Machine Learning in Imaging

BME 548L
Roarke Horstmeyer

Lecture 7: Gradient descent and going beyond linear classification
Summary of machine learning pipeline:

1. Network Training

What we need for network training:

1. Labeled examples
Summary of machine learning pipeline:

1. Network Training

What we need for network training:

1. Labeled examples

E.g., images of 1’s and 5’s with labels:

\[ x_1 = \begin{pmatrix} 1 \end{pmatrix}, \quad y_1 = +1 \]
\[ x_2 = \begin{pmatrix} 4 \end{pmatrix}, \quad y_2 = -1 \]
\[ \vdots \]
Summary of machine learning pipeline:

1. Network Training

What we need for network training:

1. Labeled examples
2. A model and loss function
Summary of machine learning pipeline:

1. Network Training

What we need for network training:

1. Labeled examples
2. A model and loss function
3. A way to minimize the loss function $L$
Summary of machine learning pipeline:

2. Network Testing

What we need for network testing:
Summary of machine learning pipeline:

2. Network Testing

What we need for network testing:

4. **Unique** labeled test data
Summary of machine learning pipeline:

2. Network Testing

Test error

\[ L_{\text{out}}(y, y^*) \]

What we need for network testing:

4. *Unique* labeled test data
5. Evaluation of model error

Test Data *not* from Training Dataset
Illustration of features

\[ \mathbf{x} = (x_0, x_1, x_2) \quad x_1: \text{intensity} \quad x_2: \text{symmetry} \]
Linear classification:

Use MSE error model

$$L = \frac{1}{N} \sum_{i=1}^{N} (w^T x_i - y_i)^2$$

Where labels determined by thresholding

$$f(x_i) = y_i^* = \text{sgn}(w^T x_i)$$

$$\text{sgn}(x) := \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ 1 & \text{if } x > 0. \end{cases}$$
Why does linear regression with \( \text{sgn}() \) achieve classification?

With \( \text{sgn}() \) operation:

\[
f(x_i) = y_i^* = \text{sgn}(w^T x_i)
\]

\[
L = \frac{1}{N} \sum_{i=1}^{N} (w^T x_i - y_i)^2
\]

- Anything point to one side of \( y=0 \) intersection is class +1, anything on the other side of intersection is class -1
Why does linear regression with \( \text{sgn()} \) achieve classification?

With \( \text{sgn()} \) operation:

\[
f(x_i) = y_i^* = \text{sgn}(w^T x_i)
\]

\[
L = \frac{1}{N} \sum_{i=1}^{N}(w^T x_i - y_i)^2
\]

Closed-form solution available for this line of best fit \( w \) via pseudo-inverse (see last lecture’s notes)

Let’s consider some other strategies to solve for \( w \)...
Simplest gradient descent: numerically compute the gradient and take a step

$$\frac{dL(W_1)}{dW_1} = \lim_{h \to 0} \frac{L(W_1 + h) - L(W_1)}{h}$$

With a matrix, compute this for each entry:

- Repeat for all entries of $W$, $dL/dW$ will have $N\times M$ entries for $N \times M$ matrix

Example:

$$W = [1,2;3,4] \quad W_1 + h = [1.001,2;3,4] \quad dL(W_1)/dW_1 = 12.8 - 12.79/0.001$$

$$L(W, x, y) = 12.79 \quad L(W_1+h, x, y) = 12.8$$

$$dL(W_1)/dW_1 = 10$$
Some quick details about gradient descent

- For non-convex functions, local minima can obscure the search for global minima.

- Analyzing critical points (plateaus) of function of interest is important.
Some quick details about gradient descent

• For non-convex functions, local minima can obscure the search for global minima

• Analyzing critical points (plateaus) of function of interest is important

• Critical points at $df/dx = 0$

• 2\textsuperscript{nd} derivative $d^2f/dx^2$ tells us the type of critical point:
  • Minima at $d^2f/dx^2 > 0$
  • Maxima at $d^2f/dx^2 < 0$
Some quick details about gradient descent

Often we’ll have functions of m variables

\[ f : \mathbb{R}^n \rightarrow \mathbb{R} \quad \text{(e.g., } f(\mathbf{x}) = \sum (A\mathbf{x} - \mathbf{y})^2 ) \]

Will take partial derivatives \( \frac{\partial}{\partial x_i} f(\mathbf{x}) \) and put them in gradient vector \( \mathbf{g} : \nabla_{\mathbf{x}} f(\mathbf{x}) \)
Some quick details about gradient descent

Often we’ll have functions of \( m \) variables

\[
f : \mathbb{R}^n \rightarrow \mathbb{R} \quad \text{(e.g., } f(\mathbf{x}) = \sum (A\mathbf{x} - \mathbf{y})^2 \text{)}
\]

Will take partial derivatives \( \frac{\partial}{\partial x_i} f(\mathbf{x}) \) and put them in gradient vector \( \mathbf{g} = \nabla_\mathbf{x} f(\mathbf{x}) \)

Will have many second derivatives:

\[
H(f)(\mathbf{x})_{i,j} = \frac{\partial^2}{\partial x_i \partial x_j} f(\mathbf{x}) \quad \text{Hessian Matrix}
\]
Some quick details about gradient descent

Often we’ll have functions of m variables

\[ f : \mathbb{R}^n \to \mathbb{R} \quad \text{(e.g., } f(x) = \Sigma (Ax-y)^2) \]

Will take partial derivatives \( \frac{\partial}{\partial x_i} f(x) \) and put them in gradient vector \( g = \nabla_x f(x) \)

Will have many second derivatives:

\[ H(f)(x)_{i,j} = \frac{\partial^2}{\partial x_i \partial x_j} f(x) \quad \text{Hessian Matrix} \]

In general, we’ll have functions that map m variables to n variables

\[ f : \mathbb{R}^m \to \mathbb{R}^n \quad \text{(e.g., } f(x) = Wx, W \text{ is } n \times m) \]

\[ J \in \mathbb{R}^{n \times m} \text{ of } f : \quad J_{i,j} = \frac{\partial}{\partial x_j} f(x)_i \quad \text{Jacobian Matrix} \]
Quick example

\[ f(x) = x_1^2 - x_2^2 \]
Quick example

\[ f(x) = x_1^2 - x_2^2 \]

\[ g = \begin{bmatrix} 2x_1 \\ -2x_2 \end{bmatrix} \]
Quick example

\[ f(x) = x_1^2 - x_2^2 \]

\[ g = \begin{bmatrix} 2x_1 \\ -2x_2 \end{bmatrix} \quad H = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix} \]

- Convex functions have positive semi-definite Hessians (Trace >= 0)
- Trace/eigenvalues of Hessian are useful evaluate critical points & guide optimization
**Steepest descent and the best step size $\epsilon$**

1. Evaluate function $f(x^{(0)})$ at an initial guess point, $x^{(0)}$
2. Compute gradient $g^{(0)} = \nabla_x f(x^{(0)})$
3. Next point $x^{(1)} = x^{(0)} - \epsilon^{(0)} g^{(0)}$
4. Repeat – $x^{(n+1)} = x^{(n)} - \epsilon^{(n)} g^{(n)}$, until $|x^{(n+1)} - x^{(n)}| < \text{threshold } t$

```python
while previous_step_size > precision and iters < max_iters:
    prev_x = cur_x
    cur_x -= gamma * df(prev_x)
    previous_step_size = abs(cur_x - prev_x)
    iters+=1
```
Steepest descent and the best step size $\varepsilon$

1. Evaluate function $f(x^{(0)})$ at an initial guess point, $x^{(0)}$
2. Compute gradient $\mathbf{g}^{(0)} = \nabla_x f(x^{(0)})$
3. Next point $x^{(1)} = x^{(0)} - \varepsilon^{(0)} \mathbf{g}^{(0)}$
4. Repeat – $x^{(n+1)} = x^{(n)} - \varepsilon^{(n)} \mathbf{g}^{(n)}$, until $|x^{(n+1)} - x^{(n)}| < \text{threshold} \ t$

$$L = \frac{1}{N} \sum_{i=1}^{N} (w^T x_i - y_i)^2$$

$$\nabla L(w) = \frac{2}{N} X^T (Xw - y)$$

```python
while previous_step_size > precision and iters < max_iters:
    prev_x = cur_x
    cur_x -= gamma * df(prev_x)
    previous_step_size = abs(cur_x - prev_x)
    iters+=1
```
Steepest descent and the best step size $\varepsilon$

What is a good step size $\varepsilon^{(n)}$?
Steepest descent and the best step size $\epsilon$

What is a good step size $\epsilon^{(n)}$?

To find out, take 2\textsuperscript{nd} order Taylor expansion of $f$ (a good approx. for nearby points):

$$f(x) \approx f(x^{(0)}) + (x - x^{(0)})^\top g + \frac{1}{2} (x - x^{(0)})^\top H(x - x^{(0)})$$
Steepest descent and the best step size $\varepsilon$

What is a good step size $\varepsilon^{(n)}$?

To find out, take 2\textsuperscript{nd} order Taylor expansion of $f$ (a good approx. for nearby points):

$$ f(x) \approx f(x^{(0)}) + (x - x^{(0)})^\top g + \frac{1}{2} (x - x^{(0)})^\top H(x - x^{(0)}) $$

Then, evaluate at the next step:

$$ f(x^{(0)} - \varepsilon g) \approx f(x^{(0)}) - \varepsilon g^\top g + \frac{1}{2} \varepsilon^2 g^\top H g $$
Steepest descent and the best step size $\epsilon$

What is a good step size $\epsilon^{(n)}$?

To find out, take 2\(^{nd}\) order Taylor expansion of $f$ (a good approx. for nearby points):

$$f(x) \approx f(x^{(0)}) + (x - x^{(0)})^\top g + \frac{1}{2} (x - x^{(0)})^\top H(x - x^{(0)})$$

Then, evaluate at the next step:

$$f(x^{(0)} - \epsilon g) \approx f(x^{(0)}) - \epsilon g^\top g + \frac{1}{2} \epsilon^2 g^\top Hg$$

Solve for optimal step (when Hessian is positive):

$$\epsilon^* = \frac{g^\top g}{g^\top Hg}$$

J. R. Shewchuck, “An Introduction to the Conjugate Gradient Method Without the Agonizing Pain”
The linear classification model – what’s not to like?

\[ L_{in} = \| Wx - y \|^2 \]

Training error

Output \( y^* \)

Model \( y^* = Wx \)

Training error \( \frac{dL}{dW} \)

Training Data

Ex. \([x_1, y_1]\)  \[\ldots\]  Ex. \([x_K, y_K]\)
The linear classification model – what’s not to like?

1. Can only separate data with lines (hyper-planes)…

- **Solution hypothesis**
  - **Training error**
    \[ L_{\text{in}} = \| Wx - y \|^2 \]
  - **Model**
    \[ y^* = Wx \]
  - **Output**
  - **Training Data**
    \[ \text{Ex. } [x_1, y_1], \text{Ex. } [x_K, y_K] \]
  - **Training error**
    \[ \frac{dL}{dW} \]
The linear classification model – what’s not to like?

1. Can only separate data with lines (hyper-planes)…
2. We only allowed for binary labels (y = +/- 1)
The linear classification model – what’s not to like?

The training error is given by:

\[ L_{\text{lin}} = \| Wx - y \|^2 \]

1. Can only separate data with lines (hyper-planes)…

2. We only allowed for binary labels (\( y = +/- 1 \))

3. Error function \( L_{\text{lin}} \) inherently makes assumptions about statistical distribution of data
Cost functions matter: a simple example

\[ f(x, W) \]

+1 = You
-1 = Bad guy

https://www.cnet.com/how-to/apple-face-id-everything-you-need-to-know/
Cost functions matter: a simple example

Two types of error: false accept and false reject

f(x, W)

<table>
<thead>
<tr>
<th>y</th>
<th>+1</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1</td>
<td>No Error</td>
<td>False reject</td>
</tr>
<tr>
<td></td>
<td>(you/you)</td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>False accept</td>
<td>No Error</td>
</tr>
<tr>
<td></td>
<td>(bad guy/bad guy)</td>
<td>(bad guy/bad guy)</td>
</tr>
</tbody>
</table>
Cost functions matter: a simple example

Two types of error: false accept and false reject

On a standard phone, what’s a good cost function?

ReLU(x) = 0, x < 0
= x, x >= 0

It’s you, but you can’t get in...

Letting an intruder in
Cost functions matter: a simple example

Two types of error: false accept and false reject

On a standard phone, what’s a good cost function?

\[
L_{in} = \text{ReLU}[f(x, W) - y] + 10 \text{ReLU}[y - f(x, W)]
\]

- **Penalty for intruder**
- **Large penalty for annoyance...**

\[
\begin{array}{cc}
+1 & \text{No Error} \\
-1 & \text{False accept}
\end{array}
\]

\[
\begin{array}{cc}
+1 & \text{False reject} \\
-1 & \text{No Error}
\end{array}
\]

It’s you, but you can’t get in...

Letting an intruder in

https://www.cnet.com/how-to/apple-face-id-everything-you-need-to-know/
Cost functions matter: a simple example

What if you’re a CIA agent?

Letting an intruder in
Cost functions matter: a simple example

What if you’re a CIA agent?

Letting an intruder in

$$L_{\text{in}} = 100,000 \text{ReLU}[f(x, W) - y] + \text{ReLU}[y - f(x, W)]$$

BIG penalty for intruder

Don’t mind about annoyance...

| $f(x, W)$ |  
|---|---|
| +1 | No Error |
| -1 | False accept |
| +1 | False reject |
| -1 | No Error |

It’s you, but you can’t get in...

https://www.cnet.com/how-to/apple-face-id-everything-you-need-to-know/
Cost functions matter: a simple example

Establishing cost function tied to conditional probabilities:

\[ P(y = -1 \mid f(x, W) = +1) \]

Establish \( L, W \) to balance and minimize these probabilities

\[ P(y = +1 \mid f(x, W) = -1) \]

The following table describes the outcomes:

<table>
<thead>
<tr>
<th>( y )</th>
<th>No Error</th>
<th>False reject</th>
<th>False accept</th>
<th>No Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

It’s you, but you can’t get in...

Letting an intruder in

https://www.cnet.com/how-to/apple-face-id-everything-you-need-to-know/
Machine learning and probability

• Probability measures help determine when to use certain cost functions

• In previous case, we can measure probability of seeing a particular label, given model & data:

\[ p(y|w, x) \]
Machine learning and probability

- Probability measures help determine when to use certain cost functions

- In previous case, we can measure probability of seeing a particular label, given model & data:

\[ p(y|w, x) \]

- In general, to find the best model, we’d like to infer \( w \), having at hand our labeled data:

Maximum Likelihood Estimation \( p(w|x_1, \ldots x_N; y_1, \ldots y_N) \)
Machine learning and probability

• Probability measures help determine when to use certain cost functions

• In previous case, we can measure probability of seeing a particular label, given model & data:

\[ p(y|w, x) \]

• In general, to find the best model, we’d like to infer \( w \), having at hand our labeled data:

Maximum Likelihood Estimation

\[ p(w|x_1, \ldots x_N; y_1, \ldots y_N) \]

• These two quantities are connected via Bayes’ Theorem

\[
p(w|x) = \frac{p(x|w)p(w)}{p(x)} \quad \text{With 2 conditioned variables:} \quad p(w|x, y) = \frac{p(x, y|w)p(w)}{p(x, y)}
\]

\[ p(w|x, y) \propto p(x, y|w) \propto p(y|x, w) \]
Machine learning and probability

- Probability measures help determine when to use certain cost functions
- In previous case, we can measure probability of seeing a particular label, given model & data:

\[ p(y \mid w, x) \]

- In general, to find the best model, we’d like to infer \( w \), having at hand our labeled data:

Maximum Likelihood Estimation

\[ p(w \mid x_1, \ldots, x_N; y_1, \ldots, y_N) \]

- These two quantities are connected via Bayes’ Theorem

\[
\begin{align*}
p(w \mid x) &= \frac{p(x \mid w)p(w)}{p(x)} \\
p(w \mid x, y) &= \frac{p(x, y \mid w)p(w)}{p(x, y)}
\end{align*}
\]

What you want: but hard to vary data to find model…

What you can do: test the model, check the result
Linear classification is the maximum likelihood for Gaussian data

- Given a close relationship between $p(w|\mathbf{x}, y) \leftrightarrow p(y|x, w)$:

  Maximum likelihood estimation asks the question,

  For what $w$ is $p(y_1, ... y_N|x_1, ... x_N; w)$ maximized?
Linear classification is the maximum likelihood for Gaussian data

- Given a close relationship between $p(\mathbf{w}|\mathbf{x}, \mathbf{y}) \leftrightarrow p(\mathbf{y}|\mathbf{x}, \mathbf{w})$:

  Maximum likelihood estimation asks the question,

  For what $\mathbf{w}$ is $p(\mathbf{y}_1, ... \mathbf{y}_N|\mathbf{x}_1, ... \mathbf{x}_N; \mathbf{w})$ maximized?

  For what $\mathbf{w}$ is $\prod_{i=1}^{N} p(\mathbf{y}_i, |\mathbf{x}_i, \mathbf{w})$ maximized?
Linear classification is the maximum likelihood for Gaussian data

- Given a close relationship between $p(\mathbf{w}|\mathbf{x}, \mathbf{y}) \leftrightarrow p(\mathbf{y}|\mathbf{x}, \mathbf{w})$:

  Maximum likelihood estimation asks the question,

  For what $\mathbf{w}$ is $p(\mathbf{y}_1, ...\mathbf{y}_N|\mathbf{x}_1, ...\mathbf{x}_N; \mathbf{w})$ maximized?

  For what $\mathbf{w}$ is $\prod_{i=1}^{N} p(\mathbf{y}_i, |\mathbf{x}_i, \mathbf{w})$ maximized?

- Let’s assume our labels are a “noisy” Gaussian process that surround the correct label:

  $$y_i = \mathbf{w}^T \mathbf{x}_i + n$$  \hspace{1cm} (n is zero-mean Gaussian noise)

- Then, the above cond. prob. for labels can be expressed as a multivariate Gaussian
Linear classification is the maximum likelihood for Gaussian data.

For what $w$ is $\prod_{i=1}^{N} p(y_i, |x_i, w)$ maximized?

For what $w$ is $\prod_{i=1}^{N} \exp\left(\frac{-(y_i-w^T x)^2}{2\sigma^2}\right)$ maximized?

For what $w$ is $\sum_{i=1}^{N} \frac{-(y_i-w^T x)^2}{2\sigma^2}$ maximized?

For what $w$ is $\sum_{i=1}^{N} (y_i - w^T x)^2$ minimized?

Summary: Linear classification assumes labels are a Gaussian random process, and then thresholds them to either -1 or +1.
The linear classification model – what’s not to like?

1. Can only separate data with lines (hyper-planes)…

2. We only allowed for binary labels ($y = +/- 1$)

3. Error function $L_{in}$ inherently makes assumptions about statistical distribution of data
Let’s think about the labels as a probabilistic measure:

Linear regression: predict some output $h(x)$ from $x$
Let’s think about the labels as a probabilistic measure:

Linear regression: predict some output \( h(x) \) from \( x \)

\[
h(x) = x \in (-\infty, \infty)
\]

Not a probabilistic mapping
Let's think about the labels as a probabilistic measure:

Linear regression: predict some output $h(x)$ from $x$

$$h(x) = x \in (-\infty, \infty)$$

Linear classifier: predict binary output $h(x)$ from $x$

$$h(x) = \text{sign}(w^T x)$$

$$h(x) = \text{sign}(x) \in \{0, 1\}$$

Not a probabilistic mapping

Probabilistic, but all-or-nothing
Let’s think about the labels as a probabilistic measure:

Linear regression:

$$h(x) = x \in (-\infty, \infty)$$

Linear classifier:

$$h(x) = \text{sign}(w^T x)$$

$$h(x) = \text{sign}(x) \in \{0, 1\}$$

Logistic classifier:

$$h(x) = \theta(x) \in [0, 1]$$
Probabilistic interpretation of function that maps outputs to labels, \( h(x) = \theta(x) \)

**Example:** You are trying to predict the probability that a patient may have a certain form of a disease, \( \theta(x) \), given a number of observations and measurements, \( x \)

**Example:** You are trying to predict the probability of rain tomorrow, \( \theta(x) \), given a set of satellite image data, \( x \)
Probabilistic interpretation of function that maps outputs to labels, \( h(x) = \theta(x) \)

**Example**: You are trying to predict the probability that a patient may have a certain form a disease, \( \theta(x) \), given a number of observations and measurements, \( x \)

**Example**: You are trying to predict the probability of rain tomorrow, \( \theta(x) \), given a set of satellite image data, \( x \)

**The Logistic Function \( \theta \)**

\[
\theta(x) = \frac{e^x}{1+e^x}
\]

Also called Sigmoid function

- Use soft threshold to map any number to \([0,1]\) range
- Sigmoid “flattens out” \( x \)
From linear classification to logistic classification

• Let’s re-derive a cost function for the case where labels are treated as probabilities
  • You’ll use derivatives of this more often than not in Tensorflow
• During learning, we will again have two classes (in this simple example), y = +/- 1
• map these binary values onto a [0,1] probability distribution

Formula for likelihood using the logistic function $\theta$, given binary labels

\[
P(y \mid x) = \begin{cases} 
\theta(x) & \text{For } y = +1 \\
1 - \theta(x) & \text{For } y = -1
\end{cases}
\]
From linear classification to logistic classification

- Let’s re-derive a cost function for the case where labels are treated as probabilities
  - You’ll use derivatives of this more often than not in Tensorflow
- During learning, we will again have two classes (in this simple example), $y = +/- 1$
- Map these binary values onto a [0,1] probability distribution

Formula for likelihood using the logistic function $\theta$, given binary labels

$$P(y \mid f(x)) = \begin{cases} 
\theta(f(x)) & \text{For } y = +1 \\
1 - \theta(f(x)) & \text{For } y = -1 
\end{cases}$$

Rough Interpretation:
- If network output $f(x)$ is large, then should map to $y=+1$ with high probability
- $\theta(f(x))$ is large for large value of $x$
- If network output $f(x)$ is small, then should map to $y=-1$ with high probability
- $\theta(f(x)) \sim 0$ for small values of $f(x)$, so $1 - \theta(f(x)) \sim 1$ is high probability to $y=-1$ mapping
Deriving cost function for logistic classification for probabilistic outputs

Instead of mapping \( f(\mathbf{x}) \) to either +1 or -1 with the sign operator, let’s use \( \theta \) to map it to lie between 0 and 1:

\[
P(y | \mathbf{x}) = \begin{cases} 
\theta(f(\mathbf{x})) & \text{For } y = +1 \\
1 - \theta(f(\mathbf{x})) & \text{For } y = -1 
\end{cases}
\]

We’ll stick with the case of linear classification, where \( f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} \)

Also, please note that for the logistic function, \( \theta(-a) = 1 - \theta(a) \). So, we can summarize the case-based definition above with a single function,

\[
P(y | \mathbf{x}) = \theta(y \mathbf{w}^T \mathbf{x})
\]

Where \( y = +/-1 \) flips \( \theta(\mathbf{w}^T \mathbf{x}) \) to be either \( \theta(\mathbf{w}^T \mathbf{x}) \) or \( \theta(-\mathbf{w}^T \mathbf{x}) = 1 - \theta(\mathbf{w}^T \mathbf{x}) \)
Deriving cost function for logistic classification for probabilistic outputs

Similar to the linear classification case, the likelihood of observing $N$ independent outputs is given by,

$$P(y_1, y_2, \ldots, y_N \mid x_1, x_2, \ldots, x_N) = \prod_{n=1}^{N} P(y_n \mid x_n)$$

$$= \prod_{n=1}^{N} \theta(y_n w^T x_n)$$

This is the probability of the labels, given the data. We’d like to maximize this probability!

*Like the linear regression case, but now the probability of classes given the data is not Gaussian distributed, but instead follows the sigmoid curve (is bound to $[0,1]$, which is more realistic)

Maximize $P(y_1, y_2, \ldots, y_N \mid x_1, x_2, \ldots, x_N) = \prod_{n=1}^{N} \theta(y_n w^T x_n)$
Deriving cost function for logistic classification for probabilistic outputs

Maximize \( P(y_1, y_2, \ldots, y_N \mid x_1, x_2, \ldots, x_N) = \prod_{n=1}^{N} \theta(y_n w^T x_n) \)

Minimize \( -\frac{1}{N} \ln \left( \prod_{n=1}^{N} \theta(y_n w^T x) \right) \)
Deriving cost function for logistic classification for probabilistic outputs

Maximize \( P(y_1, y_2, \ldots, y_N | x_1, x_2, \ldots, x_N) = \prod_{n=1}^{N} \theta(y_n w^T x_n) \)

Minimize \(- \frac{1}{N} \ln \left( \prod_{n=1}^{N} \theta(y_n w^T x) \right)\)

Minimize \( \frac{1}{N} \sum_{n=1}^{N} \ln \left( \frac{1}{\theta(y_n w^T x)} \right) \)

Use relationship \( \theta(a) = \frac{1}{1 + e^{-a}} \)
Deriving cost function for logistic classification for probabilistic outputs

Maximize \( P(y_1, y_2, \ldots, y_N \mid x_1, x_2, \ldots, x_N) = \prod_{n=1}^{N} \theta(y_n w^T x_n) \)

Minimize \(-\frac{1}{N} \ln \left( \prod_{n=1}^{N} \theta(y_n w^T x) \right)\)

Minimize \( \frac{1}{N} \sum_{n=1}^{N} \ln \left( \frac{1}{\theta(y_n w^T x)} \right) \)

Use relationship \( \theta(a) = \frac{1}{1 + e^{-a}} \)

Minimize \( L_{in}(w) = \frac{1}{N} \sum_{n=1}^{N} \ln(1 + e^{-y_n w^T x}) \)

Cross entropy error for logistic classification
Requires iterative solution to minimize
Deriving cost function for logistic classification for probabilistic outputs

Maximize  \( \mathbb{P}(y_1, y_2, \ldots, y_N \mid x_1, x_2, \ldots, x_N) = \prod_{n=1}^{N} \theta(y_n \mathbf{w}^T \mathbf{x}_n) \)

Minimize  \( -\frac{1}{N} \ln \left( \prod_{n=1}^{N} \theta(y_n \mathbf{w}^T \mathbf{x}) \right) \)

Minimize  \( \frac{1}{N} \sum_{n=1}^{N} \ln \left( \frac{1}{\theta(y_n \mathbf{w}^T \mathbf{x})} \right) \)

Use relationship  \( \theta(a) = \frac{1}{1 + e^{-a}} \)

Minimize  \( L_{in}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \ln(1 + e^{-y_n \mathbf{w}^T \mathbf{x}}) \)

Cross entropy error for logistic classification

Requires iterative solution to minimize

Mean-square error for linear classification

Closed form solution available

\( L_{in}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} (y_n - \mathbf{w}^T \mathbf{x})^2 \)
The linear classification model – what’s not to like?

Training error

\[ L_{\text{lin}}(y, f(W, x)) = \text{cross_entropy}(y, f(W, x)) \]

Output \( y^* \)

Training Data

\[ y^* = Wx \]

Probabilistic mapping to \( y \)

\[ L(x, f(W, x)) = \text{cross_entropy}(y, f(W, x)) \]
The linear classification model – what’s not to like?

1. Can only separate data with lines (hyper-planes)…
2. We only allowed for binary labels ($y = +/\text{-} 1$)
3. Error function $L_{\text{in}}$ inherently makes assumptions about statistical distribution of data
Learned $f$: not flexible

$$f = W_1 x$$
Can we add flexibility by multiplying with another weight matrix?

\[
\begin{align*}
f_1 &= W_1 x + b_1 \\
f_2 &= W_2 f_1 + b_2 \\
f_2 &= W_2(W_1 x + b_1) + b_2 \\
\end{align*}
\]

\[f_2 = W' x + b'\] Unfortunately not...
Deep imaging

Training data

\[ f = W_1 x \]

Learned \( f \): not flexible

Training data

\[ f = W_2 \max(W_1 x, 0) \]

Learned \( f \): a bit flexible

\[ f = W_2 \cdot \text{NL} \cdot W_1 \]
Machine Learning and Imaging
– Roarke Horstmeyer (2020)

$\mathbf{f} = W_1 \mathbf{x}$

Learned $f$: not flexible

$\mathbf{f} = W_2 \text{max}(W_1 \mathbf{x}, 0)$

Learned $f$: a bit flexible

$\mathbf{f} = W_3 \text{max}(0, W_2 \text{max}(W_1 \mathbf{x}, 0))$

Learned $f$: more flexible

Does it generalize???

We can keep adding these “layers”…
Getting us to Convolutional Neural Networks

After, apply non-linearity and sub-sampling

Each matrix $W$ is a convolution matrix

Repeat a few times

At the end, use a full $W$ for a final matrix multiplication

Original Image published in [LeCun et al., 1998]
Getting us to Convolutional Neural Networks

In practice, this process is repeated many times:

Original Image published in [LeCun et al., 1998]