Linear classifiers: Outline

- Examples of classification models: nearest neighbor, linear
- Empirical loss minimization framework
- Linear classification models
  1. Linear regression
  2. Logistic regression
  3. Perceptron loss
  4. SVM loss
- General recipe: Data loss, regularization
- Multi-class classification
  1. Multi-class perceptron
  2. Multi-class SVM
  3. Softmax
Recall: The shape of logistic loss

\[ l(w, x_i, y_i) = - \log \sigma(y_i w^T x_i) \]
Let’s define the *perceptron hinge loss*:

\[
l(w, x_i, y_i) = \max(0, -y_i w^T x_i)
\]
Perceptron

• Let’s define the *perceptron hinge loss*:

\[
l(w, x_i, y_i) = \max(0, -y_i w^T x_i)
\]

• Training: find \( w \) that minimizes

\[
\hat{L}(w) = \frac{1}{n} \sum_{i=1}^{n} l(w, x_i, y_i) = \frac{1}{n} \sum_{i=1}^{n} \max(0, -y_i w^T x_i)
\]

• Once again, there is no closed-form solution, so let’s go straight to SGD
Deriving the perceptron update

- Let’s differentiate the perceptron hinge loss:

\[ l(w, x_i, y_i) = \max(0, -y_i w^T x_i) \]

(Strictly speaking, this loss is not differentiable, so we need to find a sub-gradient)
Deriving the perceptron update

- Let’s differentiate the perceptron hinge loss:

\[ l(w, x_i, y_i) = \max(0, -y_i w^T x_i) \]
\[ \nabla l(w, x_i, y_i) = -\mathbb{I}[y_i w^T x_i < 0]y_i x_i \]

\[ \frac{d}{da} \max(0, a) = \]
Deriving the perceptron update

- Let’s differentiate the perceptron hinge loss:
  \[
  l(w, x_i, y_i) = \max(0, -y_i w^T x_i)
  \]
  \[
  \nabla l(w, x_i, y_i) = -\mathbb{I}[y_i w^T x_i < 0]y_i x_i
  \]

- We also used the chain rule:
  \[
  [g_2(g_1(a))]' = g_2'(g_1(a))g_1'(a)
  \]
Deriving the perceptron update

• Let’s differentiate the perceptron hinge loss:
  \[
l(w, x_i, y_i) = \max(0, -y_i w^T x_i)
  \]
  \[
  \nabla l(w, x_i, y_i) = -\mathbb{I}[y_i w^T x_i < 0] y_i x_i
  \]

• Corresponding SGD update \((w \leftarrow w - \eta \nabla l(w, x_i, y_i))\):
  \[
  w \leftarrow w + \eta \mathbb{I}[y_i w^T x_i < 0] y_i x_i
  \]

  • If \(x_i\) is correctly classified: do nothing
  • If \(x_i\) is incorrectly classified: \(w \leftarrow w + \eta y_i x_i\)
Understanding the perceptron update rule

- **Perceptron update rule**: If $y_i \neq \text{sgn}(w^T x_i)$ then update weights:
  \[
  w \leftarrow w + \eta y_i x_i
  \]

- The raw response of the classifier changes to
  \[
  w^T x_i + \eta y_i \|x_i\|^2
  \]

- How does the response change if $y_i = 1$?
  - The response $w^T x_i$ is initially *negative* and will be *increased*

- How does the response change if $y_i = -1$?
  - The response $w^T x_i$ is initially *positive* and will be *decreased*
Linear classifiers: Outline

- Example classification models: nearest neighbor, linear
- Empirical loss minimization
- Linear classification models
  1. Linear regression (least squares)
  2. Logistic regression
  3. Perceptron loss
  4. Support vector machine (SVM) loss
Support vector machines

• When the data is linearly separable, which of the many possible solutions should we prefer?

  • **Perceptron training algorithm:**
    no special criterion, solution depends on initialization
Support vector machines

- When the data is linearly separable, which of the many possible solutions should we prefer?

  - **Perceptron training algorithm**: no special criterion, solution depends on initialization

  - **SVM criterion**: maximize the *margin*, or distance between the hyperplane and the closest training example
Finding the maximum margin hyperplane

• We want to maximize the margin, or distance between the hyperplane \( w^T x = 0 \) and the closest training example \( x_0 \).

• This distance is given by \( \frac{|w^T x_0|}{\|w\|} \).

  (for derivation see, e.g., here)

• Assuming the data is linearly separable, we can fix the scale of \( w \) so that \( y_i w^T x_i = 1 \) for support vectors and \( y_i w^T x_i \geq 1 \) for all other points.

• Then the margin is given by \( \frac{1}{\|w\|} \).
Finding the maximum margin hyperplane

- We want to maximize margin \( \frac{1}{\|w\|} \) while correctly classifying all training data: \( y_i w^T x_i \geq 1 \)
- Equivalent problem:

\[
\min_w \frac{1}{2} \|w\|^2 \quad \text{s. t.} \quad y_i w^T x_i \geq 1 \quad \forall i
\]

- This is a quadratic objective with linear constraints: convex optimization problem, global optimum can be found using well-studied methods
“Soft margin” formulation

- What about non-separable data?
- And even for separable data, we may prefer a larger margin with a few constraints violated

Source
“Soft margin” formulation

- What about non-separable data?
- And even for separable data, we may prefer a larger margin with a few constraints violated
“Soft margin” formulation

• Penalize margin violations using SVM hinge loss:

$$\min_w \frac{\lambda}{2} \|w\|^2 + \sum_{i=1}^{n} \max[0, 1 - y_i w^T x_i]$$
“Soft margin” formulation

- Penalize margin violations using SVM hinge loss:

\[
\min_w \frac{\lambda}{2} \|w\|^2 + \sum_{i=1}^{n} \max[0,1 - y_i w^T x_i]
\]
“Soft margin” formulation

• Penalize margin violations using SVM hinge loss:

\[
\min_w \frac{\lambda}{2} \|w\|^2 + \sum_{i=1}^{n} \max[0, 1 - y_i w^T x_i]
\]

Maximize margin – a.k.a. regularization

Minimize misclassification loss
SGD update for SVM

\[ l(w, x_i, y_i) = \frac{\lambda}{2n} \|w\|^2 + \max[0, 1 - y_i w^T x_i] \]

\[ \nabla l(w, x_i, y_i) = \frac{\lambda}{n} w - \mathbb{I}[y_i w^T x_i < 1] y_i x_i \]

Recall: \( \frac{d}{da} \max(0, a) = \mathbb{I}[a > 0] \)
SGD update for SVM

\[
l(w, x_i, y_i) = \frac{\lambda}{2n} \|w\|^2 + \max[0, 1 - y_i w^T x_i]
\]

\[
\nabla l(w, x_i, y_i) = \frac{\lambda}{n} w - \mathbb{I}[y_i w^T x_i < 1] y_i x_i
\]

- SGD update:
  - If \( y_i w^T x_i \geq 1 \): \( w \leftarrow w - \eta \frac{\lambda}{n} w \)
  - If \( y_i w^T x_i < 1 \): \( w \leftarrow w + \eta \left( y_i x_i - \frac{\lambda}{n} w \right) \)

Linear classifiers: Outline

• Examples of classification models: nearest neighbor, linear
• Empirical loss minimization framework
• Linear classification models
  1. Linear regression
  2. Logistic regression
  3. Perceptron training algorithm
  4. Support vector machines
• General recipe: data loss, regularization
General recipe

- Find parameters $w$ that minimize the sum of a *regularization loss* and a *data loss*:

$$
\hat{L}(w) = \lambda R(w) + \frac{1}{n} \sum_{i=1}^{n} l(w, x_i, y_i)
$$

**empirical loss**  **regularization**  **data loss**

**L2 regularization:**

$$
R(w) = \frac{1}{2} \|w\|_2^2
$$
Closer look at L2 regularization

- Regularized objective: \( \hat{L}(w) = \frac{\lambda}{2} \|w\|^2 + \sum_{i=1}^{n} l(w, x_i, y_i) \)
- Gradient of objective:
  \[
  \nabla \hat{L}(w) = \lambda w + \sum_{i=1}^{n} \nabla l(w, x_i, y_i)
  \]
- SGD update:
  \[
  w \leftarrow w - \eta \left( \frac{\lambda}{n} w + \nabla l(w, x_i, y_i) \right)
  \]
  \[
  w \leftarrow \left(1 - \frac{\eta \lambda}{n}\right)w - \eta \nabla l(w, x_i, y_i)
  \]
- Interpretation: weight decay
General recipe

- Find parameters $w$ that minimize the sum of a regularization loss and a data loss:

$$
\hat{L}(w) = \lambda R(w) + \frac{1}{n} \sum_{i=1}^{n} l(w, x_i, y_i)
$$

- **L2 regularization:**
  $$
  R(w) = \frac{1}{2} \|w\|_2^2
  $$

- **L1 regularization:**
  $$
  R(w) = \|w\|_1
  $$
Closer look at L1 regularization

- Regularized objective:
  \[ \hat{L}(w) = \lambda \|w\|_1 + \sum_{i=1}^{n} l(w, x_i, y_i) \]
  \[ = \lambda \sum_{d} |w^{(d)}| + \sum_{i=1}^{n} l(w, x_i, y_i) \]

- Gradient: \( \nabla \hat{L}(w) = \lambda \text{sgn}(w) + \sum_{i=1}^{n} \nabla l(w, x_i, y_i) \)
  (here \( \text{sgn} \) is an elementwise function)

- SGD update:
  \[ w \leftarrow w - \frac{\eta \lambda}{n} \text{sgn}(w) - \eta \nabla l(w, x_i, y_i) \]

- Interpretation: encouraging sparsity
Linear classifiers: Outline

• Examples of classification models: nearest neighbor, linear
• Empirical loss minimization framework
• Linear classification models
  1. Linear regression
  2. Logistic regression
  3. Perceptron training algorithm
  4. Support vector machines
• General recipe: data loss, regularization
• Multi-class classification
  1. Multi-class perceptron
  2. Multi-class SVM
  3. Softmax
One-vs-all classification

- Let $y \in \{1, \ldots, C\}$
- Learn $C$ scoring functions $f_1, f_2, \ldots, f_C$
- Classify $x$ to class $\hat{y} = \text{argmax}_c f_c(x)$
- Let’s start with multi-class perceptrons:
  \[ f_c(x) = w_c^T x \]
Multi-class perceptrons

- Multi-class perceptrons: $f_c(x) = w_c^T x$
- Let $W$ be the matrix with rows $w_c$
- What loss should we use for multi-class classification?

Figure source: Stanford 231n
Multi-class perceptrons

- Multi-class perceptrons: $f_c(x) = w_c^T x$
- Let $W$ be the matrix with rows $w_c$
- What loss should we use for multi-class classification?
- For $(x_i, y_i)$, let the loss be the *sum of hinge losses* associated with predictions for all *incorrect* classes:

$$l(W, x_i, y_i) = \sum_{c \neq y_i} \max[0, w_c^T x_i - w_{y_i}^T x_i]$$

Score for correct class ($y_i$) has to be greater than the score for the incorrect class ($c$)
Multi-class perceptrons

\[ l(W, x_i, y_i) = \sum_{c \neq y_i} \max[0, w_c^T x_i - w_{y_i}^T x_i] \]

- Gradient w.r.t. \( w_{y_i} \):
  
  \[- \sum_{c \neq y_i} \mathbb{1} [w_c^T x_i > w_{y_i}^T x_i] x_i \]

Recall: \( \frac{d}{da} \max(0, a) = \mathbb{I}[a > 0] \)
Multi-class perceptrons

$$l(W, x_i, y_i) = \sum_{c \neq y_i} \max[0, w_c^T x_i - w_{y_i}^T x_i]$$

- Gradient w.r.t. $w_{y_i}$:
  $$- \sum_{c \neq y_i} \mathbb{I}[w_c^T x_i > w_{y_i}^T x_i] x_i$$

- Gradient w.r.t. $w_c$, $c \neq y_i$:
  $$\mathbb{I}[w_c^T x_i > w_{y_i}^T x_i] x_i$$

- Update rule: for each $c$ s.t. $w_c^T x_i > w_{y_i}^T x_i$:
  $$\begin{align*}
w_{y_i} &\leftarrow w_{y_i} + \eta x_i \\
w_c &\leftarrow w_c - \eta x_i
\end{align*}$$
Multi-class perceptrons

- Update rule: for each $c$ s.t. $w_c^T x_i > w_{y_i}^T x_i$:
  
  $w_{y_i} \leftarrow w_{y_i} + \eta x_i$
  
  $w_c \leftarrow w_c - \eta x_i$

- Is this equivalent to training $C$ independent one-vs-all classifiers?

<table>
<thead>
<tr>
<th></th>
<th>Independent</th>
<th>Multi-class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat score:</td>
<td>65.1</td>
<td>Do nothing</td>
</tr>
<tr>
<td>Dog score:</td>
<td>101.4</td>
<td>Decrease</td>
</tr>
<tr>
<td>Ship score:</td>
<td>24.9</td>
<td>Decrease</td>
</tr>
</tbody>
</table>
Multi-class SVM

- Recall single-class SVM loss:
  \[ l(w, x_i, y_i) = \frac{\lambda}{2n} \|w\|^2 + \max[0, 1 - y_i w^T x_i] \]

- Generalization to multi-class:
  \[ l(W, x_i, y_i) = \frac{\lambda}{2n} \|W\|^2 + \sum_{c \neq y_i} \max[0, 1 - w_{y_i}^T x_i + w_c^T x_i] \]

Score for correct class has to be greater than the score for the incorrect class by at least a margin of 1.

Source: Stanford 231n
Multi-class SVM

\[ l(W, x_i, y_i) = \frac{\lambda}{2n} \|W\|^2 + \sum_{c \neq y_i} \max[0, 1 - w_{y_i}^T x_i + w_c^T x_i] \]

- Gradient w.r.t. \( w_{y_i} \):
  \[ \frac{\lambda}{n} w_{y_i} - \sum_{c \neq y_i} \mathbb{I}[w_{y_i}^T x_i - w_c^T x_i < 1] x_i \]

- Gradient w.r.t. \( w_c, c \neq y_i \):
  \[ \frac{\lambda}{n} w_c + \mathbb{I}[w_{y_i}^T x_i - w_c^T x_i < 1] x_i \]

- Update rule (almost* equivalent to above):
  - For each \( c \neq y_i \) s.t. \( w_{y_i}^T x_i - w_c^T x_i < 1 \): \( w_{y_i} \leftarrow w_{y_i} + \eta x_i, \; w_c \leftarrow w_c - \eta x_i \)
  - For \( c = 1, ..., C \): \( w_c \leftarrow \left(1 - \eta \frac{\lambda}{n}\right) w_c \)
Announcements

- Assignment 1 is out, due Tuesday, February 14
- Quiz 1 will be available 9AM this Thursday, February 9, through 9AM Monday, February 13
Review: Three ways to think about linear classifiers

**Algebraic Viewpoint**

\[ f_W(x) = Wx + b \]

**Visual Viewpoint**

One template per class

**Geometric Viewpoint**

Hyperplanes cutting up space

Source: J. Johnson
Review: General recipe for training classifiers

- Find parameters $w$ that minimize the sum of a regularization loss and a data loss:

$$\hat{L}(w) = \lambda R(w) + \frac{1}{n} \sum_{i=1}^{n} l(w, x_i, y_i)$$

empirical loss  
regularization  
data loss

L2 regularization:
$$R(w) = \frac{1}{2} \|w\|_2^2$$

L1 regularization:
$$R(w) = \|w\|_1$$
Last week: Linear classifiers

- Examples of classification models: nearest neighbor, linear
- Empirical loss minimization framework
- Linear classification models
  1. Linear regression
  2. Logistic regression
  3. Perceptron training algorithm
  4. Support vector machines
- General recipe: data loss, regularization
- Multi-class classification
  - Multi-class perceptrons
  - Multi-class SVM
  - Softmax
Softmax

- We want to squash the vector of responses \((f_1, ..., f_c)\) into a vector of “probabilities”:

\[
\text{softmax}(f_1, ..., f_c) = \left( \frac{\exp(f_1)}{\sum_j \exp(f_j)}, ..., \frac{\exp(f_c)}{\sum_j \exp(f_j)} \right)
\]

- The outputs are between 0 and 1 and sum to 1
- If one of the inputs (logits) is much larger than the others, then the corresponding softmax value will be close to 1 and others will be close to 0
Softmax and sigmoid

• For two classes:

\[
\text{softmax}(f, -f) = \left( \frac{\exp(f)}{\exp(f) + \exp(-f)}, \frac{\exp(-f)}{\exp(f) + \exp(-f)} \right) \\
= \left( \frac{1}{1 + \exp(-2f)}, \frac{1}{\exp(2f) + 1} \right) \\
= (\sigma(2f), \sigma(-2f))
\]

• Thus, softmax is the generalization of sigmoid for more than two classes
Cross-entropy loss

- It is natural to use negative log likelihood loss with softmax:
  \[
  l(W, x_i, y_i) = - \log P_W(y_i | x_i) = -\log \left( \frac{\exp(w^T_{y_i} x_i)}{\sum_j \exp(w^T_j x_i)} \right)
  \]

- This is also the cross-entropy between the “empirical” distribution \( \hat{P}(c | x_i) = \mathbb{I}[c = y_i] \) and “estimated” distribution \( P_W(c | x_i) \):
  \[
  - \sum_c \hat{P}(c | x_i) \log P_W(c | x_i)
  \]
SVM loss vs. cross-entropy loss

Correct class is the third one (blue)

Source: Stanford 231n
SGD with cross-entropy loss

\[ l(W, x_i, y_i) = -\log P_W(y_i|x_i) = -\log \left( \frac{\exp(w_{y_i}^T x_i)}{\sum_j \exp(w_j^T x_i)} \right) \]

\[ = -w_{y_i}^T x_i + \log \left( \sum_j \exp(w_j^T x_i) \right) \]

- **Gradient w.r.t.** \( w_{y_i} \):
  \[-x_i + \frac{\exp(w_{y_i}^T x_i) x_i}{\sum_j \exp(w_j^T x_i)} = (P_W(y_i|x_i) - 1)x_i \]

- **Gradient w.r.t.** \( w_c, c \neq y_i \):
  \[-x_i + \frac{\exp(w_c^T x_i) x_i}{\sum_j \exp(w_j^T x_i)} = P_W(c|x_i)x_i \]
SGD with cross-entropy loss

- Gradient w.r.t. $w_{y_i}$: $(P_W(y_i|x_i) - 1)x_i$

- Gradient w.r.t. $w_c$, $c \neq y_i$: $P_W(c|x_i)x_i$

- Update rule:
  - For $y_i$:
    $$w_{y_i} \leftarrow w_{y_i} + \eta(1 - P_W(y_i|x_i))x_i$$
  - For $c \neq y_i$:
    $$w_c \leftarrow w_c - \eta P_W(c|x_i)x_i$$
Softmax trick: Avoiding overflow

- Exponentiated values $\exp(f_c)$ can become very large and cause overflow.
- Note that adding the same constant to all softmax inputs (logits) does not change the output of the softmax:

$$\frac{\exp(f_c)}{\sum_j \exp(f_j)} = \frac{K \exp(f_c)}{\sum_j K \exp(f_j)} = \frac{\exp(f_c + \log K)}{\sum_j \exp(f_j + \log K)}$$

- Then we can let $\log K = -\max_j f_j$ (i.e., make largest input to softmax be 0)
Softmax trick: Temperature scaling

- Suppose we divide every input to the softmax by the same constant $T$:

$$\text{softmax}(f_1, \ldots, f_c; T) = \left( \frac{\exp(f_1/T)}{\sum_j \exp(f_j/T)}, \ldots, \frac{\exp(f_c/T)}{\sum_j \exp(f_j/T)} \right)$$

- Prior to normalization, each entry $\exp(f_i)$ is raised to the power $1/T$
- If $T$ is close to 0, the largest entry will dominate and output distribution will approach one-hot
- If $T$ is high, output distribution will approach uniform

Source
Softmax trick: Temperature scaling

- Low temperature: More concentrated distribution
- Higher temperature: More uniform distribution

Figure source
Softmax trick: Label smoothing

- Recall: cross-entropy loss measures the difference between the “observed” label distribution $\hat{P}(c|x_i)$ and “estimated” distribution $P_W(c|x_i)$:

$$- \sum_c \hat{P}(c|x_i) \log P_W(c|x_i)$$

“Hard” prediction targets

Empirical distribution $\hat{P}(c|x_i)$

- $P(\text{correct class } | x_i) = 1$
- $P(\text{incorrect class } | x_i) = 0$

Estimated distribution $P_W(c|x_i)$
Softmax trick: Label smoothing

- Recall: cross-entropy loss measures the difference between the “observed” label distribution $\hat{P}(c|x_i)$ and “estimated” distribution $P_W(c|x_i)$:

$$ - \sum_c \hat{P}(c|x_i) \log P_W(c|x_i) $$

“Soft” prediction targets

Empirical distribution $\hat{P}(c|x_i)$

- $P(\text{correct class } | x_i) = 1 - \epsilon$
- $P(\text{incorrect class } | x_i) = \frac{\epsilon}{c-1}$

Estimated distribution $P_W(c|x_i)$
Softmax trick: Label smoothing

- When using softmax loss, replace hard 1 and 0 prediction targets with “soft” targets of $1 - \epsilon$ and $\frac{\epsilon}{C-1}$

- Why is this a good idea?
  - A form of regularization to avoid overly confident predictions, account for label noise