

\[ x \in X \]

An item \( x \) drawn from an instance space \( X \)

Learned Model

\[ y = f(x) \]

Output

\[ y \in Y \]

An item \( y \) drawn from a label space \( Y \)

Target function

\[ y = f^*(x) \]
Supervised learning

Input

\( x \in \mathcal{X} \)

An item \( x \) drawn from an instance space \( \mathcal{X} \)

\( x \) is represented in a feature space
- Typically \( x \in \{0,1\}^n \) or \( \mathbb{R}^N \)
- Usually represented as a vector
- We call it input vector
Supervised learning

$y$ is represented in output space (label space)
Different kinds of output:

- Binary classification:
  \[ y \in \{-1,1\} \]
- Multiclass classification:
  \[ y \in \{1,2,3,...K\} \]
- Regression:
  \[ y \in \mathbb{R} \]
- Structured output
  \[ y \in \{1,2,3,...K\}^N \]

Output

An item $y$ drawn from a label space $\mathcal{Y}$
Learning the mapping

Input

$x \in X$

An item $x$ drawn from an instance space $X$

Output

$y \in Y$

An item $y$ drawn from a label space $Y$

Target function

$y = f^*(x)$

Learned Model

$y = f(x)$

$x \in X$

An item $x$ drawn from an instance space $X$

$y \in Y$

An item $y$ drawn from a label space $Y$
Goal

- Find a good approximation of $f^*(\cdot)$
- Good in what sense?
Under-fitting and over-fitting

- Which classifier (blue line) is the best one?
Bias V.S. Variance

- Remember, training data are subsamples drawn from the true distribution

- Exam strategy:
  - Study every chapter well
    - A+: Low var & bias
  - Study only a few chapters
    - A+? B? C? Low bias; High var
  - Study every chapter roughly
    - B+: Low var; high bias
  - Go to sleep
    - B ~D: High var, high bias
Questions of interest

- Representation
  - How to represent \( x, y \) (and latent factors)
- Modeling
  - What assumptions we made
    - i.e., what is the hypothesis set of \( f \)?
- Algorithms
  - (learn) Give data, how to learn \( f \)?
  - (inference) Give test instance \( x \) and \( f \), how to evaluate \( f(x) \)?
- Learning protocols
  - What is the goal of the learning algorithm?
Different learning protocols (more technical terms)

- Supervision signals?
  - Supervised learning, semi-supervised learning, unsupervised learning, bandit feedback

- What to be optimized?
  - Batch learning: minimize the risk (expected average loss)
  - Online learning:
    Receive one sample and make prediction
    Receive the label; then update
    minimize the accumulated loss
Linear classification

Teacher

Today, we are going to learn about matrix

Expectation

Reality

\[ b = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{bmatrix} \]

\[ e = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{bmatrix} \]

\[ \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \]
Linear classifiers

- For now, we consider binary classification
- Given a training set $\mathcal{D} = \{(x, y)\}$, find a linear threshold units classify an example $x$ using the classification rule:

\[
\text{sgn}(b + w^T x) = \text{sgn}(b + \sum_i w_i x_i)
\]

- $b + w^T x \geq 0 \Rightarrow$ Predict $y = 1$
- $b + w^T x < 0 \Rightarrow$ Predict $y = -1$
The geometry interpretation

In n dimensions, a linear classifier represents a hyperplane that separates the space into two half-spaces.

\[ \text{sgn}(b + w_1 x_1 + w_2 x_2) \]

\[ b + w_1 x_1 + w_2 x_2 = 0 \]
Some data are not linearly separable
But they can be made liner

Using a different representation e.g., feature conjunctions, non-linear mapping
Exercise: can you make these data points linearly separable?
A simple trick to remove the bias term $b$

\[ w^T x + b \]
\[ = [w^T \ b] \cdot \begin{bmatrix} x \\ 1 \end{bmatrix} \]
\[ = \tilde{w} \cdot \tilde{x} \]

\[ \tilde{w} = [w^T \ b]^T \]
\[ \tilde{x} = [x^T \ 1]^T \]

For simplicity, I’ll write $\tilde{w}$ and $\tilde{x}$ as $w$ and $x$ when there is no confusion.
Linear classifiers

- Let’s take a look at a few linear classifiers
- We will show later, they can be written in the same framework!

- Perceptron
- (Linear) Support Vector Machines
- Logistic Regression
The Perceptron Algorithm [Rosenblatt 1958]

- Goal: find a **separating hyperplane**
- Can be used in an **online** setting: considers one example at a time
- Converges if data is separable -- mistake bound
The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x,y)\}$

1. **Initialize** $\mathbf{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$

2. **For** epoch $1 \ldots T$:

3. **For** $(x, y)$ **in** $\mathcal{D}$:

4. \[\hat{y} = \text{sgn}(\mathbf{w}^\top \mathbf{x})\] \hspace{1cm} (predict)

5. \[\text{if } \hat{y} \neq y, \quad \mathbf{w} \leftarrow \mathbf{w} + \eta y \mathbf{x}\] \hspace{1cm} (update)

6. **Return** $\mathbf{w}$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(\mathbf{w}^\top \mathbf{x}^{\text{test}})$
The Perceptron Algorithm [Rosenblatt 1958]

Given a training set \( \mathcal{D} = \{(x, y)\} \)

1. **Initialize** \( w \leftarrow 0 \in \mathbb{R}^n \)
2. **For** epoch 1...\( T \):
3. **For** \((x, y)\) in \( \mathcal{D} \):
4. \( \text{if } y(w^T x) < 0 \)
5. \( w \leftarrow w + \eta y x \)
6. **Return** \( w \)

**Prediction:** \( y^{\text{test}} \leftarrow \text{sgn}(w^T x^{\text{test}}) \)
Geometry Interpretation

\( \mathbf{x} \) (with \( y = +1 \)) next item to be classified

\[ \mathbf{w} \mathbf{x} = 0 \]
Current decision boundary

\[ \mathbf{x} \text{ as a vector} \]

\[ \mathbf{w} \]
Current weight vector

\[ \mathbf{x} \text{ as a vector added to } \mathbf{w} \]

\[ \mathbf{w} \]
New weight vector

\[ \mathbf{w} \mathbf{x} = 0 \]
New decision boundary

(Figures from Bishop 2006)
Perceptron in action

\[ wx = 0 \]

Current decision boundary

\[ wx = 0 \]

New decision boundary

**x** (with \( y = +1 \))
next item to be classified

**x** as a vector

**x** as a vector added to \( w \)

(Figures from Bishop 2006)

(Figures from Bishop 2006)
Convergence of Perceptron

- Mistake bound
  - If data is linearly separable (i.e., a good linear model exists), the perceptron will converge after a fixed number of mistakes [Novikoff 1962]
Marginal Perceptron -- Motivation

- Which separating hyper-plane is better?

![Diagram showing two hyper-planes](image_url)
The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x,y)\}$

1. **Initialize** $w \leftarrow 0 \in \mathbb{R}^n$

2. **For** epoch $1...T$:

3. **For** $(x,y)$ in $\mathcal{D}$:

4. **if** $y(w^T x) < 0$

5. $w \leftarrow w + \eta y x$

6. **Return** $w$

**Prediction:** $y^{test} \leftarrow \text{sgn}(w^T x^{test})$
The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x,y)\}$

1. **Initialize** $w \leftarrow 0 \in \mathbb{R}^n$
2. **For** epoch $1 \ldots T$:
3. **For** $(x,y)$ in $\mathcal{D}$:
4. **if** $y(w^T x) < \delta$
5. $w \leftarrow w + \eta y x$
6. **Return** $w$

**Prediction:** $y^{test} \leftarrow \text{sgn}(w^T x^{test})$
How about batch setting?

- Learning as loss minimization

1. Collect training data \( \hat{D} = \{(x, y)\} \)

2. Pick a hypothesis class
   - E.g., linear classifiers, deep neural networks

3. Choose a **loss function**
   - Hinge loss, negative log-likelihood
   - We can impose a preference (i.e., prior) over hypotheses, e.g., simpler is better

4. Minimize the **expected loss**
   - SGD, coordinate descent, Newton methods, LBFGS
Batch learning setup

- \( \hat{D} = \{(x, y)\} \) drawn from a fixed, unknown distribution \( D \)
- A hidden oracle classifier \( f^* \), \( y = f^*(x) \)
- We wish to find a hypothesis \( f \in H \) that mimics \( f^* \)
- We define a loss function \( L(f(x), f^*(x)) \) that penalizes mistakes
- What is the ideal \( f \)?

\[
\arg\min_{f \in H} \mathbb{E}_{x \sim D} \left[ L(f(x), f^*(x)) \right]
\]

expected loss
Batch learning setup

- \( \hat{D} = \{(x, y)\} \) drawn from a fixed, unknown distribution \( D \)
- A hidden oracle classifier \( f^* \), \( y = f^*(x) \)
- We wish to find a hypothesis \( f \in H \) that mimics \( f^* \)
- We define a loss function \( L(f(x), f^*(x)) \) that penalizes mistakes

- What is the ideal \( f \)?

Let's define

\[
L_{0-1}(y, y') = \begin{cases} 
1 & \text{if } y \neq y' \\
0 & \text{if } y = y'
\end{cases}
\]

\[
\min_{f \in H} \mathbb{E}_{x \sim D} \left[ L_{0-1}(f(x), f^*(x)) \right] = \min_{f \in H} \mathbb{E}_{x \sim D}[ \# \text{mistakes} ]
\]
How can we learn $f$ from $\hat{D}$

- We don’t know $D$, we only see samples in $\hat{D}$
- Instead, we minimize empirical loss

$$\min_{f \in H} \frac{1}{|\hat{D}|} \sum_{(x,y) \in \hat{D}} [L(f(x), y)]$$
How can we prevent over-fitting?

- With sufficient data, \( \hat{D} \approx D \)
- However, if data is insufficient \( \Rightarrow \) overfitting
- We can impose a preference over models

\[
\min_{f \in H} R(f) + \frac{1}{|\hat{D}|} \sum_{(x,y) \in \hat{D}} [L(f(x), y)]
\]

- We will discuss the choices of \( R(f) \) later
How about the loss function?

- Usually, we cannot minimize 0-1 loss
  - It is a combinatorial optimization problem: NP-hard
- Idea: minimizing its upper-bound
How about the loss function?

- Usually, we cannot minimize $0 - 1$ loss.
- It is a combinatorial optimization problem: NP-hard.
- Idea: minimizing its upper-bound.

![Graph showing zero-one loss, logistic loss, and hinge loss.](image-url)
Many choices

- We are minimizing with $R$, $L$, $H$ with your choice

$$\min_{f \in H} R(f) + \frac{1}{|D|} \sum_{(x,y) \in D} [L(f(x), y)]$$

- Let consider $H$ is a set of $d$-dimensional linear function

- $H$ can be parameterized as

$$\{f(x) : w^T x \geq 0\}, w \in R^d$$
Back to Linear model

- We are minimizing with R, L, H with your choice

\[
\min_{f \in H} R(f) + \frac{1}{|\hat{D}|} \sum_{(x,y) \in \hat{D}} [L(f(x), y)]
\]

- Let decide H to be a set of d-dimensional linear function

- H can be parameterized as

\[
\{f(x): w^T x \geq 0\}, w \in \mathbb{R}^d
\]

- We are going to fine the best one based on \( \hat{D} \)
  - i.e., find the best setting of w and b
Rewrite our optimization problem

- Minimizing the empirical loss:
  \[
  \min_{f \in H} R(f) + \frac{1}{|\mathcal{D}|} \sum_{(x,y) \in \mathcal{D}} [L(f(x), y)]
  \]

- Minimizing the empirical loss with linear function
  \[
  \min_{w \in \mathbb{R}^d} R(w) + \frac{1}{|\mathcal{D}|} \sum_{(x,y) \in \mathcal{D}} [L(x, w, y)]
  \]

- What choices of \( R \) and \( L \) we have?
Many choices of loss function ($L$)

Many loss functions exist

- Perceptron loss
  $$L_{Perceptron}(y, x, w) = \max(0, -yw^T x)$$

- Hinge loss (SVM)
  $$L_{Hinge}(y, x, w) = \max(0, 1 - yw^T x)$$

- Exponential loss (AdaBoost)
  $$L_{Exponential}(y, x, w) = e^{-yw^T x}$$

- Logistic loss (logistic regression)
  $$L_{Logistic}(y, x, w) = \log(1 + e^{-yw^T x})$$
Many choices of $R(w)$

- Minimizing the empirical loss with linear function

  $$\min_{w \in \mathbb{R}^d} R(w) + \frac{1}{|\hat{D}|} \sum_{(x,y) \in \hat{D}} [L(x, w, y)]$$

- Prefer simpler model: (how?)
  - Sparse:
    - $R(w) = \#\text{non-zero elements in } w$ (L0 regularizer)
    - $R(w) = \sum_i |w_i|$ (L1 regularizer)
  - Gaussian prior (large margin w/ hinge loss):
    - $R(w) = \sum_i w_i^2 = w^T w$ (L2 regularizer)
Lecture on 1/25
Recap: Linear classifiers

- For now, we consider binary classification
- Given a training set $\mathcal{D} = \{(x, y)\}$, find a linear threshold units classify an example $x$ using the classification rule:

$$\text{sgn}(b + w^T x) = \text{sgn}(b + \sum_i w_i x_i)$$

- $b + w^T x \geq 0 \Rightarrow$ Predict $y = 1$
- $b + w^T x < 0 \Rightarrow$ Predict $y = -1$
Recap: Many choices

- We are minimizing with R, L, H with your choice

\[ \min_{f \in H} R(f) + \frac{1}{|D|} \sum_{(x,y) \in \hat{D}} [L(f(x), y)] \]

- Let consider H is a set of d-dimensional linear function

- H can be parameterized as

\[ \{ f(x): w^T x \geq 0 \}, w \in R^d \]

\[ \min_{w \in R^d} R(w) + \frac{1}{|\hat{D}|} \sum_{(x,y) \in \hat{D}} [L(x, w, y)] \]
Many choices of loss function \((L)\)

<table>
<thead>
<tr>
<th>Loss Function</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perceptron loss</td>
<td>(L_{\text{Perceptron}}(y, x, w) = \max(0, -yw^T x))</td>
</tr>
<tr>
<td>Hinge loss (SVM)</td>
<td>(L_{\text{Hinge}}(y, x, w) = \max(0, 1 - yw^T x))</td>
</tr>
<tr>
<td>Exponential loss (AdaBoost)</td>
<td>(L_{\text{Exponential}}(y, x, w) = e^{-yw^T x})</td>
</tr>
<tr>
<td>Logistic loss (logistic regression)</td>
<td>(L_{\text{Logistic}}(y, x, w) = \log(1 + e^{-yw^T x}))</td>
</tr>
</tbody>
</table>
Which loss are continuous function?
Which loss are smooth function?
Many choices of $R(w)$

- Minimizing the empirical loss **with linear function**

$$\min_{w \in \mathbb{R}^d} R(w) + \frac{1}{|\hat{D}|} \sum_{(x,y) \in \hat{D}} [L(x, w, y)]$$

- Prefer simpler model: (how?)
  - Sparse:
    \[ R(w) = \#\text{non-zero elements in } w \quad (L0 \text{ regularizer}) \]
    \[ R(w) = \sum_i |w_i| \quad (L1 \text{ regularizer}) \]
  - Gaussian prior (large margin w/ hinge loss):
    \[ R(w) = \sum_i w_i^2 = w^T w \quad (L2 \text{ regularizer}) \]
Support Vector Machines
Support Vector Machines (SVMs)

- $R(w)$: l2-loss, $L(w, x, y)$: hinge loss

\[
\min_w \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i (w^T x_i))
\]

- Maximizing margin (why?!!)
The hinge loss

\[ yw^T x \]
Hinge: Incorrect predictions get a linearly increasing penalty with $w^Tx$

Hinge: Penalize predictions even if they are correct, but too close to the margin

Hinge: No penalty if $w^Tx$ is far away from 1 (-1 for negative examples)
Let’s view it from another direction

- SVM learns a model \( w \) on \( D = \{(x_i, y_i)\} \) by solving:

\[
\begin{align*}
\min_{w,b} & \quad \frac{1}{2} w^T w \\
\text{s.t.} & \quad y_i (w^T x_i + b) \geq 1, \forall (x_i, y_i) \in D
\end{align*}
\]

(Hard SVM)

Why the margin is \( \frac{2}{||w||} \)?

\[
\begin{align*}
\arg \max_{w} & \quad \frac{2}{||w||} \\
= & \quad \arg \min_{w} ||w|| \\
= & \quad \arg \min_{w} ||w||^2 \\
= & \quad \arg \min_{w} w^T w
\end{align*}
\]
Soft SVMs

- Data is not separable ⇒ hard SVM fails
  Why?
- Introduce a set of slack variable \( \{\xi_i\} \)
  ⇒ relax the constraints
- Given \( \mathcal{D} = \{(x_i, y_i)\} \), soft SVM solves:

\[
\begin{align*}
\min_{w, b, \xi} & \quad \frac{1}{2} w^T w + C \sum_i \xi_i \\
\text{s.t.} & \quad y_i (w^T x_i + b) \geq 1 - \xi_i; \; \xi_i \geq 0 \quad \forall i
\end{align*}
\]

(Soft SVM)

penalty parameter
An alternative formulation

\[
\min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_i \xi_i
\]

s.t. \( y_i(w^T x_i + b) \geq 1 - \xi_i; \xi_i \geq 0 \quad \forall i \)

- Rewrite the constraints:
  \( \xi_i \geq 1 - y_i(w^T x_i + b); \xi_i \geq 0 \quad \forall i \)

- In the optimum, \( \xi_i = \max(0, 1 - y_i(w^T x_i + b)) \)

- Soft SVM can be rewritten as:
  \[
  \min_{w,b} \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i(w^T x_i + b))
  \]

  ![Diagram](Diagram.png)
An alternative formulation

\[
\min w, b \quad \frac{1}{2} w^T w + C \sum_i \xi_i
\]

- We can simply "b" using the trick
  However, we will add b into the regularization term
  It is often okay, if we have many features

\[
\min 2 \quad \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i(w^T x_i + b))
\]

Regularization term  Empirical loss
Balance between regularization and empirical loss

(a) Training data and an over-fitting classifier

(b) Testing data and an over-fitting classifier
Balance between regularization and empirical loss

(c) Training data and a better classifier

(d) Testing data and a better classifier
Regularized loss minimization

- **L1-Loss SVM**
  \[
  \min_w \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i (w^T x_i))
  \]

- **L2-Loss SVM**
  \[
  \min_w \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i (w^T x_i))^2
  \]

- **Logistic Regression (regularized)**
  \[
  \min_w \frac{1}{2} w^T w + C \sum_i \log(1 + e^{-y_i (w^T x_i)})
  \]

- **Loss over training data + regularizer**
Loss Functions

![Graph showing Logistic regression, L1-loss function, and L2-loss function]

- Logistic regression
- L1-loss function
- L2-loss function

-y w^T x vs. Loss
Logistic Regression

Regression

Logistic regression

Logistic function
logistic function or sigmoid function

- When $z \to \infty$ what is $\sigma(z)$?
- When $z \to -\infty$ what is $\sigma(z)$?
- When $z = 0$ what is $\sigma(z)$?

\[
\sigma(z) = \frac{1}{1 + e^{-z}}
\]
Why sigmoid?

Least squares fit

\[ \sigma(wx + b) \] fit to \( y \)

\[ wx + b \] fit to \( y \)
Probabilistic Interpretation

\[
\min_w \frac{1}{2} w^T w + C \sum_i \log(1 + e^{-y_i(w^T x_i)})
\]

Assume labels are generated using the following probability distribution:

\[
P(y = 1|x, w) = \frac{e^{w^T x}}{1 + e^{w^T x}} = \frac{1}{1 + e^{-w^T x}}
\]

\[
P(y = -1|x, w) = \frac{1}{1 + e^{w^T x}}
\]
How to make prediction?

Predict $y=1$ if $P(y=1|x,w) > p(y=-1|x,w)$
Decision boundary?

\[
P(y = 1|x, w) = \frac{e^{w^T x}}{1 + e^{w^T x}} = \frac{1}{1 + e^{-w^T x}}
\]

\[
P(y = -1|x, w) = \frac{1}{1 + e^{w^T x}}
\]

\[
\log \left( \frac{P(Y = 1 \mid x)}{P(Y = -1 \mid x)} \right) = w^T x + b.
\]

- The decision boundary?

\[
w^T x + b = 0.
\]
Alternative view

- Predict $y=1$ if $P(y=1|x,w) > P(y=-1|x, w)$

- When does this happen?

- \[ \frac{1}{1+\exp(-w^T x)} > 0.5 \]
  \[ \Rightarrow 1 + \exp(-w^T x) < 2 \]
  \[ \Rightarrow \exp(-w^T x) < 1 \]
  \[ \Rightarrow w^T x > 0 \]
Maximum likelihood estimation

- Probabilistic model assumption:

\[
P(y|x, w) = \frac{1}{1 + \exp \left( -y w^T x \right)}
\]

- The log-likelihood of seeing a dataset \( D = \{(x, y)\} \) if the true weight vector was \( w \):

\[
\log P(D|w) = -\sum \log \left( 1 + \exp\left( -y w^T x \right) \right)
\]

\[
P(D|w) = \Pi_i P(y_i|x_i, w) \\
\Rightarrow \log P(D|w) = \sum_i \log P(y_i|x_i, w)
\]
Minimizing negative log-likelihood

- Log likelihood

\[ \log P(D|w) = - \sum \log (1 + \exp(-yw^T x)) \]

- Logistic regression

\[ \min_{w,b} \sum_i \log(1 + e^{-y_i(w^T x_i)}) \]

- Let’s add some prior
  - Simpler is better ⇒ add Gaussian Prior
Add Gaussian Prior

- Simpler is better $\Rightarrow$ add Gaussian Prior
- Suppose each element in $w$ is drawn independently from the normal distribution centered at zero with variance $\sigma^2$
- Bias towards smaller weights

$$P(w_i) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{w_i^2}{2\sigma^2}\right)$$
Regularized Logistic regression

\[ P(w_i) = \frac{1}{\sqrt{2\pi\sigma}} \exp \left( -\frac{w_i^2}{2\sigma^2} \right) \]

- Remember we are in the log space

\[ \log P(w) = -\frac{1}{2\sigma^2} w^T w + \text{constant terms} \]

- Put them together
  - \( P(w|D) \propto P(w, D) = P(D | w)P(w) \)
  - Learning:
    Find weight vector by maximizing the posterior distribution \( P(w | D) \)
Maximum a posteriori estimation

- Put them together
  - $P(w|D) \propto P(w, D) = P(D |w)P(w)$

- Learning: Find weight vector by maximizing the posterior distribution $P(w | D)$

\[
\log P(D, w) = -\frac{1}{2\sigma^2}w^T w - \sum_i \log (1 + \exp(-y_i w^T x_i))
\]

\[
\min_w \frac{1}{2}w^T w + C \sum_i \log(1 + e^{-y_i w^T x_i})
\]
Regularized loss minimization

- **L1-Loss SVM**
  
  $$\min_w \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i(w^T x_i))$$

- **L2-Loss SVM**
  
  $$\min_w \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i(w^T x_i))^2$$

- **Logistic Regression (regularized)**
  
  $$\min_w \frac{1}{2} w^T w + C \sum_i \log(1 + e^{-y_i(w^T x_i)})$$

- **Loss over training data + regularizer**
How to learn
(how to optimize the objective function?)

$$\min_w \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i (w^T x_i))$$

- This function is convex
- Many convex optimization methods can be used
  - Stochastic (sub)-gradient descent
  - Coordinate descent methods
  - Newton methods
  - LBFGS
Convexity
Non convex minimization is hard

- You may end up with some local minimum
Convex optimization is relatively easy

- Ensure that there are no local minima
- Note: need special design for functions that are not differentiable (e.g., hinge loss)
Different optimization techniques

Some methods (e.g., SGD, CD) are fast in the early stage of optimization

Some methods (e.g., Newton methods) converge faster

Results from http://www.cs.virginia.edu/~kc2wc/papers/ChangHsLi08.pdf
Gradient descent

$J(w)$

$w^4 \quad w^3 \quad w^2 \quad w^1$
Why not gradient descent?

- For some functions, gradient may not exist

\[ \min_w \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i(w^T x_i)) \]

- Solution: use sub-gradient

\[
\begin{align*}
  f(w) &= \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i w^T x_i) \\
  &= \frac{1}{|D|} \sum_i \left( \frac{1}{2} w^T w + C' \max(0, 1 - y_i w^T x_i) \right) \\
  &\equiv \frac{1}{|D|} \sum_i f_i(w) \\
  \nabla f(w) &= \frac{1}{|D|} \sum_i \nabla f_i(w) \\
  \nabla f_i(w) &\equiv \begin{cases} 
    w & \text{if } y_i(w^T x_i) > 1 \\
    w - C' y_i x_i & \text{otherwise}
  \end{cases}
\]

\[ C' = |D| \times C \]
Stochastic gradient descent

\[
f(w) = \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i w^T x_i)
= \frac{1}{|D|} \sum_i \left( \frac{1}{2} w^T w + C' \max(0, 1 - y_i w^T x_i) \right)
\equiv f_i(w)
\]
\[
\nabla f(w) = \frac{1}{|D|} \sum_i \nabla f_i(w) = E_{i \sim D} \nabla f_i(w)
\]

- Approximate the true gradient by a gradient at a single example at a time

Repeat until converge:
Randomly pick one sample \((x_i, y_i)\)
Update \(w \leftarrow w - \eta \nabla f_i(w)\)
Stochastic Sub-gradient Descent

Given a training set \( \mathcal{D} = \{(x,y)\} \)

1. Initialize \( \mathbf{w} \leftarrow \mathbf{0} \in \mathbb{R}^n \)
2. For epoch 1...\( T \):
3. For \((x,y)\) in \( \mathcal{D} \):
4. Update \( \mathbf{w} \leftarrow \mathbf{w} - \eta \nabla f(\mathbf{w}) \)
5. Return \( \mathbf{w} \)

\[
f(\mathbf{w}) \equiv \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_i \max(0, 1 - y_i (\mathbf{w}^T \mathbf{x}_i))
\]
Stochastic (sub)-gradient descent for SVM

Given a training set \( \mathcal{D} = \{(x,y)\} \)

1. **Initialize** \( w \leftarrow 0 \in \mathbb{R}^n \)

2. For epoch 1...\( T \):

3. For (\( x, y \)) in \( \mathcal{D} \):

4. if \( y(w^\top x) < 1 \)

5. \[ w \leftarrow (1 - \eta)w + \eta C yx \]

6. else

7. \[ w \leftarrow (1 - \eta)w \]

8. Return \( w \)
The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x,y)\}$

1. **Initialize** $w \leftarrow 0 \in \mathbb{R}^n$

2. **For** epoch $1...T$:

3. **For** $(x,y)$ **in** $\mathcal{D}$:

4. **if** $y(w^T x) < 0$

5. $w \leftarrow w + \eta y x$

6. **Return** $w$

**Prediction:** $y^{\text{test}} \leftarrow \text{sgn}(w^T x^{\text{test}})$
The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x,y)\}$

1. Initialize $\mathbf{w} \leftarrow 0 \in \mathbb{R}^n$
2. For epoch 1...$T$:
3. For $(x,y)$ in $\mathcal{D}$:
4. if $y(\mathbf{w}^\top x) < 0$
5. $\mathbf{w} \leftarrow \mathbf{w} + \eta y x$
6. Return $\mathbf{w}$

Prediction: $y^{\text{test}} = \sum_i \max(0, 1 - y_i (\mathbf{w}^\top x_i))$

A General Formula

\[ \hat{y} = \text{argmax}_{y \in \mathcal{Y}} f(y; w, x) \]

- **Inference/Test:** given \( w, x \), solve argmax
- **Learning/Training:** find a good \( w \)
- **Today:** \( x \in \mathbb{R}^n, \mathcal{Y} = \{-1, 1\} \) (binary classification)
Binary Linear Classifiers

\[ \hat{y} = \arg\max_{y \in \mathcal{Y}} f(y; w, x) \]

- \( x \in \mathbb{R}^n, \mathcal{Y} = \{-1,1\} \)
- \( f(y; w, x) \overset{\text{def}}{=} y(w^\top x + b) = y(\sum_i w_i x_i + b) \)

- \( \arg\max_{y \in \mathcal{Y}} f(y; w, x) = \begin{cases} 1, w^\top x + b \geq 0 \\ -1, w^\top x + b < 0 \end{cases} = \text{sgn}(w^\top x + b) \)

(break ties arbitrarily)