Binary Classification with Linear Models

CMSC 422
MARINE CARPUAT
marine@cs.umd.edu

Figures credit: Piyush Rai
Topics

• Linear Models
  – Loss functions
  – Regularization

• Gradient Descent

• Calculus refresher
  – Convexity
  – Gradients

[CIML Chapter 6]
Binary classification via hyperplanes

- A classifier is a hyperplane \((w, b)\).
- At test time, we check on what side of the hyperplane examples fall:
  \[
  \hat{y} = \text{sign}(w^T x + b)
  \]
- This is a **linear classifier**
  - Because the prediction is a linear combination of feature values \(x\).
**Task: Binary Classification**

**Given:**

1. An input space $\mathcal{X}$
2. An unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \{-1, +1\}$

**Compute:** A function $f$ minimizing: $\mathbb{E}_{(x,y) \sim \mathcal{D}}[f(x) \neq y]$
Learning a Linear Classifier as an Optimization Problem

**Objective function**

\[
\min_{w, b} L(w, b)
\]

**Loss function**
measures how well classifier fits training data

**Regularizer**
prefers solutions that generalize well
Learning a Linear Classifier as an Optimization Problem

\[
\min_{\mathbf{w}, b} L(\mathbf{w}, b) = \min_{\mathbf{w}, b} \sum_{n=1}^{N} \mathbb{I}(y_n(\mathbf{w}^T \mathbf{x}_n + b) < 0) + \lambda R(\mathbf{w}, b)
\]

- **Problem:** The 0-1 loss above is NP-hard to optimize

- **Solution:** Different loss function approximations and regularizers lead to specific algorithms (e.g., perceptron, support vector machines, logistic regression, etc.)
The 0-1 Loss

- Small changes in w, b can lead to big changes in the loss value
- 0-1 loss is non-smooth, non-convex
Calculus refresher:
Smooth functions, convex functions
Approximating the 0-1 loss with surrogate loss functions

- Examples (with $b = 0$)
  - Hinge loss $\left[1 - y_n w^T x_n\right]_+ = \max\{0, 1 - y_n w^T x_n\}$
  - Log loss $\log[1 + \exp(-y_n w^T x_n)]$
  - Exponential loss $\exp(-y_n w^T x_n)$

- All are convex upper-bounds on the 0-1 loss
Approximating the 0-1 loss with surrogate loss functions

• Examples (with $b = 0$)
  – Hinge loss
  – Log loss $\log(1 + \exp(-y_n w^T x_n))$
  – Exponential loss $\exp(-y_n w^T x_n)$

• Q: Which of these loss functions is not smooth?
Approximating the 0-1 loss with surrogate loss functions

- Examples (with $b = 0$)
  - Hinge loss: $[1 - y_n w^T x_n]_+ = \max\{0, 1 - y_n w^T x_n\}$
  - Log loss: $\log[1 + \exp(-y_n w^T x_n)]$
  - Exponential loss: $\exp(-y_n w^T x_n)$

- Q: Which of these loss functions is most sensitive to outliers?
Casting Linear Classification as an Optimization Problem

Objective function

Loss function measures how well classifier fits training data

Regularizer prefers solutions that generalize well

\[
\min_{w, b} L(w, b) = \min_{w, b} \sum_{n=1}^{N} \mathbb{I}(y_n (w^T x_n + b) < 0) + \lambda R(w, b)
\]

\(\mathbb{I}(\cdot)\) Indicator function: 1 if \(\cdot\) is true, 0 otherwise

The loss function above is called the 0-1 loss
The regularizer term

• Goal: find simple solutions (inductive bias)

• Ideally, we want most entries of $w$ to be zero, so prediction depends only on a small number of features.

• Formally, we want to minimize:

$$R^{cnt}(w, b) = \sum_{d=1}^{D} \mathbb{I}(w_d \neq 0)$$

• That’s NP-hard, so we use approximations instead.
  – E.g., we encourage $w_d$’s to be small
Norm-based Regularizers

- $l_p$ norms can be used as regularizers

$$
\|w\|_2^2 = \sum_{d=1}^{D} w_d^2 \\
\|w\|_1 = \sum_{d=1}^{D} |w_d| \\
\|w\|_p = \left(\sum_{d=1}^{D} w_d^p\right)^{1/p}
$$

Contour plots for $p = 2$, $p = 1$, $p < 1$
Norm-based Regularizers

- $l_p$ norms can be used as regularizers
- Smaller $p$ favors sparse vectors $w$
  - i.e. most entries of $w$ are close or equal to 0
- $l_2$ norm: convex, smooth, easy to optimize
- $l_1$ norm: encourages sparse $w$, convex, but not smooth at axis points
- $p < 1$: norm becomes non-convex and hard to optimize
Casting Linear Classification as an Optimization Problem

Objective function

Loss function measures how well classifier fits training data

Regularizer prefers solutions that generalize well

\[ \min_{w, b} L(w, b) = \min_{w, b} \sum_{n=1}^{N} \mathbb{I}(y_n(w^T x_n + b) < 0) + \lambda R(w, b) \]

\[ \mathbb{I}(.) \] Indicator function: 1 if (.) is true, 0 otherwise

The loss function above is called the 0-1 loss
What is the perceptron optimizing?

<table>
<thead>
<tr>
<th>Algorithm 5</th>
<th>PerceptronTrain(D, MaxIter)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: $w_d \leftarrow 0$, for all $d = 1 \ldots D$</td>
<td>// initialize weights</td>
</tr>
<tr>
<td>2: $b \leftarrow 0$</td>
<td>// initialize bias</td>
</tr>
<tr>
<td>3: for iter = 1 \ldots MaxIter do</td>
<td></td>
</tr>
<tr>
<td>4: for all $(x,y) \in D$ do</td>
<td></td>
</tr>
<tr>
<td>5: $a \leftarrow \sum_{d=1}^{D} w_d \times_d + b$</td>
<td>// compute activation for this example</td>
</tr>
<tr>
<td>6: if $ya \leq 0$ then</td>
<td></td>
</tr>
<tr>
<td>7: $w_d \leftarrow w_d + yx_d$, for all $d = 1 \ldots D$</td>
<td>// update weights</td>
</tr>
<tr>
<td>8: $b \leftarrow b + y$</td>
<td>// update bias</td>
</tr>
<tr>
<td>9: end if</td>
<td></td>
</tr>
<tr>
<td>10: end for</td>
<td></td>
</tr>
<tr>
<td>11: end for</td>
<td></td>
</tr>
<tr>
<td>12: return $w_0, w_1, \ldots, w_D, b$</td>
<td></td>
</tr>
</tbody>
</table>

- Loss function is a variant of the hinge loss
  \[
  \max\{0, -y_n(w^T x_n + b)\}
  \]
Recap: Linear Models

- General framework for binary classification
- Cast learning as optimization problem
- Optimization objective combines 2 terms
  - loss function: measures how well classifier fits training data
  - Regularizer: measures how simple classifier is
- Does not assume data is linearly separable
- Lets us separate model definition from training algorithm
Calculus refresher: Gradients
Gradient descent

• A general solution for our optimization problem

\[
\min_{w,b} L(w, b) = \min_{w,b} \sum_{n=1}^{N} \mathbb{I}(y_n(w^T x_n + b) < 0) + \lambda R(w, b)
\]

Idea: take iterative steps to update parameters in the direction of the gradient
Gradient descent algorithm

Algorithm 22 **GradientDescent** \((F, K, \eta_1, \ldots)\)

1. \(z^{(0)} \leftarrow \langle 0, 0, \ldots, 0 \rangle\)  
   // initialize variable we are optimizing
2. **for** \(k = 1 \ldots K\) **do**
3.  \(g^{(k)} \leftarrow \nabla_z F|_{z^{(k-1)}}\)  
   // compute gradient at current location
4.  \(z^{(k)} \leftarrow z^{(k-1)} - \eta^{(k)} g^{(k)}\)  
   // take a step down the gradient
5. **end for**
6. **return** \(z^{(K)}\)
Recap: Linear Models

• General framework for binary classification
• Cast learning as optimization problem
• Optimization objective combines 2 terms
  – loss function: measures how well classifier fits training data
  – Regularizer: measures how simple classifier is
• Does not assume data is linearly separable
• Lets us separate model definition from training algorithm (Gradient Descent)