Inductive bias and regularization
Inductive Bias in Linear Regression
Motivation

- How to cope with non-uniqueness of solution to normal equations?
- How is prior knowledge incorporated into linear models?
Example: Fitting affine functions

Example: Fit affine function \( f(x) = mx + \theta \) to single data point \((\frac{1}{2}, 3) \in \mathbb{R} \times \mathbb{R}\)
Example: Fitting affine functions

Example: Fit affine function $f(x) = mx + \theta$ to single data point $(\frac{1}{2}, 3) \in \mathbb{R} \times \mathbb{R}$

Normal equations:

$$
\overline{x^2} \cdot m + \overline{x} \cdot \theta = \overline{xy}
$$

$$
\overline{x} \cdot m + \theta = \overline{y}
$$
Example: Fitting affine functions

Example: Fit affine function $f(x) = mx + \theta$ to single data point $(\frac{1}{2}, 3) \in \mathbb{R} \times \mathbb{R}$

Normal equations:

\[
\begin{align*}
\frac{1}{4} \cdot m &+ \frac{1}{2} \cdot \theta = \frac{3}{2} \\
\frac{1}{2} \cdot m &+ \theta = 3
\end{align*}
\]
Example: Fitting affine functions

Example: Fit affine function $f(x) = mx + \theta$ to single data point $(\frac{1}{2}, 3) \in \mathbb{R} \times \mathbb{R}$

Normal equations:

\[
\begin{align*}
\frac{1}{4} \cdot m + \frac{1}{2} \cdot \theta &= \frac{3}{2} \\
\frac{1}{2} \cdot m + \theta &= 3
\end{align*}
\]
OLS is not fully-specified

**OLS**: Given labeled examples $S$ from $\mathbb{R}^d \times \mathbb{R}$, return solution $\vec{w}$ to normal equations

$$\sum_{(\vec{x}, y) \in S} x_1 (\vec{x} \cdot \vec{w} - y) = 0$$

$$\vdots$$

$$\sum_{(\vec{x}, y) \in S} x_d (\vec{x} \cdot \vec{w} - y) = 0$$

Oops! Not a fully-specified algorithm:

**Doesn’t say which solution to pick if there are multiple solutions**
One possible approach:

- Choose the solution \( \vec{w} \in \mathbb{R}^d \) of minimum (squared) Euclidean norm \( \| \vec{w} \|_2^2 \)
Minimum Euclidean norm solution

One possible approach:

- Choose the solution $\vec{w} \in \mathbb{R}^d$ of minimum (squared) Euclidean norm $\|\vec{w}\|^2_2$
- Regard $\|\vec{w}\|_2$ as a measure of “steepness” of the linear function $f(\vec{x}) = \vec{x} \cdot \vec{w}$:

$$|f(\vec{x} + \vec{\Delta}) - f(\vec{x})| = |\vec{\Delta} \cdot \vec{w}| \leq \|\vec{\Delta}\|_2 \|\vec{w}\|_2$$

(Cauchy-Schwarz inequality)
Minimum Euclidean norm solution

One possible approach:

Choose the solution $\vec{w} \in \mathbb{R}^d$ of minimum (squared) Euclidean norm $\|\vec{w}\|^2_2$

Regard $\|\vec{w}\|_2$ as a measure of “steepness” of the linear function $f(\vec{x}) = \vec{x} \cdot \vec{w}$:

$$|f(\vec{x} + \vec{\Delta}) - f(\vec{x})| = |\vec{\Delta} \cdot \vec{w}| \leq \|\vec{\Delta}\|_2 \|\vec{w}\|_2$$

(Cauchy-Schwarz inequality)

If $\vec{\Delta} = c \vec{x}$ for some $c \in \mathbb{R}$, then inequality above holds as an equality.
Minimum Euclidean norm solution

One possible approach:

▶ Choose the solution \( \mathbf{w} \in \mathbb{R}^d \) of minimum (squared) Euclidean norm \( \| \mathbf{w} \|^2 \):

▶ Regard \( \| \mathbf{w} \|^2 \) as a measure of “steepness” of the linear function \( f(\mathbf{x}) = \mathbf{x} \cdot \mathbf{w} \):

\[
|f(\mathbf{x} + \mathbf{\Delta}) - f(\mathbf{x})| = |\mathbf{\Delta} \cdot \mathbf{w}| \leq |\mathbf{\Delta}|_2 \| \mathbf{w} \|_2
\]

(Cauchy-Schwarz inequality)

▶ If \( \mathbf{\Delta} = c \mathbf{x} \) for some \( c \in \mathbb{R} \), then inequality above holds as an equality

Note: This is definitely not the only option!
Properties of minimum Euclidean norm solution

**Theorem.** Solution \( \vec{w} \) to normal equations \((A^T A)\vec{w} = A^T \vec{b}\) of minimum Euclidean norm is

1. unique \hspace{1cm} (i.e., no two distinct solutions can both have same minimum norm)
2. contained in the row space of \( A \) \hspace{1cm} (i.e., span of feature vectors in training data)
Properties of minimum Euclidean norm solution

**Theorem.** Solution $\vec{w}$ to normal equations $(A^T A)\vec{w} = A^T \vec{b}$ of minimum Euclidean norm is

1. unique (i.e., no two distinct solutions can both have same minimum norm)
2. contained in the row space of $A$ (i.e., span of feature vectors in training data)

▶ If $\vec{w}$ and $\vec{w}'$ are both solutions, then so is $\frac{1}{2}(\vec{w} + \vec{w}')$, and

$$\| \frac{1}{2}(\vec{w} + \vec{w}') \|_2^2 = \frac{1}{2} \left( \| \vec{w} \|_2^2 + \| \vec{w}' \|_2^2 \right) - \frac{1}{4} \| \vec{w} - \vec{w}' \|_2^2$$
**Theorem.** Solution $\vec{w}$ to normal equations $(A^T A)\vec{w} = A^T \vec{b}$ of minimum Euclidean norm is

1. unique (i.e., no two distinct solutions can both have same minimum norm)
2. contained in the row space of $A$ (i.e., span of feature vectors in training data)

- If $\vec{w}$ and $\vec{w}'$ are both solutions, then so is $\frac{1}{2}(\vec{w} + \vec{w}')$, and

$$\| \frac{1}{2}(\vec{w} + \vec{w}') \|^2_2 = \frac{1}{2} \left( \| \vec{w} \|^2_2 + \| \vec{w}' \|^2_2 \right) - \frac{1}{4} \| \vec{w} - \vec{w}' \|^2_2$$

- If $\vec{w} = \vec{u} + \vec{v}$ is a solution, where $\vec{u}$ is in row space of $A$ and $\vec{v}$ is in null space of $A$, then

$$\| \vec{u} \|^2_2 = \| \vec{w} \|^2_2 - \| \vec{v} \|^2_2$$

and $\vec{u}$ is also a solution:

$$(A^T A)\vec{u} = (A^T A)\vec{w} - (A^T A)\vec{v} = (A^T A)\vec{w} = A^T \vec{b}$$
Example: Different solutions to normal equations

Example: Linear regression with trigonometric feature expansion
\[ \tilde{\varphi}(x) := (\sin(x), \cos(x), \sin(2x), \cos(2x), \ldots, \sin(32x), \cos(32x)) \in \mathbb{R}^{64} \]
Example: Linear regression with trigonometric feature expansion

$$\tilde{\varphi}(x) := (\sin(x), \cos(x), \sin(2x), \cos(2x), \ldots, \sin(32x), \cos(32x)) \in \mathbb{R}^{64}$$

- $d = 64$ features
- $n = 32$ training examples
- Infinitely-many solutions to normal equations
Example: Linear regression with trigonometric feature expansion

\[ \vec{\phi}(x) := (\sin(x), \cos(x), \sin(2x), \cos(2x), \ldots, \sin(32x), \cos(32x)) \in \mathbb{R}^{64} \]

- \( d = 64 \) features
- \( n = 32 \) training examples
- Infinitely-many solutions to normal equations
Example: Different solutions to normal equations

Example: Linear regression with trigonometric feature expansion

\[ \phi(x) := (\sin(x), \cos(x), \sin(2x), \cos(2x), \ldots, \sin(32x), \cos(32x)) \in \mathbb{R}^{64} \]

- \( d = 64 \) features
- \( n = 32 \) training examples
- Infinitely-many solutions to normal equations
- (Squared) Euclidean norm:
  \[ \sum_{i=1}^{64} w_i^2 \]
Example: Different solutions to normal equations

Example: Linear regression with trigonometric feature expansion

$$\vec{\phi}(x) := (\sin(x), \cos(x), \sin(2x), \cos(2x), \ldots, \sin(32x), \cos(32x)) \in \mathbb{R}^{64}$$

- $d = 64$ features
- $n = 32$ training examples
- Infinitely-many solutions to normal equations
- (Squared) Euclidean norm:
  $$\sum_{i=1}^{64} w_i^2$$
- (Squared) "weighted" norm:
  $$\sum_{i=1}^{64} 2^{\lceil i/2 \rceil} w_i^2$$
Example: Fitting affine functions, revisited

Example: Fit affine function $f(x) = mx + \theta$ to single data point $(0.5, 3) \in \mathbb{R} \times \mathbb{R}$

(Sq.) Euclidean norm of $(m, \theta)$:

$$1.2^2 + 2.4^2 = 7.2$$
$$0^2 + 3^2 = 9$$
$$4^2 + 1^2 = 17$$
Example: Fitting affine functions, revisited

Example: Fit affine function \( f(x) = mx + \theta \) to single data point \((0.5, 3) \in \mathbb{R} \times \mathbb{R}\)

(Sq.) Euclidean norm of \((m, \theta)\):

\[
1.2^2 + 2.4^2 = 7.2 \\
0^2 + 3^2 = 9 \\
4^2 + 1^2 = 17
\]

(Sq.) Euclidean norm of \(m\):

\[
1.2^2 = 1.44 \\
0^2 = 0 \\
4^2 = 16
\]
In OLS examples, “quality of fit” (to training data) is same for all solutions to normal equations.

Preference to pick one solution over another is “external” to the training data. Such a preference is called an inductive bias (akin to notions of “simplicity” à la Occam’s Razor).

Preferences also come from:

- Modeling assumptions (e.g., IID model, normal linear regression model)
- Feature engineering, feature expansion
- Choice of predictor type (e.g., decision tree, linear model)

All ML algorithms encode some form of inductive bias.

Upshot: No such thing as “letting data speak for itself” in ML.
Inductive bias

- In OLS examples, “quality of fit” (to training data) is same for all solutions to normal equations
- Preference to pick one solution over another is “external” to the training data
Inductive bias

- In OLS examples, “quality of fit” (to training data) is same for all solutions to normal equations.
- Preference to pick one solution over another is “external” to the training data.
- Such a preference is called an **inductive bias** (akin to notions of “simplicity” à la Occam’s Razor).
Inductive bias

- In OLS examples, “quality of fit” (to training data) is same for all solutions to normal equations
- Preference to pick one solution over another is “external” to the training data
- Such a preference is called an **inductive bias**
  (Akin to notions of “simplicity” à la Occam’s Razor)

“Preferences” also come from:

- Modeling assumptions (e.g., IID model, normal linear regression model)
- Feature engineering, feature expansion
- Choice of predictor type (e.g., decision tree, linear model)
- . . .
Inductive bias

- In OLS examples, “quality of fit” (to training data) is same for all solutions to normal equations.
- Preference to pick one solution over another is “external” to the training data.
- Such a preference is called an inductive bias (Akin to notions of “simplicity” à la Occam’s Razor).

“Preferences” also come from:

- Modeling assumptions (e.g., IID model, normal linear regression model)
- Feature engineering, feature expansion
- Choice of predictor type (e.g., decision tree, linear model)
- . . .

All ML algorithms encode some form of inductive bias.
In OLS examples, “quality of fit” (to training data) is same for all solutions to normal equations

Preference to pick one solution over another is “external” to the training data

Such a preference is called an **inductive bias**
(Akin to notions of “simplicity” à la Occam’s Razor)

“Preferences” also come from:

- Modeling assumptions (e.g., IID model, normal linear regression model)
- Feature engineering, feature expansion
- Choice of predictor type (e.g., decision tree, linear model)
- . . .

All ML algorithms encode some form of inductive bias

**Upshot:** No such thing as “letting data speak for itself” in ML
Regularization in Linear Regression
Consider the two concerns:

- SSE on training data: \( \| A\vec{w} - \vec{b} \|_2^2 \)
- (Squared) Euclidean norm of weight vector: \( \| \vec{w} \|_2^2 \)
Consider the two concerns:

- SSE on training data: \( \| A\vec{w} - \vec{b} \|_2^2 \)
- (Squared) Euclidean norm of weight vector: \( \| \vec{w} \|_2^2 \)

**Ridge regression**: For \( \lambda > 0 \), minimize the **regularized objective function**

\[
\| A\vec{w} - \vec{b} \|_2^2 + \lambda \| \vec{w} \|_2^2
\]
Consider the two concerns:

- SSE on training data: \( \| A\vec{w} - \vec{b} \|^2_2 \)
- (Squared) Euclidean norm of weight vector: \( \| \vec{w} \|^2_2 \)

**Ridge regression:** For \( \lambda > 0 \), minimize the **regularized objective function**

\[
\| A\vec{w} - \vec{b} \|^2_2 + \lambda \| \vec{w} \|^2_2
\]

- If \( \lambda = 0 \), then equivalent to OLS
Regularized objectives

Consider the two concerns:

▶ SSE on training data: $\|A\vec{w} - \vec{b}\|_2^2$

▶ (Squared) Euclidean norm of weight vector: $\|\vec{w}\|_2^2$

**Ridge regression:** For $\lambda > 0$, minimize the regularized objective function

$$\|A\vec{w} - \vec{b}\|_2^2 + \lambda \|\vec{w}\|_2^2$$

▶ If $\lambda = 0$, then equivalent to OLS

▶ If $\lambda > 0$, then minimizer is always unique!
Regularized objectives

Consider the two concerns:

▶ SSE on training data: \( \| A\vec{w} - \vec{b} \|_2^2 \)
▶ (Squared) Euclidean norm of weight vector: \( \| \vec{w} \|_2^2 \)

**Ridge regression:** For \( \lambda > 0 \), minimize the regularized objective function

\[
\| A\vec{w} - \vec{b} \|_2^2 + \lambda \| \vec{w} \|_2^2
\]

▶ If \( \lambda = 0 \), then equivalent to OLS
▶ If \( \lambda > 0 \), then minimizer is always unique!
▶ Term \( \lambda \| \vec{w} \|_2^2 \) is called **ridge** (a.k.a. **Tikhonov, \( \ell_2 \)**) regularization or penalty
Regularized objectives

Consider the two concerns:

- SSE on training data: $\|A\vec{w} - \vec{b}\|_2^2$
- (Squared) Euclidean norm of weight vector: $\|\vec{w}\|_2^2$

**Ridge regression:** For $\lambda > 0$, minimize the regularized objective function

$$\|A\vec{w} - \vec{b}\|_2^2 + \lambda \|\vec{w}\|_2^2$$

- If $\lambda = 0$, then equivalent to OLS
- If $\lambda > 0$, then minimizer is always unique!
- Term $\lambda \|\vec{w}\|_2^2$ is called ridge (a.k.a. Tikhonov, $\ell_2$) regularization or penalty
- $\lambda$ is a hyperparameter that controls how much we pay attention to the second concern (compared to the first one)
Regularized objectives

Consider the two concerns:

- SSE on training data: \( \|A\vec{w} - \vec{b}\|_2^2 \)
- (Squared) Euclidean norm of weight vector: \( \|\vec{w}\|_2^2 \)

**Ridge regression:** For \( \lambda > 0 \), minimize the regularized objective function

\[
\|A\vec{w} - \vec{b}\|_2^2 + \lambda \|\vec{w}\|_2^2
\]

- If \( \lambda = 0 \), then equivalent to OLS
- If \( \lambda > 0 \), then minimizer is always unique!
- Term \( \lambda \|\vec{w}\|_2^2 \) is called ridge (a.k.a. Tikhonov, \( \ell_2 \)) regularization or penalty
- \( \lambda \) is a hyperparameter that controls how much we pay attention to the second concern (compared to the first one)
- Also called regularized MLE (since OLS = MLE in normal linear regression model)
Example: Ridge regression

Example: Linear regression with “weighted” trigonometric feature expansion

\[ \varphi(x) := \left( \frac{\sin(x)}{2^1}, \frac{\cos(x)}{2^1}, \frac{\sin(2x)}{2^2}, \frac{\cos(2x)}{2^2}, \ldots, \frac{\sin(32x)}{2^{32}}, \frac{\cos(32x)}{2^{32}} \right) \in \mathbb{R}^{64} \]

- \( d = 64 \) features
- \( n = 32 \) training examples
- Infinitely-many solutions to normal equations
- Find \( \vec{w} \) that minimizes

\[ \|A\vec{w} - \vec{b}\|_2^2 + \lambda \|\vec{w}\|_2^2 \]
Example: Ridge regression

Example: Linear regression with “weighted” trigonometric feature expansion

\[ \varphi(x) := \left( \frac{\sin(x)}{2^1}, \frac{\cos(x)}{2^1}, \frac{\sin(2x)}{2^2}, \frac{\cos(2x)}{2^2}, \ldots, \frac{\sin(32x)}{2^{32}}, \frac{\cos(32x)}{2^{32}} \right) \in \mathbb{R}^{64} \]

- \( d = 64 \) features
- \( n = 32 \) training examples
- Infinitely-many solutions to normal equations
- Find \( \vec{w} \) that minimizes

\[ \|A\vec{w} - \vec{b}\|_2^2 + \lambda \|\vec{w}\|_2^2 \]
Example: Ridge regression

Example: Linear regression with “weighted” trigonometric feature expansion

\[ \varphi(x) := \left( \frac{\sin(x)}{2^1}, \frac{\cos(x)}{2^1}, \frac{\sin(2x)}{2^2}, \frac{\cos(2x)}{2^2}, \ldots, \frac{\sin(32x)}{2^{32}}, \frac{\cos(32x)}{2^{32}} \right) \in \mathbb{R}^{64} \]

- \( d = 64 \) features
- \( n = 32 \) training examples
- Infinitely-many solutions to normal equations

Find \( \vec{w} \) that minimizes

\[ \| A\vec{w} - \vec{b} \|_2^2 + \lambda \| \vec{w} \|_2^2 \]
Example: Ridge regression

Example: Linear regression with “weighted” trigonometric feature expansion

\[ \phi(x) := \left( \frac{\sin(x)}{2^1}, \frac{\cos(x)}{2^1}, \frac{\sin(2x)}{2^2}, \frac{\cos(2x)}{2^2}, \ldots, \frac{\sin(32x)}{2^{32}}, \frac{\cos(32x)}{2^{32}} \right) \in \mathbb{R}^{64} \]

- \( d = 64 \) features
- \( n = 32 \) training examples
- Infinitely-many solutions to normal equations
- Find \( \vec{w} \) that minimizes

\[ \| A\vec{w} - \vec{b} \|^2_2 + \lambda \| \vec{w} \|^2_2 \]
Ridge regression as training data augmentation

An interpretation of ridge regularization: Augment training data with \(d\) extra, made-up examples

\[
\begin{align*}
(\tilde{x}_{n+1}, y_{n+1}) &= \left( (\sqrt{\lambda}, 0, \ldots, 0), 0 \right) \\
(\tilde{x}_{n+2}, y_{n+2}) &= \left( (0, \sqrt{\lambda}, \ldots, 0), 0 \right) \\
& \quad \vdots \\
(\tilde{x}_{n+d}, y_{n+d}) &= \left( (0, 0, \ldots, \sqrt{\lambda}), 0 \right)
\end{align*}
\]

Contribution to SSE from these extra, made-up examples is:

\[
\left( (\sqrt{\lambda}, 0, \ldots, 0) \right) \cdot \vec{w} - 0^2 + \left( (0, \sqrt{\lambda}, \ldots, 0) \right) \cdot \vec{w} - 0^2 + \cdots + \left( (0, 0, \ldots, \sqrt{\lambda}) \right) \cdot \vec{w} - 0^2
\]

which is equivalent to

\[
\left( \sqrt{\lambda} w_1 \right)^2 + \left( \sqrt{\lambda} w_2 \right)^2 + \cdots + \left( \sqrt{\lambda} w_d \right)^2 = \lambda \| \vec{w} \|^2
\]

Ridge regression objective is same as OLS objective for training data augmented with these \(d\) examples.
Ridge regression as training data augmentation

**An interpretation of ridge regularization:** Augment training data with $d$ extra, made-up examples

\[
\begin{align*}
(\tilde{x}_{n+1}, y_{n+1}) &= (\sqrt{\lambda}, 0, \ldots, 0, 0) \\
(\tilde{x}_{n+2}, y_{n+2}) &= (0, \sqrt{\lambda}, \ldots, 0, 0) \\
& \quad \vdots \\
(\tilde{x}_{n+d}, y_{n+d}) &= (0, 0, \ldots, \sqrt{\lambda}, 0)
\end{align*}
\]

Contribution to SSE from these extra, made-up examples is:

\[
\left( (\sqrt{\lambda}, 0, \ldots, 0) \cdot \vec{w} - 0 \right)^2 + \left( (0, \sqrt{\lambda}, \ldots, 0) \cdot \vec{w} - 0 \right)^2 + \cdots + \left( (0, 0, \ldots, \sqrt{\lambda}) \cdot \vec{w} - 0 \right)^2
\]

Ridge regression objective is same as OLS objective for training data augmented with these $d$ examples.
Ridge regression as training data augmentation

An interpretation of ridge regularization: Augment training data with \( d \) extra, made-up examples

\[
\begin{align*}
(\vec{x}_{n+1}, y_{n+1}) &= (\sqrt{\lambda}, 0, \ldots, 0), 0 \\
(\vec{x}_{n+2}, y_{n+2}) &= (0, \sqrt{\lambda}, \ldots, 0), 0 \\
&\vdots \\
(\vec{x}_{n+d}, y_{n+d}) &= (0, 0, \ldots, \sqrt{\lambda}), 0
\end{align*}
\]

Contribution to SSE from these extra, made-up examples is:

\[
\left( (\sqrt{\lambda}, 0, \ldots, 0) \cdot \vec{w} - 0 \right)^2 + \left( (0, \sqrt{\lambda}, \ldots, 0) \cdot \vec{w} - 0 \right)^2 + \cdots + \left( (0, 0, \ldots, \sqrt{\lambda}) \cdot \vec{w} - 0 \right)^2
\]

which is equivalent to

\[
(\sqrt{\lambda}w_1)^2 + (\sqrt{\lambda}w_2)^2 + \cdots + (\sqrt{\lambda}w_d)^2 = \lambda \|\vec{w}\|^2
\]
Ridge regression as training data augmentation

An interpretation of ridge regularization: Augment training data with \( d \) extra, made-up examples

\[
(\vec{x}_{n+1}, y_{n+1}) = \left( (\sqrt{\lambda}, 0, \ldots, 0), 0 \right)
\]

\[
(\vec{x}_{n+2}, y_{n+2}) = \left( (0, \sqrt{\lambda}, \ldots, 0), 0 \right)
\]

\[
\vdots
\]

\[
(\vec{x}_{n+d}, y_{n+d}) = \left( (0, 0, \ldots, \sqrt{\lambda}), 0 \right)
\]

Contribution to SSE from these extra, made-up examples is:

\[
\left( (\sqrt{\lambda}, 0, \ldots, 0) \cdot \vec{w} - 0 \right)^2 + \left( (0, \sqrt{\lambda}, \ldots, 0) \cdot \vec{w} - 0 \right)^2 + \cdots + \left( (0, 0, \ldots, \sqrt{\lambda}) \cdot \vec{w} - 0 \right)^2
\]

which is equivalent to

\[
(\sqrt{\lambda}w_1)^2 + (\sqrt{\lambda}w_2)^2 + \cdots + (\sqrt{\lambda}w_d)^2 = \lambda \|\vec{w}\|_2^2
\]

Ridge regression objective is same as OLS objective for training data augmented with these \( d \) examples
Uniqueness of minimizer for ridge regression objective

Augmented training data, now in matrix/vector form:

\[ A_{\text{aug}} := \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix} \in \mathbb{R}^{(n+d)\times d}, \quad \vec{b}_{\text{aug}} := \begin{bmatrix} \vec{b} \\ 0 \end{bmatrix} \in \mathbb{R}^{n+d} \]
Augmented training data, now in matrix/vector form:

$$A_{\text{aug}} := \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix} \in \mathbb{R}^{(n+d) \times d}, \quad \vec{b}_{\text{aug}} := \begin{bmatrix} \vec{b} \\ \vec{0} \end{bmatrix} \in \mathbb{R}^{n+d}$$

Normal equations based on augmented training data:

$$(A_{\text{aug}}^T A_{\text{aug}}) \vec{w} = A_{\text{aug}}^T \vec{b}_{\text{aug}}$$
Uniqueness of minimizer for ridge regression objective

Augmented training data, now in matrix/vector form:

\[
A_{\text{aug}} := \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix} \in \mathbb{R}^{(n+d) \times d}, \quad b_{\text{aug}} := \begin{bmatrix} \bar{b} \\ \bar{0} \end{bmatrix} \in \mathbb{R}^{n+d}
\]

Normal equations based on augmented training data:

\[
(A_{\text{aug}}^T A_{\text{aug}}) \bar{w} = A_{\text{aug}}^T b_{\text{aug}}
\]

Equivalent to:

\[
(A^T A + \lambda I) \bar{w} = A^T \bar{b}
\]
Uniqueness of minimizer for ridge regression objective

Augmented training data, now in matrix/vector form:

\[ A_{\text{aug}} := \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix} \in \mathbb{R}^{(n+d) \times d}, \quad \vec{b}_{\text{aug}} := \begin{bmatrix} \vec{b} \\ \vec{0} \end{bmatrix} \in \mathbb{R}^{n+d} \]

Normal equations based on augmented training data:

\[ (A_{\text{aug}}^T A_{\text{aug}}) \vec{w} = A_{\text{aug}}^T \vec{b}_{\text{aug}} \]

Equivalent to:

\[ (A^T A + \lambda I) \vec{w} = A^T \vec{b} \]

**Fact:** If \( \lambda > 0 \), then matrix \( A^T A + \lambda I \) is invertible!
Uniqueness of minimizer for ridge regression objective

Augmented training data, now in matrix/vector form:

\[ A_{\text{aug}} := \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix} \in \mathbb{R}^{(n+d) \times d}, \quad \vec{b}_{\text{aug}} := \begin{bmatrix} \vec{b} \\ 0 \end{bmatrix} \in \mathbb{R}^{n+d} \]

Normal equations based on augmented training data:

\[
(A_{\text{aug}}^T A_{\text{aug}}) \vec{w} = A_{\text{aug}}^T \vec{b}_{\text{aug}}
\]

Equivalent to:

\[
(A^T A + \lambda I) \vec{w} = A^T \vec{b}
\]

**Fact:** If \( \lambda > 0 \), then matrix \( A^T A + \lambda I \) is invertible!

So ridge regression objective (with parameter \( \lambda > 0 \)) has a unique minimizer

\[
\vec{w}_{\text{ridge},\lambda} := (A^T A + \lambda I)^{-1} A^T \vec{b}
\]
Data augmentation, in general

**Data augmentation**: Fabricate examples to encode *prior knowledge* or other inductive bias.
Data augmentation, in general

**Data augmentation:** Fabricate examples to encode *prior knowledge* or other inductive bias

**Example:** Computer vision

- Each feature vector $\vec{x}$ is an encoding of an image
Data augmentation, in general

**Data augmentation:** Fabricate examples to encode *prior knowledge* or other inductive bias

**Example:** Computer vision
- Each feature vector $\vec{x}$ is an encoding of an image

- **Prior knowledge:** Label should be invariant to rigid transformations and reflections
Data augmentation, in general

**Data augmentation:** Fabricate examples to encode *prior knowledge* or other inductive bias

**Example:** Computer vision

- Each feature vector $\vec{x}$ is an encoding of an image

- **Prior knowledge:** Label should be invariant to rigid transformations and reflections

- **Data augmentation:** For each example $(\vec{x}, y)$ in data set, generate many new examples $(\vec{x}', y)$ with the same label, where each $\vec{x}'$ is obtained by applying some rigid transformation or reflection
Postscript and recap

▶ Common to formulate ML method as optimization problem that trades-off between “data fitting objective” and a “regularizer”
  ▶ Many interpretations of regularization (e.g., Bayesian statistics, statistical learning theory, information theory)
  ▶ Ridge regression is “just” a particular instance
▶ Many other ways to encode inductive bias in ML method
  ▶ Data augmentation—make up data so the “data fitting objective” will “encourage” the learned predictor to have particular desirable properties
  ▶ Customized statistical models beyond standard IID model
  ▶ ...
Inductive Bias in Linear Classification
Inductive bias in linear classification

For linearly separable training data, there may be many linear classifiers that perfectly separate the data.
Inductive bias in linear classification

For linearly separable training data, there may be many linear classifiers that perfectly separate the data

Why might one linear classifier be preferred over another?

![Graph showing linear classification with training examples and decision boundary.]
Inductive bias in linear classification

For linearly separable training data, there may be many linear classifiers that perfectly separate the data.

Why might one linear classifier be preferred over another?

One possibility:

- Prefer decision boundary far from training examples.
Inductive bias in linear classification

For linearly separable training data, there may be many linear classifiers that perfectly separate the data.

Why might one linear classifier be preferred over another?

One possibility:
- Prefer decision boundary far from training examples
- I.e., find classifier with large “margins” around boundary
Decision boundary of linear classifier $f_{\vec{w}}(\vec{x}) = \mathbb{1}\{\vec{x} \cdot \vec{w} > 0\}$:

$$H = \{\vec{x} \in \mathbb{R}^d : \vec{x} \cdot \vec{w} = 0\}$$
Margins

Decision boundary of linear classifier \( f_{\vec{w}}(\vec{x}) = 1\{\vec{x} \cdot \vec{w} > 0\} \):

\[
H = \{ \vec{x} \in \mathbb{R}^d : \vec{x} \cdot \vec{w} = 0 \}
\]

- Distance from \( \vec{x} \) to decision boundary:

\[
\|\vec{x}\|_2 |\cos(\alpha)| = \frac{|\vec{x} \cdot \vec{w}|}{\|\vec{w}\|_2}
\]

Possible inductive bias: Larger margins are better!

Given linearly separable data \( S \), want to maximize the worst margin achieved on \( S \).
Margins

Decision boundary of linear classifier \( f_\vec{w}(\vec{x}) = 1\{\vec{x} \cdot \vec{w} > 0\} \):

\[
H = \{ \vec{x} \in \mathbb{R}^d : \vec{x} \cdot \vec{w} = 0 \}
\]

- Distance from \( \vec{x} \) to decision boundary:

\[
\|\vec{x}\|_2 |\cos(\alpha)| = \frac{|\vec{x} \cdot \vec{w}|}{\|\vec{w}\|_2}
\]

(Doesn’t change if \( \vec{w} \) is scaled by any \( r > 0 \))
Margins

Decision boundary of linear classifier $f_{\vec{w}}(\vec{x}) = \mathbb{1}\{\vec{x} \cdot \vec{w} > 0\}$:

$$H = \{\vec{x} \in \mathbb{R}^d : \vec{x} \cdot \vec{w} = 0\}$$

Distance from $\vec{x}$ to decision boundary:

$$\|\vec{x}\|_2 \left| \cos(\alpha) \right| = \frac{|\vec{x} \cdot \vec{w}|}{\|\vec{w}\|_2}$$

(Doesn’t change if $\vec{w}$ is scaled by any $r > 0$)

Say $f_{\vec{w}}$ achieves margin $\gamma > 0$ on example $(\vec{x}, y)$ if:

1. $f_{\vec{w}}(\vec{x}) = y$
2. Distance from $\vec{x}$ to the decision boundary is $\geq \gamma$
Margins

Decision boundary of linear classifier $f_{\vec{w}}(\vec{x}) = 1\{\vec{x} \cdot \vec{w} > 0\}$:

$$H = \{\vec{x} \in \mathbb{R}^d : \vec{x} \cdot \vec{w} = 0\}$$

- Distance from $\vec{x}$ to decision boundary:

$$\|\vec{x}\|_2 \cos(\alpha) = \frac{\|\vec{x} \cdot \vec{w}\|}{\|\vec{w}\|_2}$$

(Doesn’t change if $\vec{w}$ is scaled by any $r > 0$)

- Say $f_{\vec{w}}$ achieves margin $\gamma > 0$ on example $(\vec{x}, y)$ if:
  1. $f_{\vec{w}}(\vec{x}) = y$
  2. Distance from $\vec{x}$ to the decision boundary is $\geq \gamma$

(Possible) **inductive bias**: Larger margins are better!

Given linearly separable data $S$, want to maximize the worst margin achieved on $S$
Maximizing the worst margin achieved on training data

- Training feature vectors $\vec{x}$ closest to boundary satisfy

$$|\vec{x} \cdot \vec{w}| = \min_{(\vec{x}', y') \in S} |\vec{x}' \cdot \vec{w}|$$

- Distance of each such feature vector $\vec{x}$ from boundary:

$$|\vec{x} \cdot \vec{w}| / \|\vec{w}\|_2$$

- If $\vec{w}$ is scaled so that $|\vec{x} \cdot \vec{w}| = 1$, then distance is $1 / \|\vec{w}\|_2$

- So goal is: Find $\vec{w}$ that maximizes $1 / \|\vec{w}\|_2$ while satisfying

1. $f(\vec{w})(\vec{x}) = y$ for all $(\vec{x}, y) \in S$
2. $\min_{(\vec{x}, y) \in S} |\vec{x} \cdot \vec{w}| = 1$

(Maximizing $1 / \|\vec{w}\|_2$ is equivalent to minimizing $\|\vec{w}\|_2$)
Maximizing the worst margin achieved on training data

- Training feature vectors $\vec{x}$ closest to boundary satisfy

$$|\vec{x} \cdot \vec{w}| = \min_{(\vec{x}', y') \in \mathcal{S}} |\vec{x}' \cdot \vec{w}|$$

- Distance of each such feature vector $\vec{x}$ from boundary:

$$|\vec{x} \cdot \vec{w}| \|\vec{w}\|_2$$

- If $\vec{w}$ is scaled so that $|\vec{x} \cdot \vec{w}| = 1$, then distance is $\frac{1}{\|\vec{w}\|_2}$.

- So goal is: Find $\vec{w}$ that maximizes $\frac{1}{\|\vec{w}\|_2}$ while satisfying

1. $f(\vec{x}) = y$ for all $((\vec{x}, y)) \in \mathcal{S}$
2. $\min_{(\vec{x}, y) \in \mathcal{S}} |\vec{x} \cdot \vec{w}| = 1$

(Maximizing $\frac{1}{\|\vec{w}\|_2}$ is equivalent to minimizing $\|\vec{w}\|_2$.)
Maximizing the worst margin achieved on training data

- Training feature vectors $\vec{x}$ closest to boundary satisfy
  \[ |\vec{x} \cdot \vec{w}| = \min_{(\vec{x}',y') \in S} |\vec{x}' \cdot \vec{w}| \]

- Distance of each such feature vector $\vec{x}$ from boundary:
  \[ \frac{|\vec{x} \cdot \vec{w}|}{\|\vec{w}\|_2} \]

If $\vec{w}$ is scaled so that $|\vec{x} \cdot \vec{w}| = 1$, then distance is $\frac{1}{\|\vec{w}\|_2}$.

So goal is: Find $\vec{w}$ that maximizes $\frac{1}{\|\vec{w}\|_2}$ while satisfying:

1. $f(\vec{x}) = y$ for all $(\vec{x}, y) \in S$
2. $\min_{(\vec{x},y) \in S} |\vec{x} \cdot \vec{w}| = 1$

(Maximizing $\frac{1}{\|\vec{w}\|_2}$ is equivalent to minimizing $\|\vec{w}\|_2$.)
Maximizing the worst margin achieved on training data

- Training feature vectors $\vec{x}$ closest to boundary satisfy
  \[ |\vec{x} \cdot \vec{w}| = \min_{(\vec{x}', y') \in S} |\vec{x}' \cdot \vec{w}| \]

- Distance of each such feature vector $\vec{x}$ from boundary:
  \[ \frac{|\vec{x} \cdot \vec{w}|}{\|\vec{w}\|_2} \]

- If $\vec{w}$ is scaled so that $|\vec{x} \cdot \vec{w}| = 1$, then distance is $\frac{1}{\|\vec{w}\|_2}$
Maximizing the worst margin achieved on training data

- Training feature vectors $\vec{x}$ closest to boundary satisfy

$$|\vec{x} \cdot \vec{w}| = \min_{(\vec{x}', y') \in S} |\vec{x}' \cdot \vec{w}|$$

- Distance of each such feature vector $\vec{x}$ from boundary:

$$\frac{|\vec{x} \cdot \vec{w}|}{\|\vec{w}\|_2}$$

- If $\vec{w}$ is scaled so that $|\vec{x} \cdot \vec{w}| = 1$, then distance is $\frac{1}{\|\vec{w}\|_2}$

- So goal is: Find $\vec{w}$ that maximizes $\frac{1}{\|\vec{w}\|_2}$ while satisfying

1. $f_{\vec{w}}(\vec{x}) = y$ for all $(\vec{x}, y) \in S$
2. $\min_{(\vec{x}, y) \in S} |\vec{x} \cdot \vec{w}| = 1$
Maximizing the worst margin achieved on training data

- Training feature vectors $\vec{x}$ closest to boundary satisfy
  \[
  |\vec{x} \cdot \vec{w}| = \min_{(\vec{x}', y') \in S} |\vec{x}' \cdot \vec{w}|
  \]

- Distance of each such feature vector $\vec{x}$ from boundary:
  \[
  \frac{|\vec{x} \cdot \vec{w}|}{\|\vec{w}\|_2}
  \]

- If $\vec{w}$ is scaled so that $|\vec{x} \cdot \vec{w}| = 1$, then distance is $\frac{1}{\|\vec{w}\|_2}$

- So goal is: Find $\vec{w}$ that maximizes $\frac{1}{\|\vec{w}\|_2}$ while satisfying

1. $f_{\vec{w}}(\vec{x}) = y$ for all $(\vec{x}, y) \in S$
2. $\min_{(\vec{x}, y) \in S} |\vec{x} \cdot \vec{w}| = 1$

(Maximizing $\frac{1}{\|\vec{w}\|_2}$ is equivalent to minimizing $\|\vec{w}\|_2$)
Maximum margin linear classifiers

**Theorem.** If training data $S$ is linearly separable, then weight vector $\vec{w}$ of smallest Euclidean norm satisfying

1. $f_{\vec{w}}(\vec{x}) = y$ for all $(\vec{x}, y) \in S$

2. $\min_{(\vec{x}, y) \in S} |\vec{x} \cdot \vec{w}| = 1$

determines the linear classifier $f_{\vec{w}}(\vec{x}) = 1 \{\vec{x} \cdot \vec{w} > 0\}$ that maximizes the worst margin achieved on training data.
Theorem. If training data $S$ is linearly separable, then weight vector $\vec{w}$ of smallest Euclidean norm satisfying

1. $f_{\vec{w}}(\vec{x}) = y$ for all $(\vec{x}, y) \in S$
2. $\min_{(\vec{x}, y) \in S} |\vec{x} \cdot \vec{w}| = 1$

determines the linear classifier $f_{\vec{w}}(\vec{x}) = 1 \{\vec{x} \cdot \vec{w} > 0\}$ that maximizes the worst margin achieved on training data.

ML method based on solving this opt. problem:

Support Vector Machine (SVM)
a.k.a. maximum margin linear classifier

(Vapnik & Chervonenkis, 1968)
Theorem. If training data $S$ is linearly separable, then weight vector $\vec{w}$ of smallest Euclidean norm satisfying

1. $f_{\vec{w}}(\vec{x}) = y$ for all $(\vec{x}, y) \in S$
2. $\min_{(\vec{x}, y) \in S} |\vec{x} \cdot \vec{w}| = 1$

determines the linear classifier $f_{\vec{w}}(\vec{x}) = 1\{\vec{x} \cdot \vec{w} > 0\}$ that maximizes the worst margin achieved on training data.

ML method based on solving this opt. problem: Support Vector Machine (SVM) a.k.a. maximum margin linear classifier

(Vapnik & Chervonenkis, 1968)

(Usually SVM optimization problem is written in slightly different form)
Example: Iris data set

- **Two** classes of irises: Setosa (0), Versicolor/Virginica (1)
- Two numerical features:
  - $x_1$: ratio of sepal width to sepal length
  - $x_2$: ratio of petal width to petal length
- 120 training data (40 from class 0, 80 from class 1)
Example: Iris data set

- **Two** classes of irises: Setosa (0), Versicolor/Virginica (1)
- Two numerical features:
  - $x_1$: ratio of sepal width to sepal length
  - $x_2$: ratio of petal width to petal length
- 120 training data (40 from class 0, 80 from class 1)
Example: Iris data set

- **Two** classes of irises: Setosa (0), Versicolor/Virginica (1)
- Two numerical features:
  - $x_1$: ratio of sepal width to sepal length
  - $x_2$: ratio of petal width to petal length
- 120 training data (40 from class 0, 80 from class 1)

![Iris data set graph]

Use affine feature expansion

$$\varphi(\vec{x}) = (\vec{x}, 1)$$

so weight vector $(\vec{w}, b) \in \mathbb{R}^3$ defines affine classifier

$$f_{\vec{w}, b}(\vec{x}) = 1\{\vec{x} \cdot \vec{w} + b > 0\}$$
Example: Results on iris data set

Perceptron on iris data set

- Ratio of sepal width to sepal length
- Ratio of petal width to petal length
- Decision boundary

On linearly separable data, iterative optimization algorithm for maximizing logistic regression likelihood $\rightarrow$ same "maximum margin" linear classifier as SVM!
Example: Results on iris data set

SVM on iris data set

On linearly separable data, iterative optimization algorithm for maximizing logistic regression likelihood → same "maximum margin" linear classifier as SVM!
Example: Results on iris data set

Logistic regression (approx.) MLE on iris data set (after $10^2$ iterations)
Example: Results on iris data set

Logistic regression (approx.) MLE on iris data set (after $10^3$ iterations)

- Setosa (0)
- Versicolor/Virginica (1)

Decision boundary
Example: Results on iris data set

Logistic regression (approx.) MLE on iris data set (after $10^4$ iterations)

On linearly separable data, iterative optimization algorithm for maximizing logistic regression likelihood $\to$ same "maximum margin" linear classifier as SVM!
Example: Results on iris data set

Logistic regression (approx.) MLE on iris data set (after $10^5$ iterations)

On linearly separable data, iterative optimization algorithm for maximizing logistic regression likelihood $\rightarrow$ same "maximum margin" linear classifier as SVM!
Example: Results on iris data set

Logistic regression (approx.) MLE on iris data set (after $10^6$ iterations)

On linearly separable data, iterative optimization algorithm for maximizing logistic regression likelihood $\rightarrow$ same "maximum margin" linear classifier as SVM!
Logistic regression (approx.) MLE on iris data set (after $10^7$ iterations)

Setosa (0)  
Versicolor/Virginica (1)  
decision boundary

On linearly separable data, iterative optimization algorithm for maximizing logistic regression likelihood $\rightarrow$ same "maximum margin" linear classifier as SVM!
Example: Results on iris data set

On linearly separable data, iterative optimization algorithm for maximizing logistic regression likelihood → same “maximum margin” linear classifier as SVM!
Taking inspiration from ridge regression, consider “regularized” log-likelihood for logistic regression:

\[
\ln L(\tilde{w}) - \lambda \| \tilde{w} \|^2
\]

where \( L(\tilde{w}) \) is likelihood of parameter \( \tilde{w} \) given training data \( S \) in logistic regression model.
Taking inspiration from ridge regression, consider “regularized” log-likelihood for logistic regression:

\[ \ln L(\vec{w}) - \lambda \| \vec{w} \|^2 \]

where \( L(\vec{w}) \) is likelihood of parameter \( \vec{w} \) given training data \( S \) in logistic regression model

- \( \lambda \) is a hyperparameter that controls how much we pay attention to the second concern (compared to the first one)
Taking inspiration from ridge regression, consider “regularized” log-likelihood for logistic regression:

$$\ln L(\vec{w}) - \lambda \| \vec{w} \|^2_2$$

where $L(\vec{w})$ is likelihood of parameter $\vec{w}$ given training data $\mathcal{S}$ in logistic regression model

- $\lambda$ is a hyperparameter that controls how much we pay attention to the second concern (compared to the first one)
- $\lambda = 0$ corresponds to logistic regression MLE
Taking inspiration from ridge regression, consider “regularized” log-likelihood for logistic regression:

\[
\ln L(\vec{w}) - \lambda \|\vec{w}\|^2_2
\]

where \(L(\vec{w})\) is likelihood of parameter \(\vec{w}\) given training data \(S\) in logistic regression model.

- \(\lambda\) is a hyperparameter that controls how much we pay attention to the second concern (compared to the first one).
- \(\lambda = 0\) corresponds to logistic regression MLE.
- \(\lambda > 0\) implies unique maximizer, but still requires numerical optimization procedure.
Taking inspiration from ridge regression, consider “regularized” log-likelihood for logistic regression:

$$\ln L(\vec{w}) - \lambda \|\vec{w}\|^2_2$$

where $L(\vec{w})$ is likelihood of parameter $\vec{w}$ given training data $S$ in logistic regression model

- $\lambda$ is a hyperparameter that controls how much we pay attention to the second concern (compared to the first one)
- $\lambda = 0$ corresponds to logistic regression MLE
- $\lambda > 0$ implies unique maximizer, but still requires numerical optimization procedure

Trades-off between fitting data (log-likelihood) and desire to achieve fit with a short $\vec{w}$ (large margins)
Example: Regularized MLE on iris data

Logistic regression regularized MLE on iris data set ($\lambda = 0.1$)

Setosa (0)
Versicolor/Virginica (1)
decision boundary
Aside: Logarithmic loss

Alternative interpretation of logistic regression negative log-likelihood:

- Let $p_{\vec{w}}(\vec{x})$ denote $\text{Pr}_{\vec{w}}(Y = 1 \mid \vec{X} = \vec{x})$ in logistic regression model with parameter $\vec{w}$:

$$p_{\vec{w}}(\vec{x}) := \text{logistic}(\vec{x} \cdot \vec{w})$$

so that

$$\ln \left( \frac{p_{\vec{w}}(\vec{x})}{1 - p_{\vec{w}}(\vec{x})} \right) = \vec{x} \cdot \vec{w} \quad \text{(log odds ratio)}$$

- Term corresponding to $(\vec{x}_i, y_i) \in \mathbb{R}^d \times \{0, 1\}$ in negative log-likelihood:

$$\ln(1 + e^{\vec{x}_i \cdot \vec{w}}) - y_i \vec{x}_i \cdot \vec{w} = \ln \left( \frac{1}{1 - p_{\vec{w}}(\vec{x}_i)} \right) + (1 - y_i) \ln \left( \frac{1}{1 - p_{\vec{w}}(\vec{x}_i)} \right)$$

- If $y_i = 1$, then term decreases as $p_{\vec{w}}(\vec{x}_i) \to 1$

- If $y_i = 0$, then term decreases as $p_{\vec{w}}(\vec{x}_i) \to 0$

- Term as function of $p = p_{\vec{w}}(\vec{x}_i)$ is called log(arithmic) loss $\ell_{\text{log}}(y, p)$ (a.k.a. cross entropy)

Upshot: Maximizing log-likelihood is same as minimizing sum of log losses
Aside: Logarithmic loss

Alternative interpretation of logistic regression negative log-likelihood:

- Let $p_{\tilde{w}}(\vec{x})$ denote $\Pr_{\tilde{w}}(Y = 1 \mid \vec{X} = \vec{x})$ in logistic regression model with parameter $\tilde{w}$:

$$p_{\tilde{w}}(\vec{x}) := \text{logistic}(\vec{x} \cdot \tilde{w})$$

so that

$$\ln \left( \frac{p_{\tilde{w}}(\vec{x})}{1 - p_{\tilde{w}}(\vec{x})} \right) = \vec{x} \cdot \tilde{w} \quad \text{ (log odds ratio)}$$

- Term corresponding to $(\vec{x}_i, y_i) \in \mathbb{R}^d \times \{0, 1\}$ in negative log-likelihood:

$$\ln(1 + e^{\vec{x}_i \cdot \tilde{w}}) - y_i \vec{x}_i \cdot \tilde{w} = \ln \left( 1 + \frac{p_{\tilde{w}}(\vec{x})}{1 - p_{\tilde{w}}(\vec{x})} \right) - y_i \ln \left( \frac{p_{\tilde{w}}(\vec{x})}{1 - p_{\tilde{w}}(\vec{x})} \right)$$

$$= y_i \ln \left( \frac{1}{p_{\tilde{w}}(\vec{x}_i)} \right) + (1 - y_i) \ln \left( \frac{1}{1 - p_{\tilde{w}}(\vec{x}_i)} \right)$$
Aside: Logarithmic loss

Alternative interpretation of logistic regression negative log-likelihood:

- Let $p_{\vec{w}}(\vec{x})$ denote $\Pr_{\vec{w}}(Y = 1 \mid \vec{X} = \vec{x})$ in logistic regression model with parameter $\vec{w}$:

$$p_{\vec{w}}(\vec{x}) := \text{logistic}(\vec{x} \cdot \vec{w})$$

so that

$$\ln \left( \frac{p_{\vec{w}}(\vec{x})}{1 - p_{\vec{w}}(\vec{x})} \right) = \vec{x} \cdot \vec{w} \quad \text{(log odds ratio)}$$

- Term corresponding to $(\vec{x}_i, y_i) \in \mathbb{R}^d \times \{0, 1\}$ in negative log-likelihood:

$$\ln(1 + e^{\vec{x}_i \cdot \vec{w}}) - y_i \vec{x}_i \cdot \vec{w} = \ln \left( 1 + \frac{p_{\vec{w}}(\vec{x}_i)}{1 - p_{\vec{w}}(\vec{x}_i)} \right) - y_i \ln \left( \frac{p_{\vec{w}}(\vec{x}_i)}{1 - p_{\vec{w}}(\vec{x}_i)} \right)$$

$$= y_i \ln \left( \frac{1}{p_{\vec{w}}(\vec{x}_i)} \right) + (1 - y_i) \ln \left( \frac{1}{1 - p_{\vec{w}}(\vec{x}_i)} \right)$$

- If $y_i = 1$, then term decreases as $p_{\vec{w}}(\vec{x}_i) \to 1$
Aside: Logarithmic loss

Alternative interpretation of logistic regression negative log-likelihood:

Let $p_\vec{w}(\vec{x})$ denote $\Pr_\vec{w}(Y = 1 \mid \vec{X} = \vec{x})$ in logistic regression model with parameter $\vec{w}$:

$$p_\vec{w}(\vec{x}) := \text{logistic}(\vec{x} \cdot \vec{w})$$

so that

$$\ln \left( \frac{p_\vec{w}(\vec{x})}{1 - p_\vec{w}(\vec{x})} \right) = \vec{x} \cdot \vec{w} \quad \text{(log odds ratio)}$$

Term corresponding to $(\vec{x}_i, y_i) \in \mathbb{R}^d \times \{0, 1\}$ in negative log-likelihood:

$$\ln(1 + e^{\vec{x}_i \cdot \vec{w}}) - y_i \vec{x}_i \cdot \vec{w} = \ln \left( 1 + \frac{p_\vec{w}(\vec{x}_i)}{1 - p_\vec{w}(\vec{x}_i)} \right) - y_i \ln \left( \frac{p_\vec{w}(\vec{x}_i)}{1 - p_\vec{w}(\vec{x}_i)} \right)$$

$$= y_i \ln \left( \frac{1}{p_\vec{w}(\vec{x}_i)} \right) + (1 - y_i) \ln \left( \frac{1}{1 - p_\vec{w}(\vec{x}_i)} \right)$$

If $y_i = 1$, then term decreases as $p_\vec{w}(\vec{x}_i) \to 1$

If $y_i = 0$, then term decreases as $p_\vec{w}(\vec{x}_i) \to 0$
Aside: Logarithmic loss

Alternative interpretation of logistic regression negative log-likelihood:

- Let $p_{\vec{w}}(\vec{x})$ denote $Pr_{\vec{w}}(Y = 1 \mid \vec{X} = \vec{x})$ in logistic regression model with parameter $\vec{w}$:
  \[
  p_{\vec{w}}(\vec{x}) := \text{logistic}(\vec{x} \cdot \vec{w}) \quad \text{so that} \quad \ln \left( \frac{p_{\vec{w}}(\vec{x})}{1 - p_{\vec{w}}(\vec{x})} \right) = \vec{x} \cdot \vec{w} \quad \text{(log odds ratio)}
  \]

- Term corresponding to $(\vec{x}_i, y_i) \in \mathbb{R}^d \times \{0, 1\}$ in negative log-likelihood:
  \[
  \ln(1 + e^{\vec{x}_i \cdot \vec{w}}) - y_i \vec{x}_i \cdot \vec{w} = \ln \left( 1 + \frac{p_{\vec{w}}(\vec{x})}{1 - p_{\vec{w}}(\vec{x})} \right) - y_i \ln \left( \frac{p_{\vec{w}}(\vec{x})}{1 - p_{\vec{w}}(\vec{x})} \right)
  = y_i \ln \left( \frac{1}{p_{\vec{w}}(\vec{x}_i)} \right) + (1 - y_i) \ln \left( \frac{1}{1 - p_{\vec{w}}(\vec{x}_i)} \right)
  \]

  - If $y_i = 1$, then term decreases as $p_{\vec{w}}(\vec{x}_i) \to 1$
  - If $y_i = 0$, then term decreases as $p_{\vec{w}}(\vec{x}_i) \to 0$

- Term as function of $p = p_{\vec{w}}(\vec{x}_i)$ is called logarithmic loss $\ell_{\text{log}}(y, p)$ (a.k.a. cross entropy)
Aside: Logarithmic loss

Alternative interpretation of logistic regression negative log-likelihood:

- Let $p_{\vec{w}}(\vec{x})$ denote $\Pr_{\vec{w}}(Y = 1 \mid \vec{X} = \vec{x})$ in logistic regression model with parameter $\vec{w}$:
  
  $$p_{\vec{w}}(\vec{x}) := \text{logistic}(\vec{x} \cdot \vec{w})$$

  so that
  
  $$\ln \left( \frac{p_{\vec{w}}(\vec{x})}{1 - p_{\vec{w}}(\vec{x})} \right) = \vec{x} \cdot \vec{w} \quad \text{(log odds ratio)}$$

- Term corresponding to $(\vec{x}_i, y_i) \in \mathbb{R}^d \times \{0, 1\}$ in negative log-likelihood:
  
  $$\ln(1 + e^{\vec{x}_i \cdot \vec{w}}) - y_i \vec{x}_i \cdot \vec{w} = \ln \left( 1 + \frac{p_{\vec{w}}(\vec{x}_i)}{1 - p_{\vec{w}}(\vec{x}_i)} \right) - y_i \ln \left( \frac{p_{\vec{w}}(\vec{x}_i)}{1 - p_{\vec{w}}(\vec{x}_i)} \right)$$

  $$= y_i \ln \left( \frac{1}{p_{\vec{w}}(\vec{x}_i)} \right) + (1 - y_i) \ln \left( \frac{1}{1 - p_{\vec{w}}(\vec{x}_i)} \right)$$

  - If $y_i = 1$, then term decreases as $p_{\vec{w}}(\vec{x}_i) \to 1$
  - If $y_i = 0$, then term decreases as $p_{\vec{w}}(\vec{x}_i) \to 0$

- Term as function of $p = p_{\vec{w}}(\vec{x}_i)$ is called log(arithmetic) loss $\ell_{\text{log}}(y, p)$ (a.k.a. cross entropy)

Upshot: Maximizing log-likelihood is same as minimizing sum of log losses
Sum of log losses versus number of classification mistakes

\[ \sum \text{log losses} = \sum_{i=1}^{n} \ell_{\log}(y_i, \vec{w}^{\top} \vec{x}_i) \]

as surrogate for sum of zero-one losses

\[ \sum_{i=1}^{n} \ell_{0/1}(y_i, p_{\vec{w}^{\top} \vec{x}_i}) \]

where

\[ \ell_{0/1}(y, p) := \begin{cases} 0 & \text{if } y = \frac{1}{2} \text{ and } p > \frac{1}{2} \\ 1 & \text{otherwise} \end{cases} \]
Sum of log losses versus number of classification mistakes

\[
\sum (\text{scaled}) \log \text{losses}
\]

as surrogate for sum of zero-one losses

\[
\frac{1}{\ln 2} \cdot \sum_{i=1}^{n} \ell_{\log}(y_i, \hat{p}(\vec{x}_i))
\]

where

\[
\ell_{0/1}(y, \hat{p}) :=
\begin{cases} 1 & \text{if } 1 \{ \hat{p} > 1/2 \} \neq y \\ 0 & \text{otherwise} \end{cases}
\]
Sum of (scaled) log losses as surrogate for sum of zero-one losses (i.e., number of classification mistakes)

\[
\frac{1}{\ln 2} \cdot \sum_{i=1}^{n} \ell_{\log}(y_i, p_{\vec{w}}(\vec{x}_i))
\]

\[
\sum_{i=1}^{n} \ell_{0/1}(y_i, p_{\vec{w}}(\vec{x}_i))
\]

where

\[
\ell_{0/1}(y, p) := \begin{cases} 
1 & \text{if } 1\{p > 1/2\} \neq y \\
0 & \text{otherwise}
\end{cases}
\]
Desire for large margins = example of inductive bias in linear classification
SVM = maximize the worst margin achieved on training data
Can explicitly combine this inductive bias with logistic regression MLE
In general:
\[
\text{[data fitting objective]} + \lambda \text{ [regularizer]}
\]
Algorithmic regularization: Optimization algorithms can have “implicit” inductive biases