Chapter ML:III

III. Linear Models

- Logistic Regression
- Loss Computation in Detail
- Overfitting
- Regularization
- Gradient Descent in Detail

Logistic Regression

Binary Classification Problems

Setting:

- X is a multiset of feature vectors from an inner product space $\mathbf{X}$, $\mathbf{X} \subseteq \mathbf{R}^p$.
- $C = \{0, 1\}$ is a set of two classes. Similarly: $\{-1, 1\}$, $\{\ominus, \oplus\}$, $\{\text{no}, \text{yes}\}$, etc.
- $D = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\} \subseteq X \times C$ is a multiset of examples.

Learning task:

- Fit the examples in D with a logistic model function.

Logistic Regression

Binary Classification Problems

Setting:

- X is a multiset of feature vectors from an inner product space $\mathbf{X}$, $\mathbf{X} \subseteq \mathbf{R}^p$.
- $C = \{0, 1\}$ is a set of two classes. Similarly: $\{-1, 1\}$, $\{\ominus, \oplus\}$, $\{\text{no}, \text{yes}\}$, etc.
- $D = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\} \subseteq X \times C$ is a multiset of examples.

Learning task:

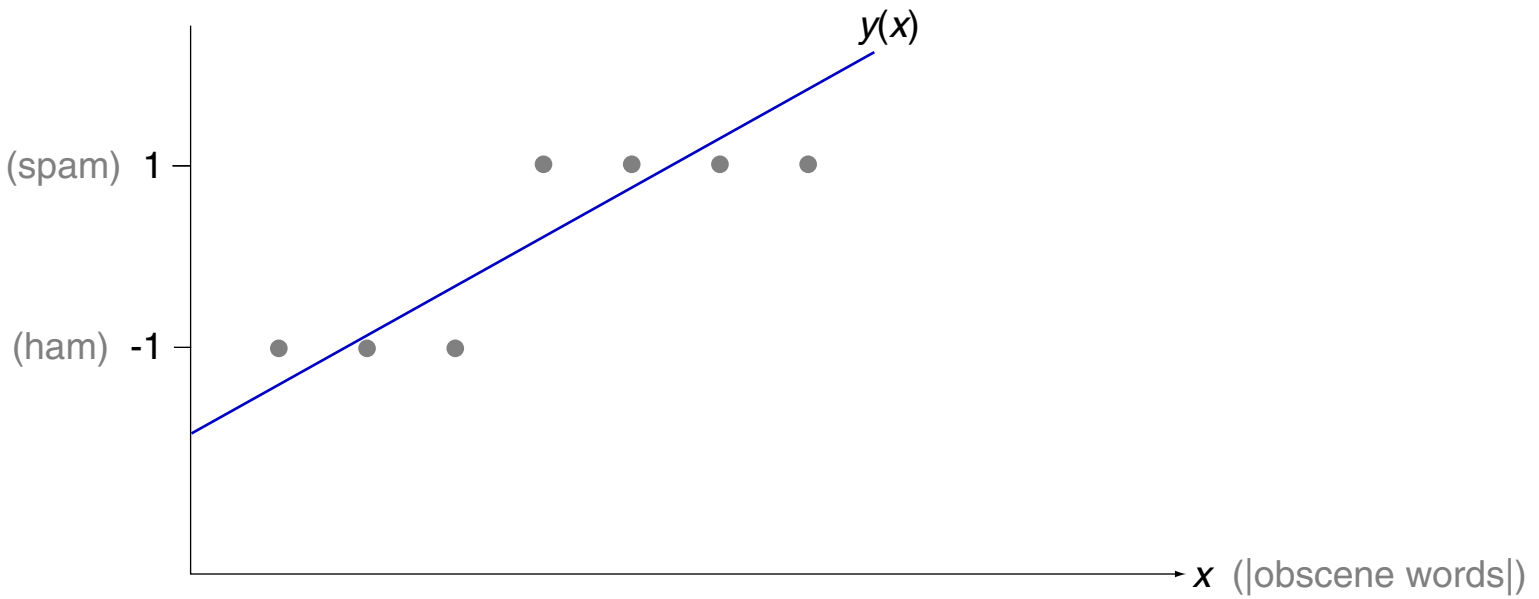
- Fit the examples in D with a logistic model function.

Examples for binary classification problems:

- Is an email spam or ham?
- Is a patient infected or healthy?
- Is a bank customer creditworthy or not?

Logistic Regression

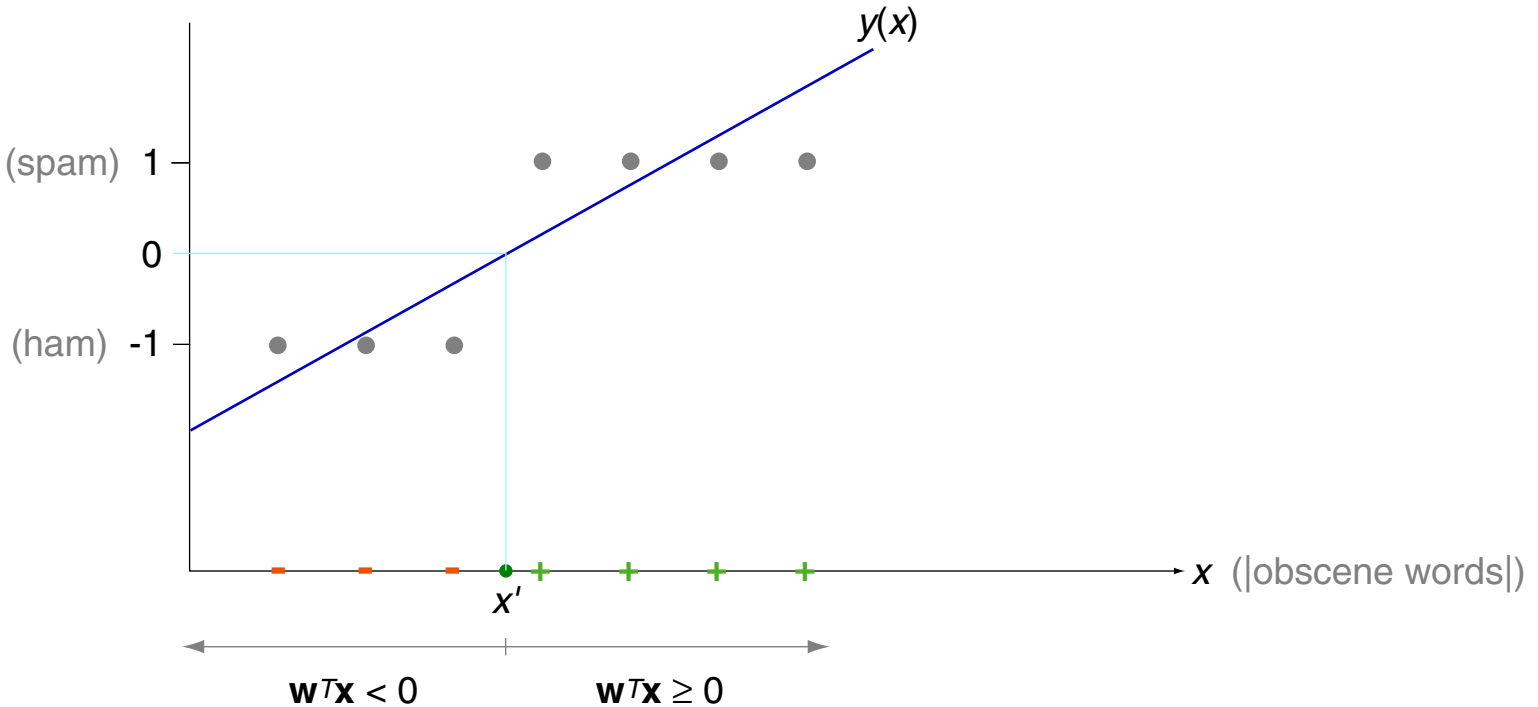
Linear Regression



- Linear regression: $y(\mathbf{x}) \overset{(\star)}{=} \mathbf{w}^T\mathbf{x}$

Logistic Regression

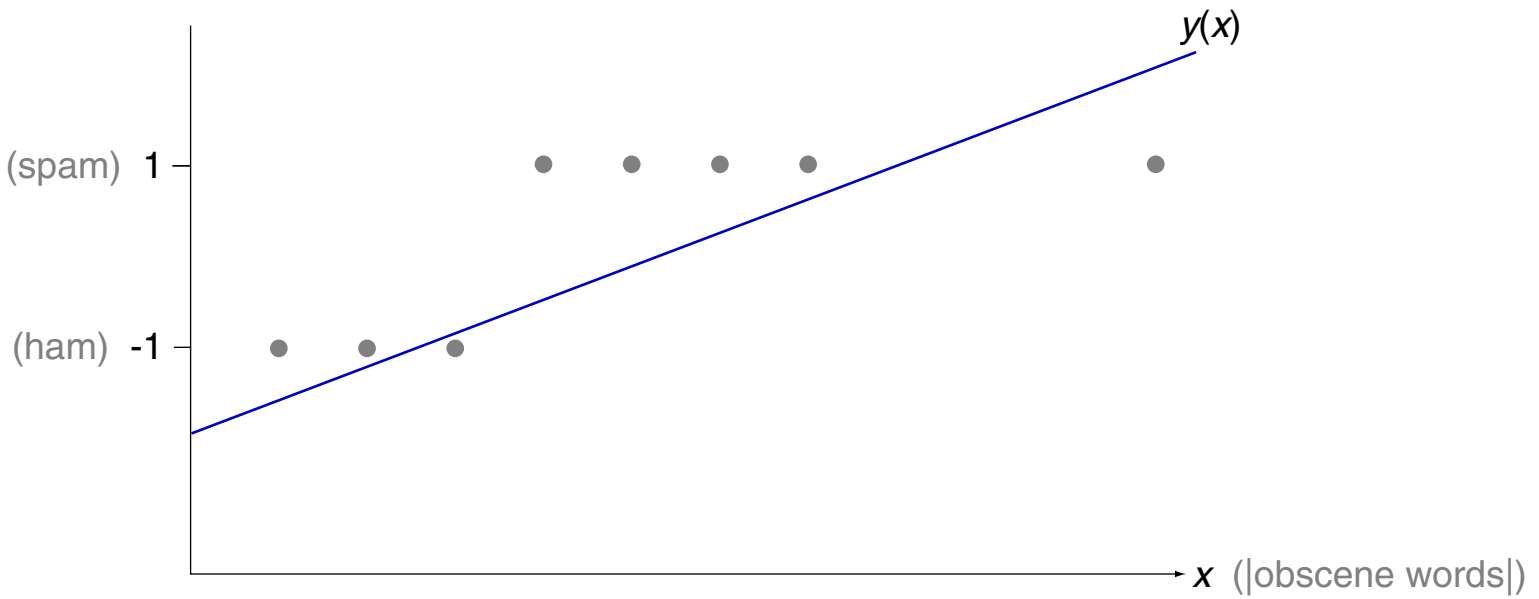
Linear Regression (continued)



- Linear regression: $y(\mathbf{x}) \overset{(\star)}{=} \mathbf{w}^T\mathbf{x}$
- Classification: Predict $\begin{cases} \text{“spam”, if } \mathbf{w}^T\mathbf{x} \geq 0 \\ \text{“ham”, if } \mathbf{w}^T\mathbf{x} < 0 \end{cases}$

Logistic Regression

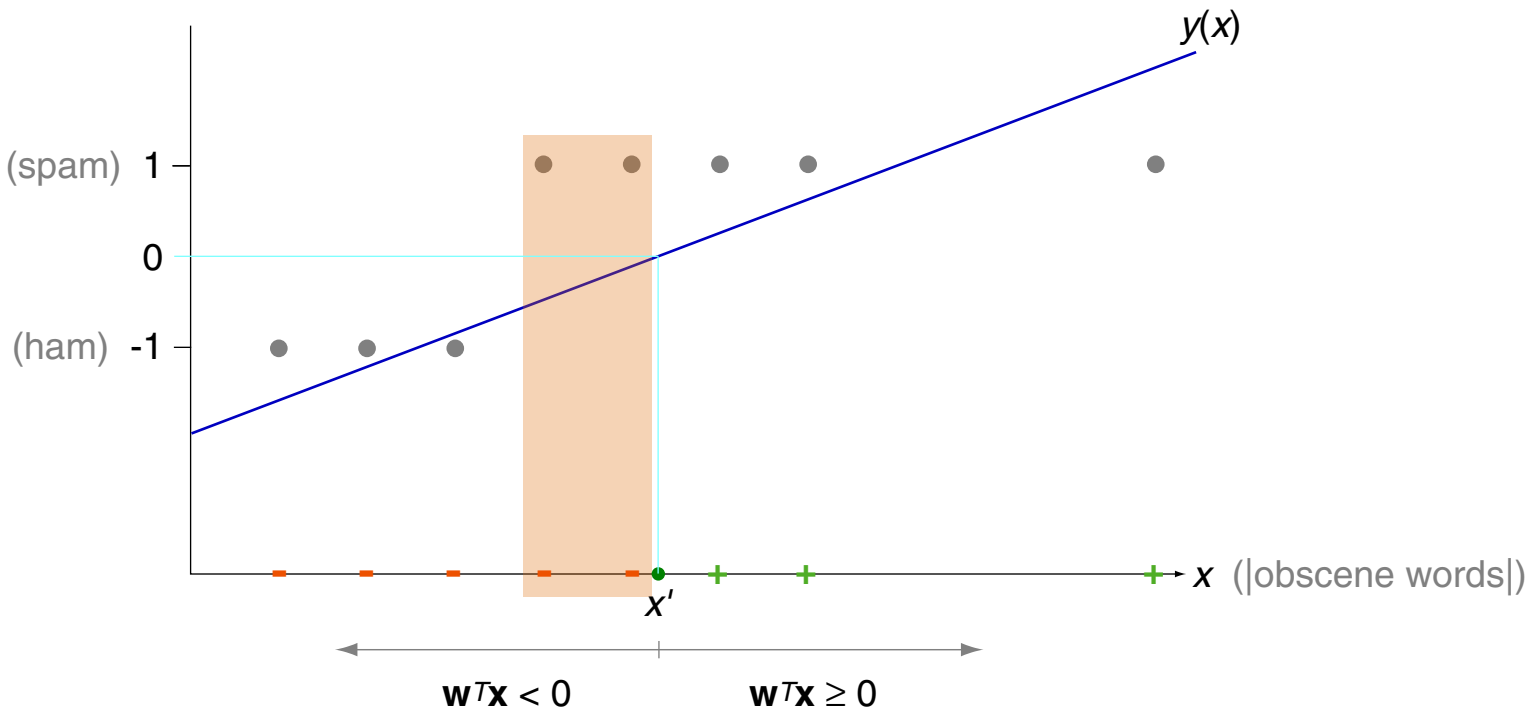
Linear Regression (continued)



- Linear regression: $y(\mathbf{x}) \overset{(\star)}{=} \mathbf{w}^T\mathbf{x}$
- Classification: Predict $\begin{cases} \text{“spam”}, & \text{if } \mathbf{w}^T\mathbf{x} \geq 0 \\ \text{“ham”}, & \text{if } \mathbf{w}^T\mathbf{x} < 0 \end{cases}$

Logistic Regression

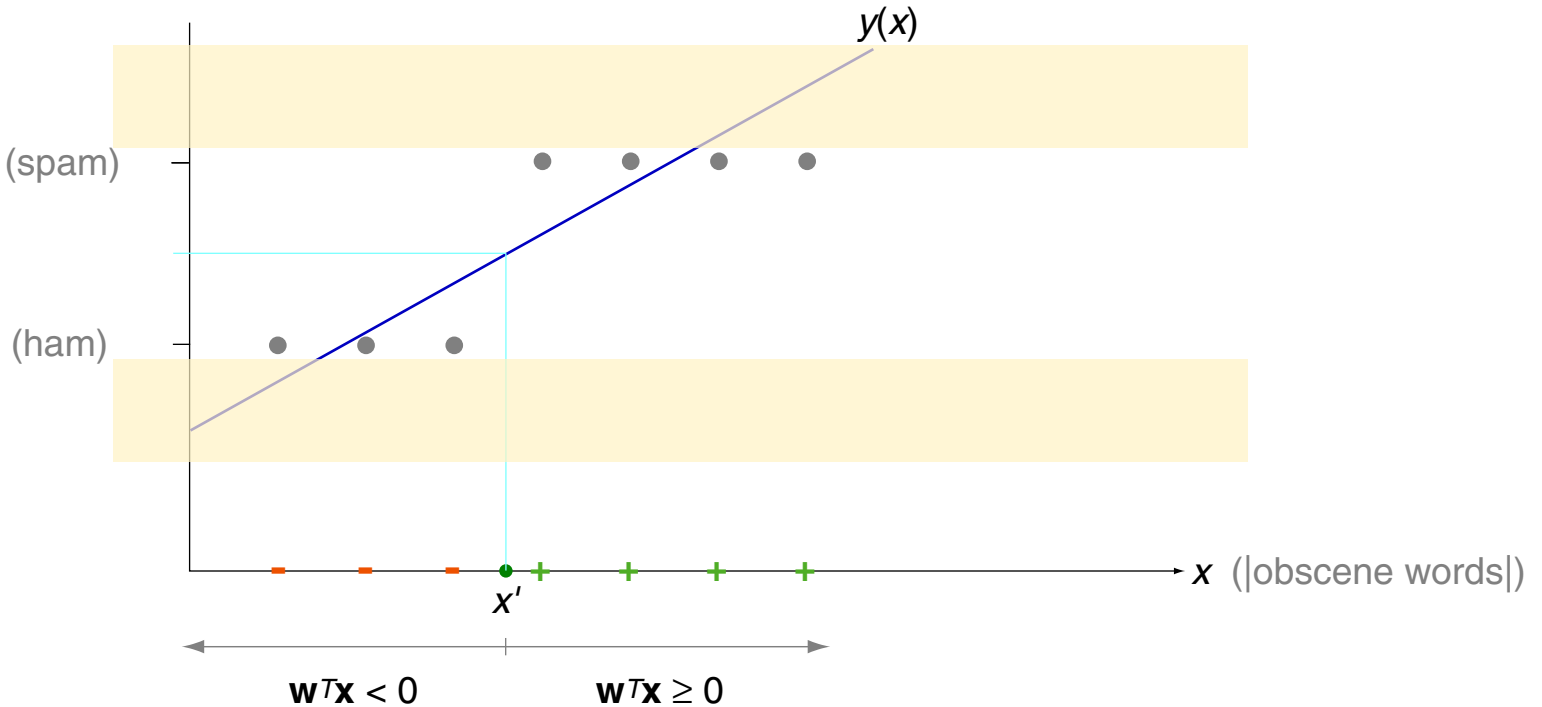
Linear Regression (continued)



- Linear regression: $y(\mathbf{x}) \overset{(\star)}{=} \mathbf{w}^T\mathbf{x}$
- Classification: Predict $\begin{cases} \text{“spam”}, & \text{if } \mathbf{w}^T\mathbf{x} \geq 0 \\ \text{“ham”}, & \text{if } \mathbf{w}^T\mathbf{x} < 0 \end{cases}$

Logistic Regression

Linear Regression (continued)



Restrict the range of $y(\mathbf{x})$ to reflect the two-class classification semantics:

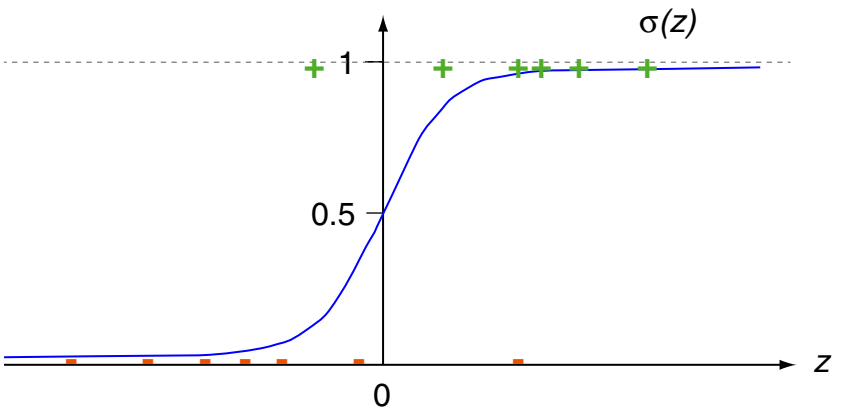
$$-1 \leq y(\mathbf{x}) \leq 1 \qquad \text{or} \qquad 0 \leq y(\mathbf{x}) \leq 1$$

Remarks:

($\star$) Recap. We consider the feature vector $\mathbf{x}$ in its extended form when used as operand in a scalar product with the weight vector, $\mathbf{w}^T\mathbf{x}$, and consequently, when noted as argument of the model function, $y(\mathbf{x})$. I.e., $\mathbf{x} = (1, x_1, \ldots, x_p)^T \in \mathbf{R}^{p+1}$, and $x_0 = 1$.

Logistic Regression

Sigmoid (Logistic) Function

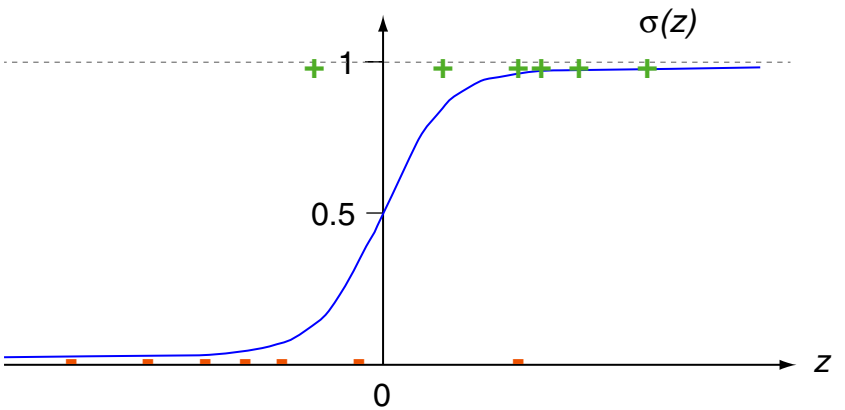


Sigmoid function

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Logistic Regression

Sigmoid (Logistic) Function (continued)



Linear regression

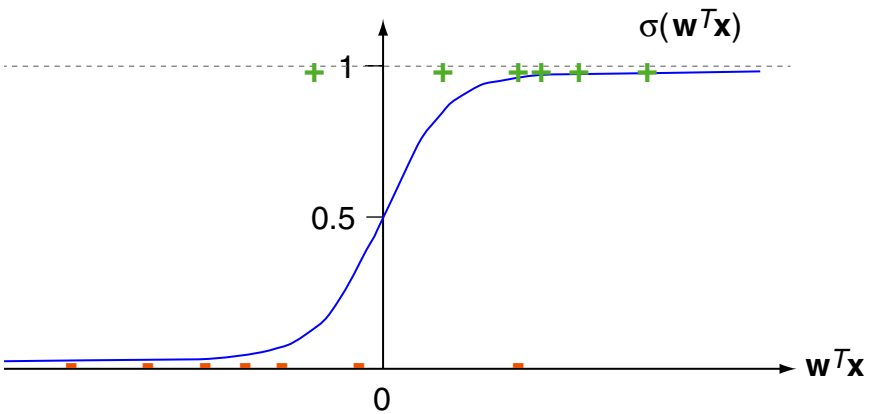
$$\mathbf{w}^T\mathbf{x}$$

Sigmoid function

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Logistic Regression

Sigmoid (Logistic) Function (continued)

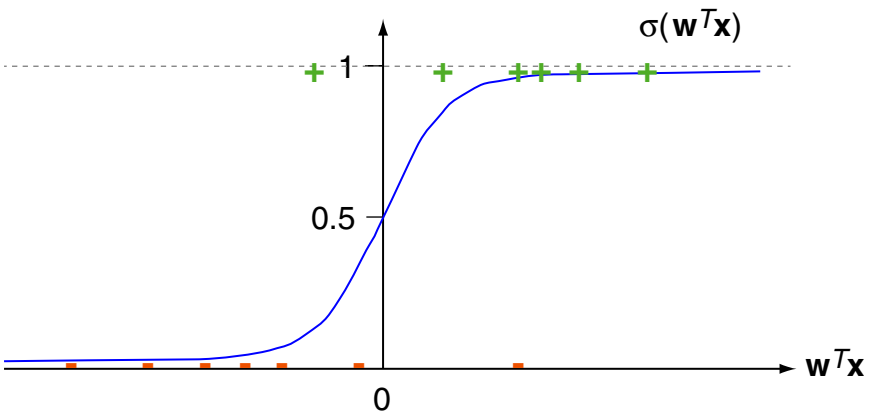


Linear regression ∘ Sigmoid function ⇝ Logistic model function

$$\mathbf{w}^T\mathbf{x} \quad \circ \quad \sigma(z) = \frac{1}{1+e^{-z}} \quad \rightsquigarrow \quad y(\mathbf{x}) \equiv \sigma(\mathbf{w}^T\mathbf{x}) \overset{(\star)}{=} \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$$

Logistic Regression

Sigmoid (Logistic) Function (continued)



Linear regression $\quad$ Sigmoid function $\quad$ Logistic model function

$$\mathbf{w}^T\mathbf{x} \quad \circ \quad \sigma(z) = \frac{1}{1+e^{-z}} \quad \rightsquigarrow \quad y(\mathbf{x}) \equiv \sigma(\mathbf{w}^T\mathbf{x}) \overset{(\star)}{=} \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$$

$$y(\mathbf{x}): \ \mathbf{R}^{p+1} \rightarrow (0;1)$$

Logistic Regression

Interpretation of the Logistic Model Function

$y(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$ is interpreted as the estimated probability for the event $C{=}1$:

- $y(\mathbf{x}) = P(C{=}1 \mid \mathbf{X}{=}\mathbf{x}; \mathbf{w}) =: p(1 \mid \mathbf{x}; \mathbf{w})$, "Probability for $C{=}1$ given $\mathbf{x}$, parameterized by $\mathbf{w}$."
- $1 - y(\mathbf{x}) = P(C{=}0 \mid \mathbf{X}{=}\mathbf{x}; \mathbf{w}) =: p(0 \mid \mathbf{x}; \mathbf{w})$, "Probability for $C{=}0$ given $\mathbf{x}$, parameterized by $\mathbf{w}$."

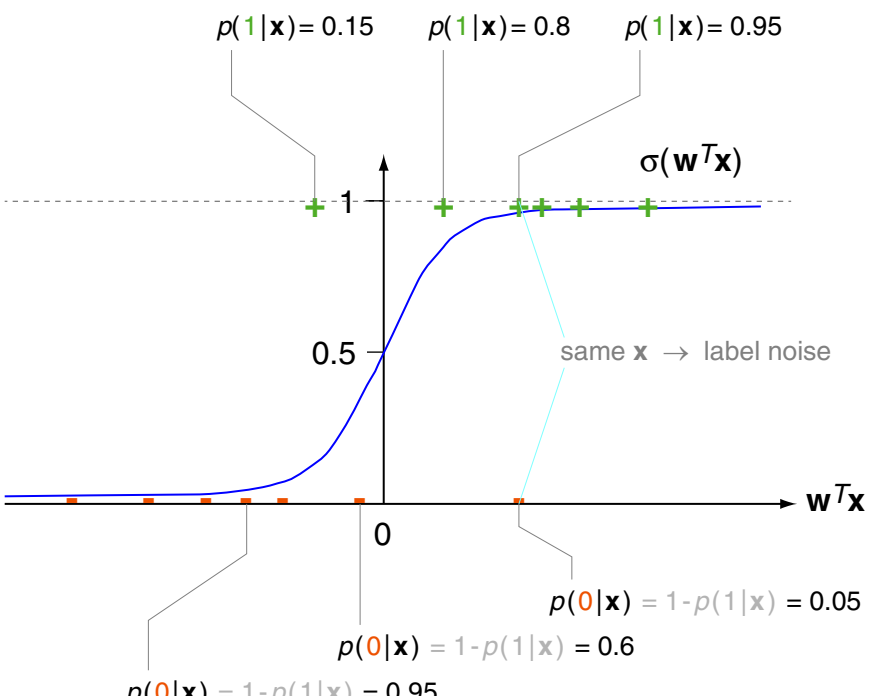
Logistic Regression

Interpretation of the Logistic Model Function (continued)

$y(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$ is interpreted as the estimated probability for the event $C{=}1$:

- $y(\mathbf{x}) = P(C{=}1 \mid \mathbf{X}{=}\mathbf{x}; \mathbf{w}) =: p(1 \mid \mathbf{x}; \mathbf{w})$, "Probability for $C{=}1$ given $\mathbf{x}$, parameterized by $\mathbf{w}$."
- $1 - y(\mathbf{x}) = P(C{=}0 \mid \mathbf{X}{=}\mathbf{x}; \mathbf{w}) =: p(0 \mid \mathbf{x}; \mathbf{w})$, "Probability for $C{=}0$ given $\mathbf{x}$, parameterized by $\mathbf{w}$."

Observations D, given as "+" and "-", along with the probability values as specified by $\sigma(\mathbf{w}^T\mathbf{x})$:



Logistic Regression

Interpretation of the Logistic Model Function (continued)

$y(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$ is interpreted as the estimated probability for the event $C{=}1$:

- $y(\mathbf{x}) = P(C{=}1 \mid \mathbf{X}{=}\mathbf{x}; \mathbf{w}) =: p(1 \mid \mathbf{x}; \mathbf{w})$, "Probability for $C{=}1$ given $\mathbf{x}$, parameterized by $\mathbf{w}$."
- $1 - y(\mathbf{x}) = P(C{=}0 \mid \mathbf{X}{=}\mathbf{x}; \mathbf{w}) =: p(0 \mid \mathbf{x}; \mathbf{w})$, "Probability for $C{=}0$ given $\mathbf{x}$, parameterized by $\mathbf{w}$."

Example (email spam classification):

$$\mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} 1 \\ |\text{obscene words}| \end{pmatrix}, \qquad \mathbf{x}_1 = \begin{pmatrix} 1 \\ 5 \end{pmatrix} \quad \text{and} \quad y(\mathbf{x}_1) = 0.67$$

$\rightsquigarrow$ 67% chance that this email is spam.

Logistic Regression

Interpretation of the Logistic Model Function (continued)

$y(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$ is interpreted as the estimated probability for the event $C{=}1$:

- $y(\mathbf{x}) = P(C{=}1 \mid \mathbf{X}{=}\mathbf{x}; \mathbf{w}) =: p(1 \mid \mathbf{x}; \mathbf{w})$, "Probability for $C{=}1$ given $\mathbf{x}$, parameterized by $\mathbf{w}$."
- $1 - y(\mathbf{x}) = P(C{=}0 \mid \mathbf{X}{=}\mathbf{x}; \mathbf{w}) =: p(0 \mid \mathbf{x}; \mathbf{w})$, "Probability for $C{=}0$ given $\mathbf{x}$, parameterized by $\mathbf{w}$."

Example (email spam classification):

$$\mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} 1 \\ |\text{obscene words}| \end{pmatrix}, \qquad \mathbf{x}_1 = \begin{pmatrix} 1 \\ 5 \end{pmatrix} \quad \text{and} \quad y(\mathbf{x}_1) = 0.67$$

$\rightsquigarrow$ 67% chance that this email is spam.

Logistic Regression

Interpretation of the Logistic Model Function (continued)

$y(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$ is interpreted as the estimated probability for the event $C{=}1$:

- $y(\mathbf{x}) = P(C{=}1 \mid \mathbf{X}{=}\mathbf{x}; \mathbf{w}) =: p(1 \mid \mathbf{x}; \mathbf{w})$, "Probability for $C{=}1$ given $\mathbf{x}$, parameterized by $\mathbf{w}$."
- $1 - y(\mathbf{x}) = P(C{=}0 \mid \mathbf{X}{=}\mathbf{x}; \mathbf{w}) =: p(0 \mid \mathbf{x}; \mathbf{w})$, "Probability for $C{=}0$ given $\mathbf{x}$, parameterized by $\mathbf{w}$."

Example (email spam classification):

$$\mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} 1 \\ |\text{obscene words}| \end{pmatrix}, \qquad \mathbf{x}_1 = \begin{pmatrix} 1 \\ 5 \end{pmatrix} \quad \text{and} \quad y(\mathbf{x}_1) = 0.67$$

$\rightsquigarrow$ 67% chance that this email is spam.

Classification: Predict $\begin{cases} 1, & \text{if } \sigma(\mathbf{w}^T\mathbf{x}) \geq 0.5 \\ 0, & \text{if } \sigma(\mathbf{w}^T\mathbf{x}) < 0.5 \end{cases}$

Logistic Regression

Interpretation of the Logistic Model Function (continued)

$y(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$ is interpreted as the estimated probability for the event $C{=}1$:

- $y(\mathbf{x}) = P(C{=}1 \mid \mathbf{X}{=}\mathbf{x}; \mathbf{w}) =: p(1 \mid \mathbf{x}; \mathbf{w})$, "Probability for $C{=}1$ given $\mathbf{x}$, parameterized by $\mathbf{w}$."
- $1 - y(\mathbf{x}) = P(C{=}0 \mid \mathbf{X}{=}\mathbf{x}; \mathbf{w}) =: p(0 \mid \mathbf{x}; \mathbf{w})$, "Probability for $C{=}0$ given $\mathbf{x}$, parameterized by $\mathbf{w}$."

Example (email spam classification):

$$\mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} 1 \\ |\text{obscene words}| \end{pmatrix}, \qquad \mathbf{x}_1 = \begin{pmatrix} 1 \\ 5 \end{pmatrix} \quad \text{and} \quad y(\mathbf{x}_1) = 0.67$$

$\rightsquigarrow$ 67% chance that this email is spam.

Classification: Predict $\begin{cases} 1, \text{ if } \sigma(\mathbf{w}^T\mathbf{x}) \geq 0.5 & \Leftrightarrow \ \mathbf{w}^T\mathbf{x} \geq 0 \\ 0, \text{ if } \sigma(\mathbf{w}^T\mathbf{x}) < 0.5 & \Leftrightarrow \ \mathbf{w}^T\mathbf{x} < 0 \end{cases}$

Logistic Regression

Interpretation of the Logistic Model Function (continued)

$y(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$ is interpreted as the estimated probability for the event $C{=}1$:

- $y(\mathbf{x}) = P(C{=}1 \mid \mathbf{X}{=}\mathbf{x}; \mathbf{w}) =: p(1 \mid \mathbf{x}; \mathbf{w})$, "Probability for $C{=}1$ given $\mathbf{x}$, parameterized by $\mathbf{w}$."
- $1 - y(\mathbf{x}) = P(C{=}0 \mid \mathbf{X}{=}\mathbf{x}; \mathbf{w}) =: p(0 \mid \mathbf{x}; \mathbf{w})$, "Probability for $C{=}0$ given $\mathbf{x}$, parameterized by $\mathbf{w}$."

Estimate optimum $\mathbf{w}$ by maximizing the probability, $p(D; \mathbf{w})$:

$$\mathbf{w}_{\text{ML}} = \underset{\mathbf{w}\in\mathbf{R}^{p+1}}{\text{argmax}}\, p(D; \mathbf{w})$$

$\Leftrightarrow$ Estimate optimum $\mathbf{w}$ by minimizing the logistic loss, $L_\sigma(\mathbf{w})$:

$$\mathbf{w}_{\text{ML}} = \underset{\mathbf{w}\in\mathbf{R}^{p+1}}{\text{argmin}}\, L_\sigma(\mathbf{w})$$

[RSS minimization]

Remarks (probabilistic view to classification) :

- If $y(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$ is interpreted as "probability for $C{=}1$ given feature vector $\mathbf{x}$", then $\mathbf{w}$ is the (unique) characterizing parameter vector of the hidden stochastic process that generates the observed data D.

  Recap. As a consequence, $\mathbf{w}$ is not the realization of a random variable—which would come along with a distribution—but an *exogenous parameter*, which is varied in order to find the maximum probability $p(D; \mathbf{w})$ or the minimum loss $L_\sigma(\mathbf{w})$.

  The fact that $\mathbf{w}$ is an exogenous parameter and not a the realization of a random variable is reflected by the notation, which uses a »;« instead of a »|« in $p(\,\cdot\,)$.

- The underlying probability space—which can be left implicit—looks as follows:

  The sample space Ω corresponds to a set O of real-world objects, P is a probability measure defined on $\mathcal{P}(\Omega)$. The classification task (experiment) suggests two types of random variables, $C : \Omega \to \mathbf{X}$ and $C : \Omega \to \{0, 1\}$.

  See section Evaluating Effectiveness of part Machine Learning Basics for an illustration of the probabilistic view to classification, and section Probability Basics of part Bayesian Learning for a recap of concepts from probability theory.

- **X** and *C* denote (multivariate) random variables with ranges $\mathbf{X}$ and C respectively.

  X corresponds to a model formation function α that returns for a real-world object $o \in O$ its feature vector $\mathbf{x}$, $\mathbf{x} = \alpha(o)$, and *C* corresponds to an ideal classifier γ that returns its class c, $c = \gamma(o)$.

Remarks (probabilistic view to classification) : (continued)

- The interpretation of $y(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$ as probability for the event $C{=}1$ is not a mathematical consequence but a decision of the modeler. This decision is based on the advantageous properties of the sigmoid function, on practical considerations, and on heuristic simplifications of the real world.

  Consider the following two aspects where an interpretation of $\sigma(\mathbf{w}^T\mathbf{x})$ is questionable:

  1. The sigmoid function implies that with $\mathbf{w}^T\mathbf{x} \to +\infty$ we get $y(\mathbf{x}) \to 1$ or $P(C{=}1) \to 1$. Likewise, with $\mathbf{w}^T\mathbf{x} \to -\infty$ we get $P(C{=}1) \to 0$. Though such a strict monotonicity appears self-evident, it need not necessarily correspond to the observed behavior in a real world experiment.

  2. The sigmoid function implies a smooth, virtually linear transition from low probability values (around 0.1) to high probability values (around 0.9) as its argument $\mathbf{w}^T\mathbf{x}$ increases.

     This link between the continuous growth of $\mathbf{w}^T\mathbf{x}$ and the continuous growth of probability values $P(C{=}1)$ presumes a proportional connection between cause (in the form of $\mathbf{X}{=}\mathbf{x}$) and effect (in the form of $C{=}1$). Again, such a relation appears sensible but may not necessarily model the real world.

Remarks (derivation of $L_\sigma(\mathbf{w})$):

- The most probable (= optimum) hypothesis in the space H of possible hypotheses, h_{ML}, can be estimated with the maximum likelihood principle: $h_{\mathsf{ML}} = \underset{h \in H}{\operatorname{argmax}}\, p(D; h)$.
- Applied to logistic regression: $\mathbf{w}_{\mathsf{ML}} = \underset{\mathbf{w} \in \mathbf{R}^{p+1}}{\operatorname{argmax}}\, p(D; \mathbf{w})$, where

$$
\begin{aligned}
\underset{\mathbf{w} \in \mathbf{R}^{p+1}}{\operatorname{argmax}}\, p(D; \mathbf{w}) &= \underset{\mathbf{w} \in \mathbf{R}^{p+1}}{\operatorname{argmax}} \prod_{(\mathbf{x},c) \in D} p(\mathbf{x}, c; \mathbf{w}) = \underset{\mathbf{w} \in \mathbf{R}^{p+1}}{\operatorname{argmax}} \prod_{(\mathbf{x},c) \in D} (p(\mathbf{x}) \cdot p(c \mid \mathbf{x}; \mathbf{w})) \\
&= \underset{\mathbf{w} \in \mathbf{R}^{p+1}}{\operatorname{argmax}} \prod_{(\mathbf{x},c) \in D} p(\mathbf{x}) \cdot \prod_{(\mathbf{x},c) \in D} p(c \mid \mathbf{x}; \mathbf{w}) \overset{(1)}{=} \underset{\mathbf{w} \in \mathbf{R}^{p+1}}{\operatorname{argmax}} \prod_{(\mathbf{x},c) \in D} p(c \mid \mathbf{x}; \mathbf{w}) \\
&= \underset{\mathbf{w} \in \mathbf{R}^{p+1}}{\operatorname{argmax}} \prod_{(\mathbf{x},1) \in D} \sigma(\mathbf{w}^T \mathbf{x}) \cdot \prod_{(\mathbf{x},0) \in D} (1 - \sigma(\mathbf{w}^T \mathbf{x})) \\
&= \underset{\mathbf{w} \in \mathbf{R}^{p+1}}{\operatorname{argmax}} \prod_{(\mathbf{x},1) \in D} y(\mathbf{x}) \cdot \prod_{(\mathbf{x},0) \in D} (1 - y(\mathbf{x})) \\
&\overset{(2)}{=} \underset{\mathbf{w} \in \mathbf{R}^{p+1}}{\operatorname{argmax}} \log \prod_{(\mathbf{x},1) \in D} y(\mathbf{x}) + \log \prod_{(\mathbf{x},0) \in D} (1 - y(\mathbf{x})) \\
&= \underset{\mathbf{w} \in \mathbf{R}^{p+1}}{\operatorname{argmax}} \sum_{(\mathbf{x},1) \in D} \log y(\mathbf{x}) + \sum_{(\mathbf{x},0) \in D} \log(1 - y(\mathbf{x})) \\
&= \hookrightarrow \text{ p. 24}
\end{aligned}
$$

Remarks (derivation of $L_\sigma(\mathbf{w})$) : (continued)

$$= \operatorname*{argmax}_{\mathbf{w}\in\mathbf{R}^{p+1}} \sum_{(\mathbf{x},c)\in D} c\cdot\log y(\mathbf{x}) \;+\; (1-c)\cdot\log(1-y(\mathbf{x}))$$

$$\overset{(3)}{=} \operatorname*{argmin}_{\mathbf{w}\in\mathbf{R}^{p+1}} -\sum_{(\mathbf{x},c)\in D} c\cdot\log y(\mathbf{x}) \;+\; (1-c)\cdot\log(1-y(\mathbf{x}))$$

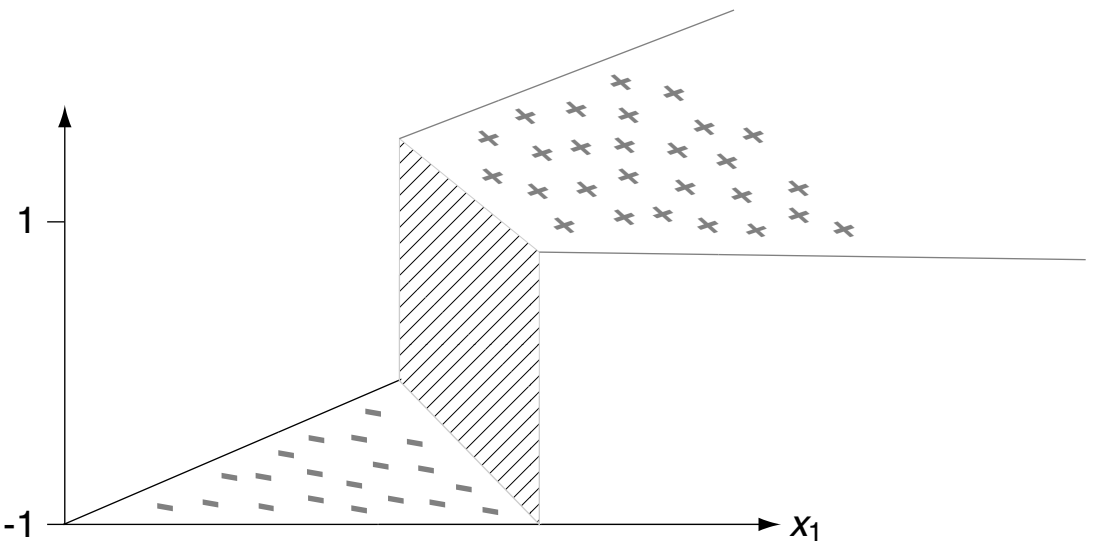
$$\overset{(4)}{=} \operatorname*{argmin}_{\mathbf{w}\in\mathbf{R}^{p+1}} \sum_{(\mathbf{x},c)\in D} \overbrace{-c\cdot\log(y(\mathbf{x})) \;-\; (1-c)\cdot\log(1-y(\mathbf{x}))}^{l_\sigma :=}$$

$$=: \operatorname*{argmin}_{\mathbf{w}\in\mathbf{R}^{p+1}} \sum_{(\mathbf{x},c)\in D} l_\sigma(c,y(\mathbf{x})) \;=\; \operatorname*{argmin}_{\mathbf{w}\in\mathbf{R}^{p+1}} L_\sigma(\mathbf{w})$$

- Hints:
  - (1) $\prod_{(\mathbf{x},c)\in D} p(\mathbf{x})$ is constant with respect to the variation of $\mathbf{w}$.
  - (2) $\operatorname{argmax}_x f(x) = \operatorname{argmax}_x g\circ f(x)$ (similarly for argmin) if $g(z)$ is a strictly monotonically increasing function. Here, $\log(z)$ is in the role of $g(z)$. Conversely, if $g(z)$ is a strictly monotonically decreasing function, then $\operatorname{argmax}_x f(x) = \operatorname{argmin}_x g\circ f(x)$.
  - (3) The maximization problem (the argmax-expression) can be reformulated as minimization problem, i.e., as an argmin-expression. See in this regard the second part of Hint (1).
  - (4) The argument of the argmin-expression quantifies $L_\sigma(\mathbf{w})$, the global logistic loss related to some $\mathbf{w}$, and, analogously, the pointwise logistic loss, $l_\sigma(c,y(\mathbf{x}))$.

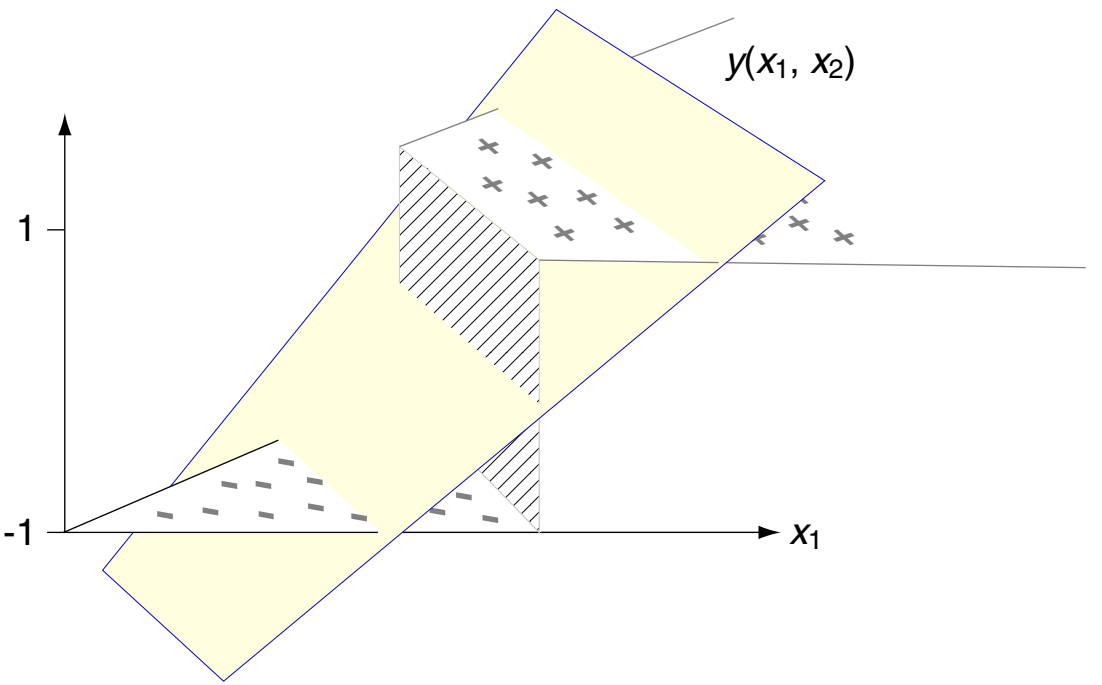
Logistic Regression

Recap. Linear Regression for Classification (illustrated for $p = 2$)



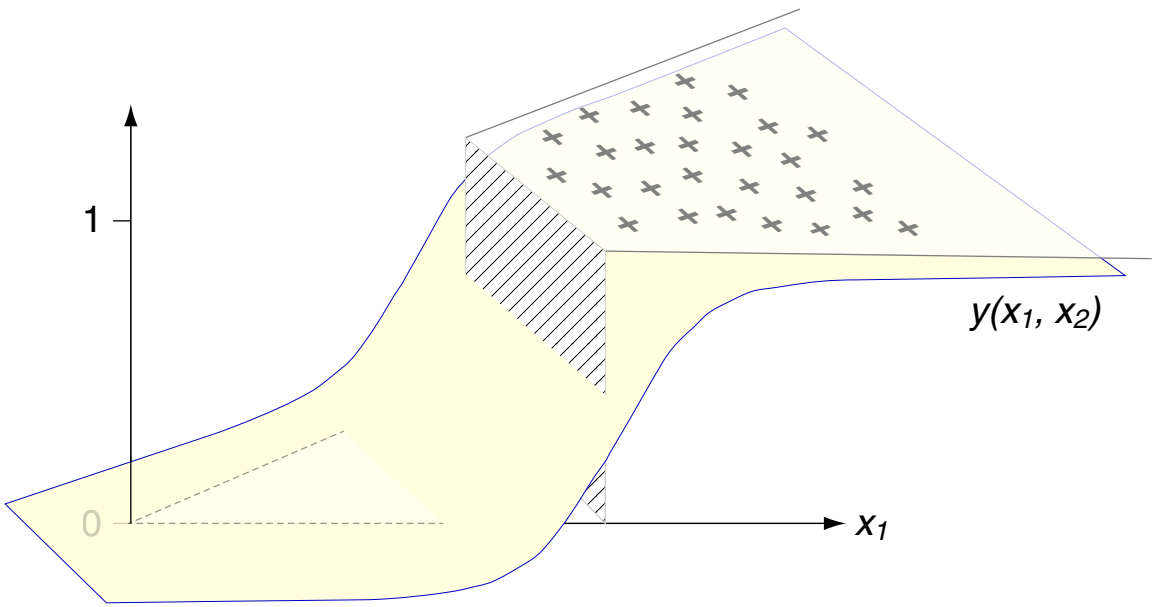
Logistic Regression

Recap. Linear Regression for Classification (illustrated for $p = 2$) (continued)



Logistic Regression

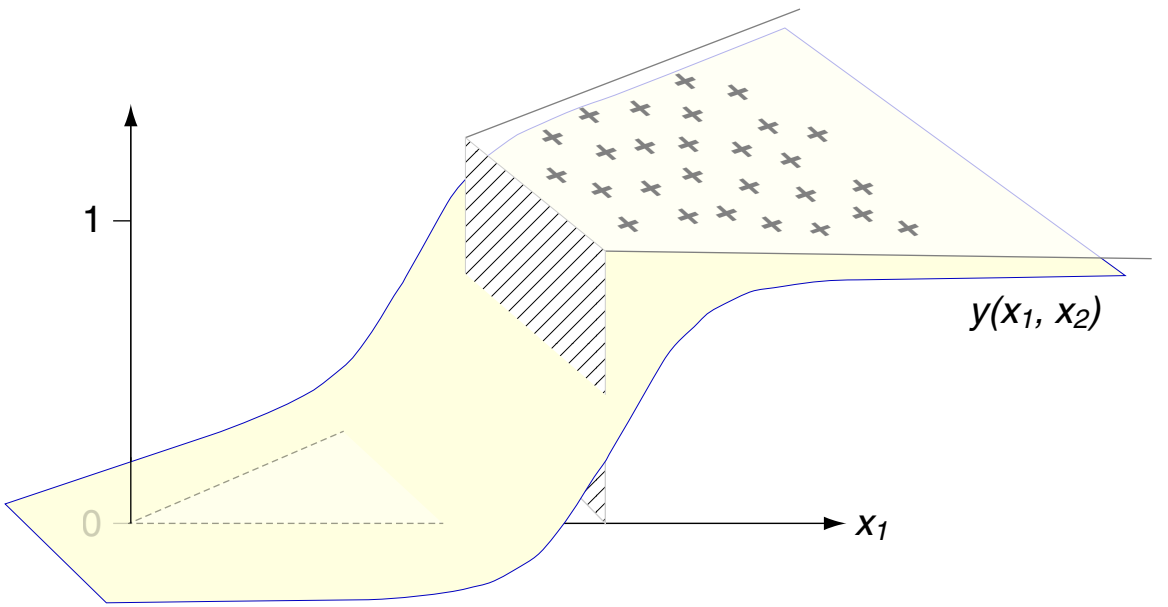
Logistic Regression for Classification (illustrated for $p = 2$)



Logistic Regression

Logistic Regression for Classification (illustrated for $p = 2$) (continued)

Use logistic regression to learn $\mathbf{w}$ from D, where $y(\mathbf{x}) = \dfrac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$.

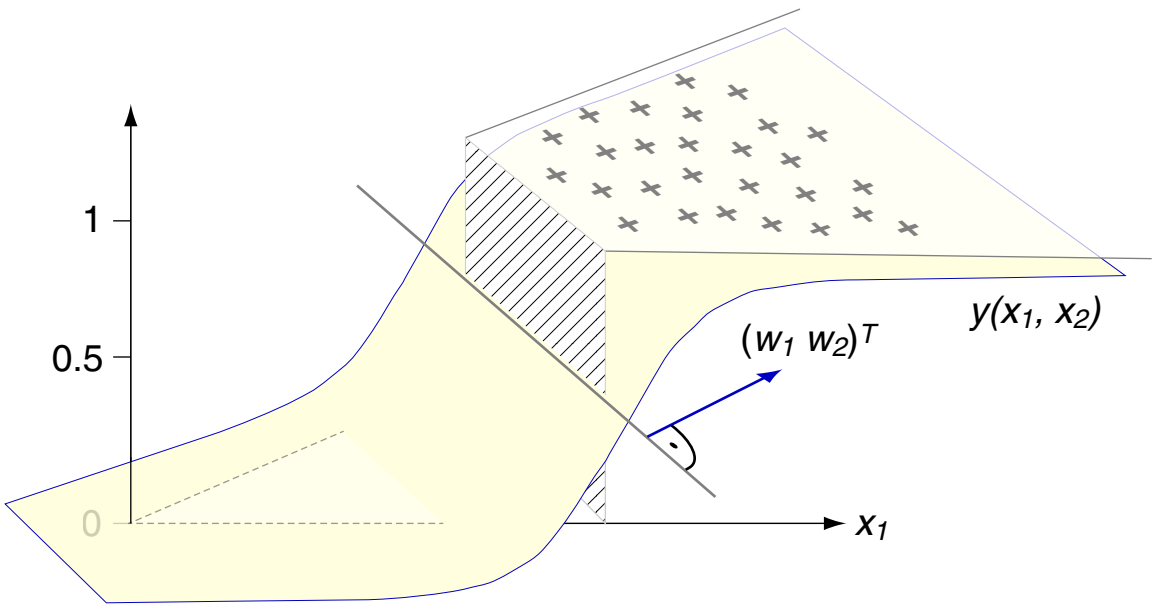


$$y(x_1, x_2) = \frac{1}{1 + e^{-(w_0 + w_1 \cdot x_1 + w_2 \cdot x_2)}}$$

Logistic Regression

Logistic Regression for Classification (illustrated for $p = 2$) (continued)

Use logistic regression to learn $\mathbf{w}$ from D, where $y(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$.

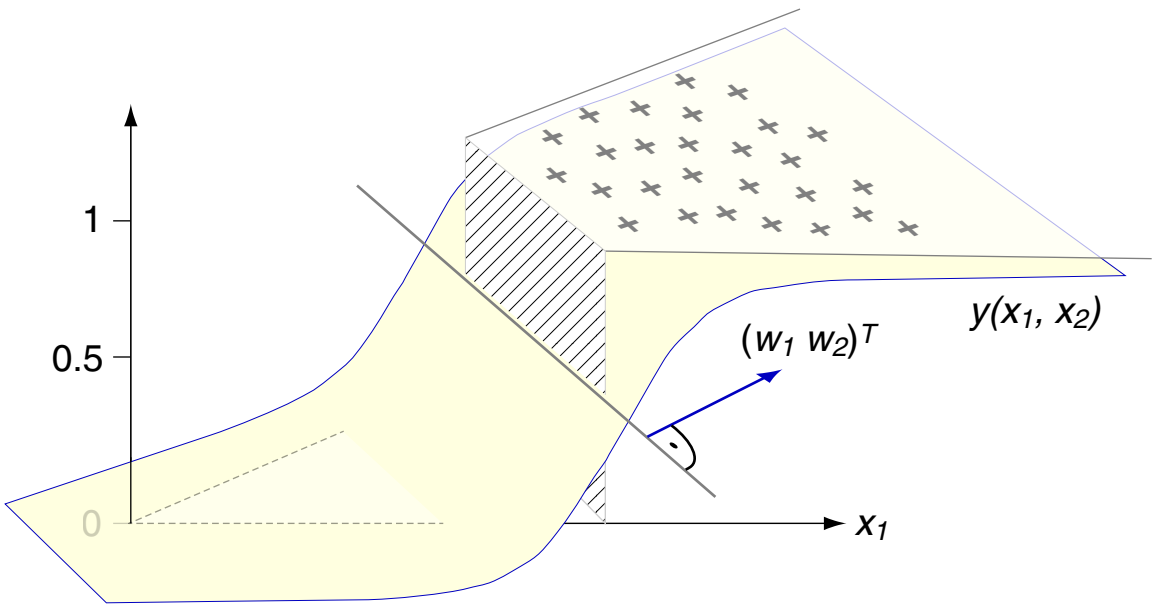


$$y(x_1, x_2) = \frac{1}{1 + e^{-(w_0 + w_1 \cdot x_1 + w_2 \cdot x_2)}}$$

Logistic Regression

Logistic Regression for Classification (illustrated for $p = 2$) (continued)

Use logistic regression to learn $\mathbf{w}$ from D, where $y(\mathbf{x}) = \dfrac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$.

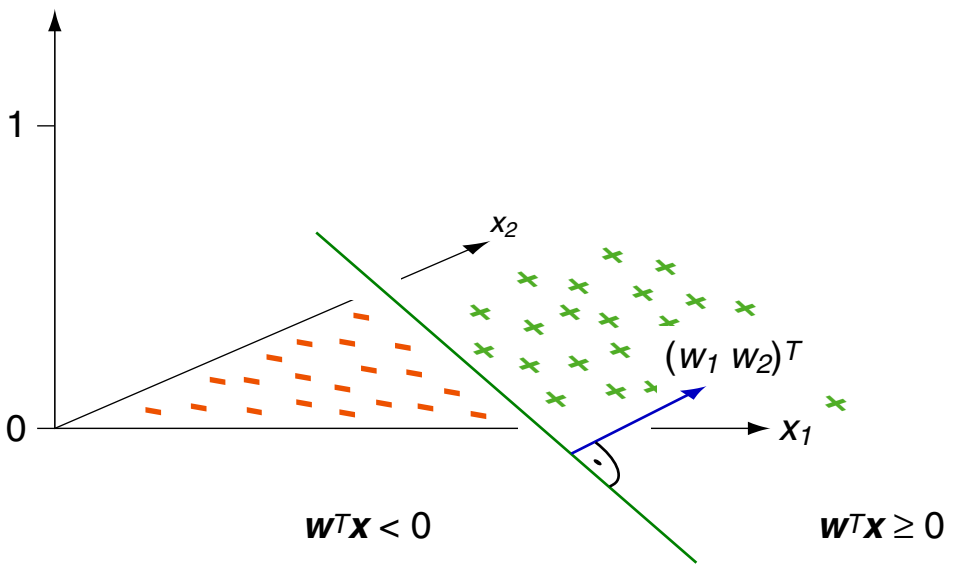


Classification: Predict $\begin{cases} 1, & \text{if } \sigma(\mathbf{w}^T\mathbf{x}) \geq 0.5 \\ 0, & \text{if } \sigma(\mathbf{w}^T\mathbf{x}) < 0.5 \end{cases}$

Logistic Regression

Logistic Regression for Classification (illustrated for $p = 2$) (continued)

Use logistic regression to learn $\mathbf{w}$ from D, where $y(\mathbf{x}) = \dfrac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$.

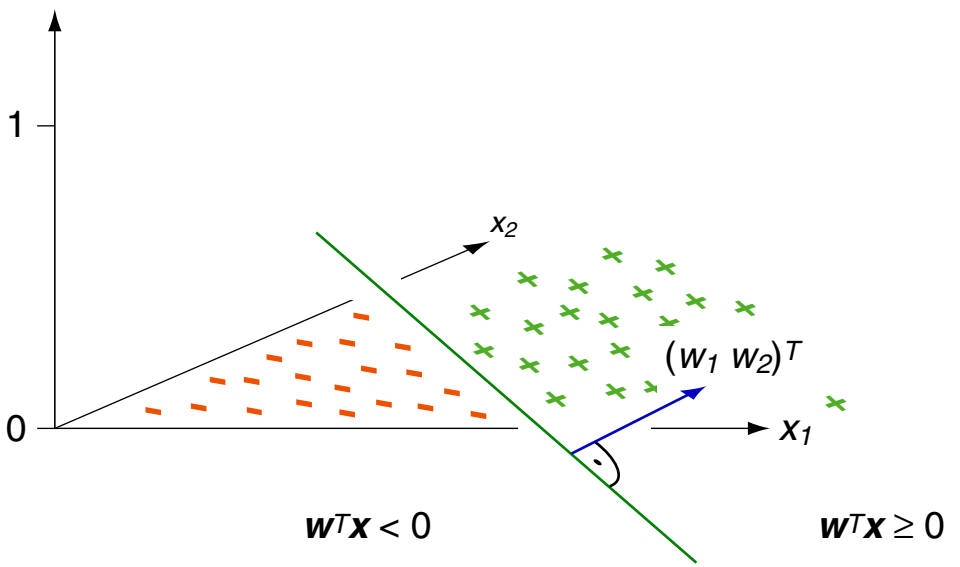


Classification: Predict $\begin{cases} 1, & \text{if } \sigma(\mathbf{w}^T\mathbf{x}) \geq 0.5 \\ 0, & \text{if } \sigma(\mathbf{w}^T\mathbf{x}) < 0.5 \end{cases}$

Logistic Regression

Logistic Regression for Classification (illustrated for $p = 2$) (continued)

Use logistic regression to learn $\mathbf{w}$ from D, where $y(\mathbf{x}) = \dfrac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$.



Classification: Predict $\begin{cases} 1, \text{ if } \sigma(\mathbf{w}^T\mathbf{x}) \geq 0.5 \quad \Leftrightarrow \ \mathbf{w}^T\mathbf{x} \geq 0 \\ 0, \text{ if } \sigma(\mathbf{w}^T\mathbf{x}) < 0.5 \quad \Leftrightarrow \ \mathbf{w}^T\mathbf{x} < 0 \end{cases}$

Logistic Regression

The BGD$_\sigma$ Algorithm

[algorithms: LMS BGD$_\sigma$ PT BGD IGD]

Algorithm:	BGD$_\sigma$	Batch Gradient Descent.
Input:	D	Multiset of examples $(\mathbf{x}, c)$ with $\mathbf{x} \in \mathbf{R}^p$, $c \in \{0, 1\}$.
	η	Learning rate, a small positive constant.
Output:	$\mathbf{w}$	Weight vector from $\mathbf{R}^{p+1}$. (= hypothesis)

BGD$_\sigma(D, \eta)$

1. *initialize_random_weights*$(\mathbf{w})$, $t = 0$
2. **REPEAT**
3. $t = t + 1$, $\Delta\mathbf{w} = 0$
4. FOREACH $(\mathbf{x}, c) \in D$ DO
5. $y(\mathbf{x}) \overset{(\star)}{=} \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$
6. $\delta = c - y(\mathbf{x})$
7. $\Delta\mathbf{w} \overset{(\star)}{=} \Delta\mathbf{w} + \eta \cdot \delta \cdot \mathbf{x}$ // $-\delta \cdot \mathbf{x}$ is the derivative of $l_\sigma(c, y(\mathbf{x}))$ wrt. $\mathbf{w}$.
8. ENDDO
9. $\mathbf{w} = \mathbf{w} + \Delta\mathbf{w}$
10. **UNTIL**(*convergence*$(D, y(\,\cdot\,), t)$)
11. *return*$(\mathbf{w})$

Logistic Regression

The BGD$_\sigma$ Algorithm

[algorithms: LMS BGD$_\sigma$ PT BGD IGD]

Algorithm:	BGD$_\sigma$	Batch Gradient Descent.
Input:	D	Multiset of examples $(\mathbf{x}, c)$ with $\mathbf{x} \in \mathbf{R}^p$, $c \in \{0, 1\}$.
	η	Learning rate, a small positive constant.
Output:	$\mathbf{w}$	Weight vector from $\mathbf{R}^{p+1}$. (= hypothesis)

BGD$_\sigma(D, \eta)$

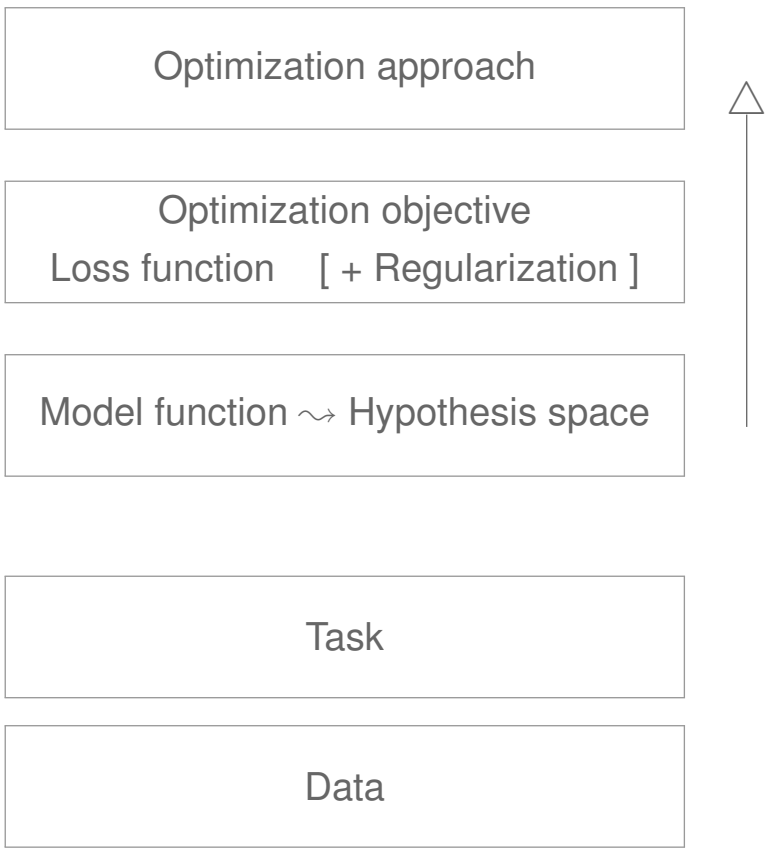
1. *initialize_random_weights*$(\mathbf{w})$, $t = 0$
2. **REPEAT**
3. $t = t + 1$, $\Delta\mathbf{w} = 0$
4. FOREACH $(\mathbf{x}, c) \in D$ DO
5. $y(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$ Model function evaluation.
6. $\delta = c - y(\mathbf{x})$ Calculation of residual.
7. $\Delta\mathbf{w} = \Delta\mathbf{w} + \eta \cdot \delta \cdot \mathbf{x}$ Calculation of derivative, accumulate for entire D.
8. ENDDO
9. $\mathbf{w} = \mathbf{w} + \Delta\mathbf{w}$ Parameter vector update = one gradient step down.
10. **UNTIL**(*convergence*$(D, y(\cdot), t)$)
11. *return*$(\mathbf{w})$

Remarks:

- The BGD$_\sigma$ Algorithm is an iterative method to estimate $\mathbf{w}_{\text{ML}}$ in the model function $y(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$. There is no direct method (such as the normal equations in linear regression) to tackle the optimization problem.
- The BGD$_\sigma$ Algorithm exploits the derivative of the pointwise logistic loss $l_\sigma(c, y(\mathbf{x}))$ with respect to $\mathbf{w}$, which is $\delta \cdot \mathbf{x} = (c - y(\mathbf{x})) \cdot \mathbf{x} = (c - \sigma(\mathbf{w}^T\mathbf{x})) \cdot \mathbf{x}$. The derivation of this term, as well as notes regarding the speed of convergence of the basic gradient descent are given in section Gradient Descent in Detail of part Linear Models.
- Each BGD$_\sigma$ iteration "`REPEAT ... UNTIL`"
  1. computes the direction of steepest loss descent as $-\nabla L_\sigma(\mathbf{w}_t) = \sum_{(\mathbf{x},c)\in D} (c - y_t(\mathbf{x})) \cdot \mathbf{x}$, and
  2. updates $\mathbf{w}_t$ by taking a step of length η in this direction.
- Recap. The function *convergence*$(\,\cdot\,)$ can analyze the global logistic loss, $L_\sigma(\mathbf{w}_t)$, or the norm of the loss gradient, $||\nabla L_\sigma(\mathbf{w}_t)||$, and compare it to a small positive bound ε. Consider in this regard the vectors of observed and computed classes, $D|_c$ and $y(D|_\mathbf{x})$ respectively. In addition, the function may check via t an upper bound on the number of iterations.

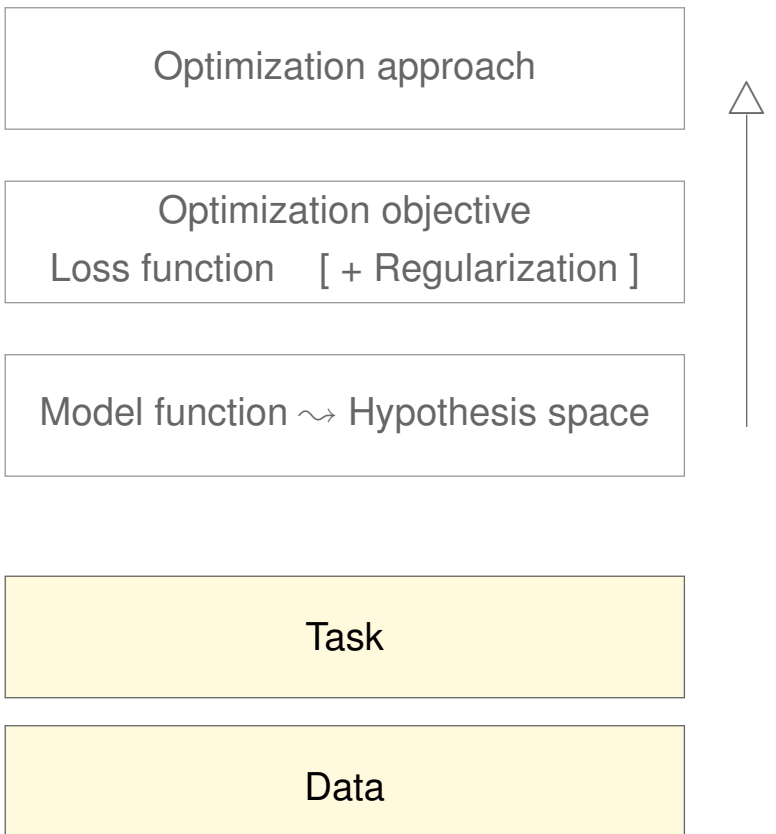
Logistic Regression

Machine Learning Stack for Logistic Regression



Logistic Regression

Machine Learning Stack for Logistic Regression (continued)



Binary classification

$$D = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\} \subseteq X \times \{0, 1\}$$

Logistic Regression

Machine Learning Stack for Logistic Regression (continued)

Optimization approach

Optimization objective
Loss function [+ Regularization]

Model function $\rightsquigarrow$ Hypothesis space

- Hypothesis space: $\mathbf{w} \in \mathbf{R}^{p+1}$
- Logistic model: $y(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$

Task

Binary classification

Data

$D = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\} \subseteq X \times \{0, 1\}$

Logistic Regression

Machine Learning Stack for Logistic Regression (continued)

Optimization approach

Optimization objective
Loss function [+ Regularization]

Model function $\rightsquigarrow$ Hypothesis space

Task

Data

- Objective: minimize logistic loss $L_\sigma(\mathbf{w})$
- Regularization: none
- Loss: $l_\sigma(c,y(\mathbf{x})) = -c\cdot\log(y(\mathbf{x}))-(1-c)\cdot\log(1-y(\mathbf{x}))$

- Hypothesis space: $\mathbf{w} \in \mathbf{R}^{p+1}$
- Logistic model: $y(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$

Binary classification

$D = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\} \subseteq X \times \{0, 1\}$

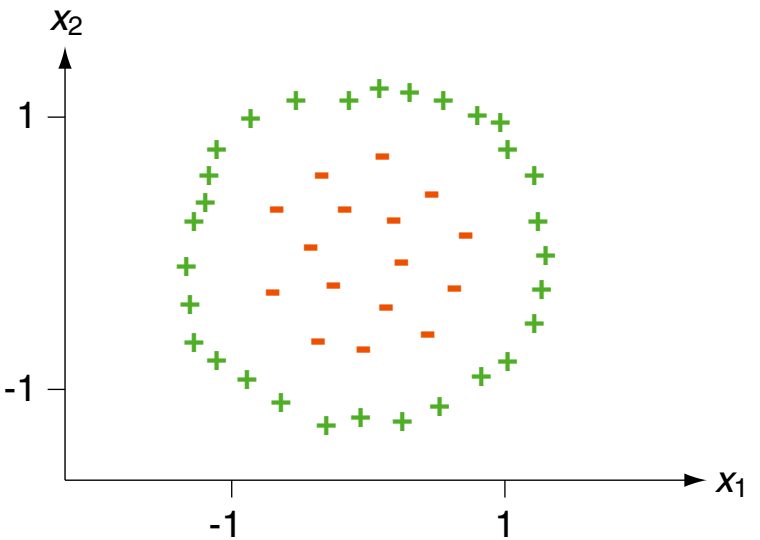
Logistic Regression

Machine Learning Stack for Logistic Regression (continued)

Optimization approach	BGD, Newton-Raphson, BFGS, Conjugate GD
Optimization objective Loss function [+ Regularization]	❑ Objective: minimize logistic loss $L_\sigma(\mathbf{w})$ ❑ Regularization: none ❑ Loss: $l_\sigma(c,y(\mathbf{x})) = -c\cdot\log(y(\mathbf{x}))-(1-c)\cdot\log(1-y(\mathbf{x}))$
Model function $\rightsquigarrow$ Hypothesis space	❑ Hypothesis space: $\mathbf{w} \in \mathbf{R}^{p+1}$ ❑ Logistic model: $y(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$
Task	Binary classification
Data	$D = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\} \subseteq X \times \{0, 1\}$

Logistic Regression

Non-Linear Decision Boundaries [linear regression]

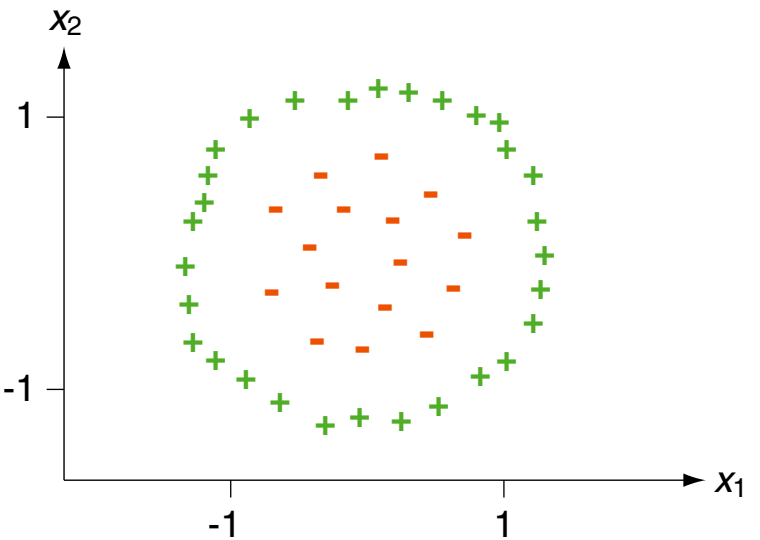


Higher order polynomial terms in the features (linear in the parameters):

$$y(\mathbf{x}) = \sigma(w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_1^2 + w_4 \cdot x_2^2)$$

Logistic Regression

Non-Linear Decision Boundaries (continued) [linear regression]

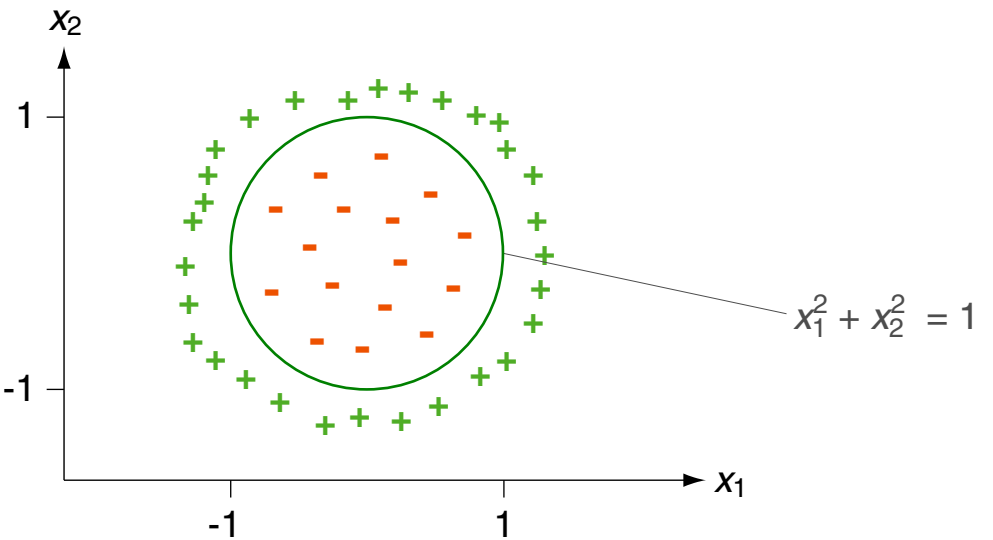


Higher order polynomial terms in the features (linear in the parameters):

$$y(\mathbf{x}) = \sigma(w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_1^2 + w_4 \cdot x_2^2)$$

Logistic Regression

Non-Linear Decision Boundaries (continued) [linear regression]



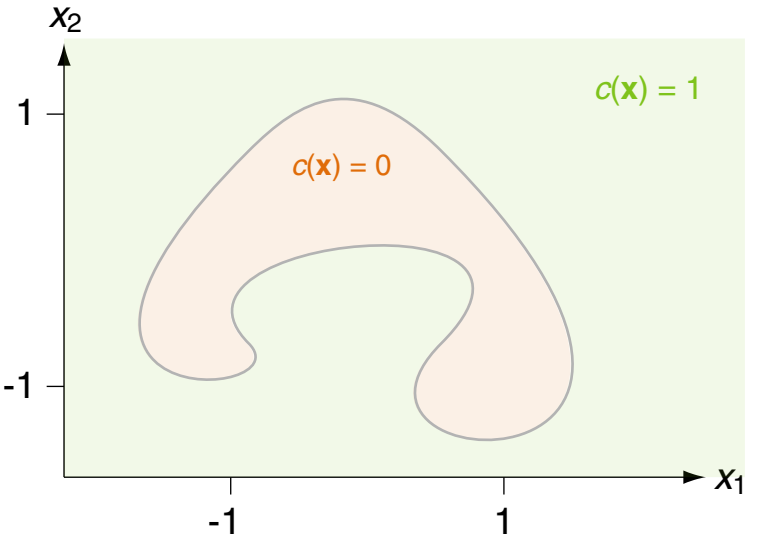
Higher order polynomial terms in the features (linear in the parameters):

$$y(\mathbf{x}) = \sigma(w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_1^2 + w_4 \cdot x_2^2)$$

with $\mathbf{w} = \begin{pmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$ $\rightsquigarrow$ $y(\mathbf{x}) = \dfrac{1}{1 + e^{-(-1 + x_1^2 + x_2^2)}}$

Logistic Regression

Non-Linear Decision Boundaries (continued) [linear regression]



Higher order polynomial terms in the features (linear in the parameters):

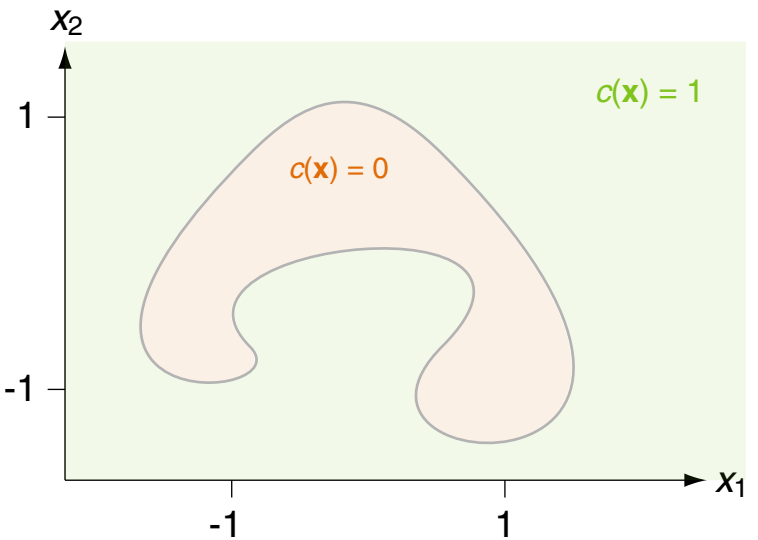
$$y(\mathbf{x}) = \sigma(w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_1^2 + w_4 \cdot x_2^2)$$

with $\mathbf{w} = \begin{pmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$ $\rightsquigarrow$ $y(\mathbf{x}) = \dfrac{1}{1 + e^{-(-1 + x_1^2 + x_2^2)}}$

Classification: Predict $\begin{cases} 1, \text{ if } x_1^2 + x_2^2 \geq 1 & \Leftrightarrow \mathbf{w}^T\mathbf{x} \geq 0 \\ 0, \text{ if } x_1^2 + x_2^2 < 1 & \Leftrightarrow \mathbf{w}^T\mathbf{x} < 0 \end{cases}$

Logistic Regression

Non-Linear Decision Boundaries (continued) [linear regression]



More complex polynomials entail more complex decision boundaries:

$$y(\mathbf{x}) = \sigma(w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_1^2 + w_4 \cdot x_1^2 \cdot x_2 + w_5 \cdot x_1^2 \cdot x_2^2 + \ldots)$$

Remarks:

- Under logistic regression the structure of a hypothesis, i.e., the forms of possible decision boundaries, is identical to the hypothesis structure under linear regression. Similarly, the respective hypothesis spaces are the same. Hence, the expressiveness, i.e., the complexity of classification problems that can be tackled (or, the effectiveness at which classification problems can be decided) is identical for the two regression approaches.
- Linear regression and logistic regression differ in the way how the model function parameters, $\mathbf{w}$, are determined. In both cases the optimum $\mathbf{w}$ is the result of a loss minimization problem.

  Recall that "loss" means "interpretation of residuals." Linear regression and logistic regression differ with respect to this interpretation: While the former simply squares the residuals, this way putting a high weight onto outliers, the latter models an increasing confidence in class membership probability with increasing hyperplane distance. This different interpretation will usually lead to a different parameter vector $\mathbf{w}$, i.e., a different hyperplane.
- Note that the term "hypothesis" is sometimes used to refer to the model function $y(\mathbf{x})$, the other time it refers to the *parameters* $\mathbf{w}$ of a model function. However, this is more of a convention that makes no difference to the overall argument.

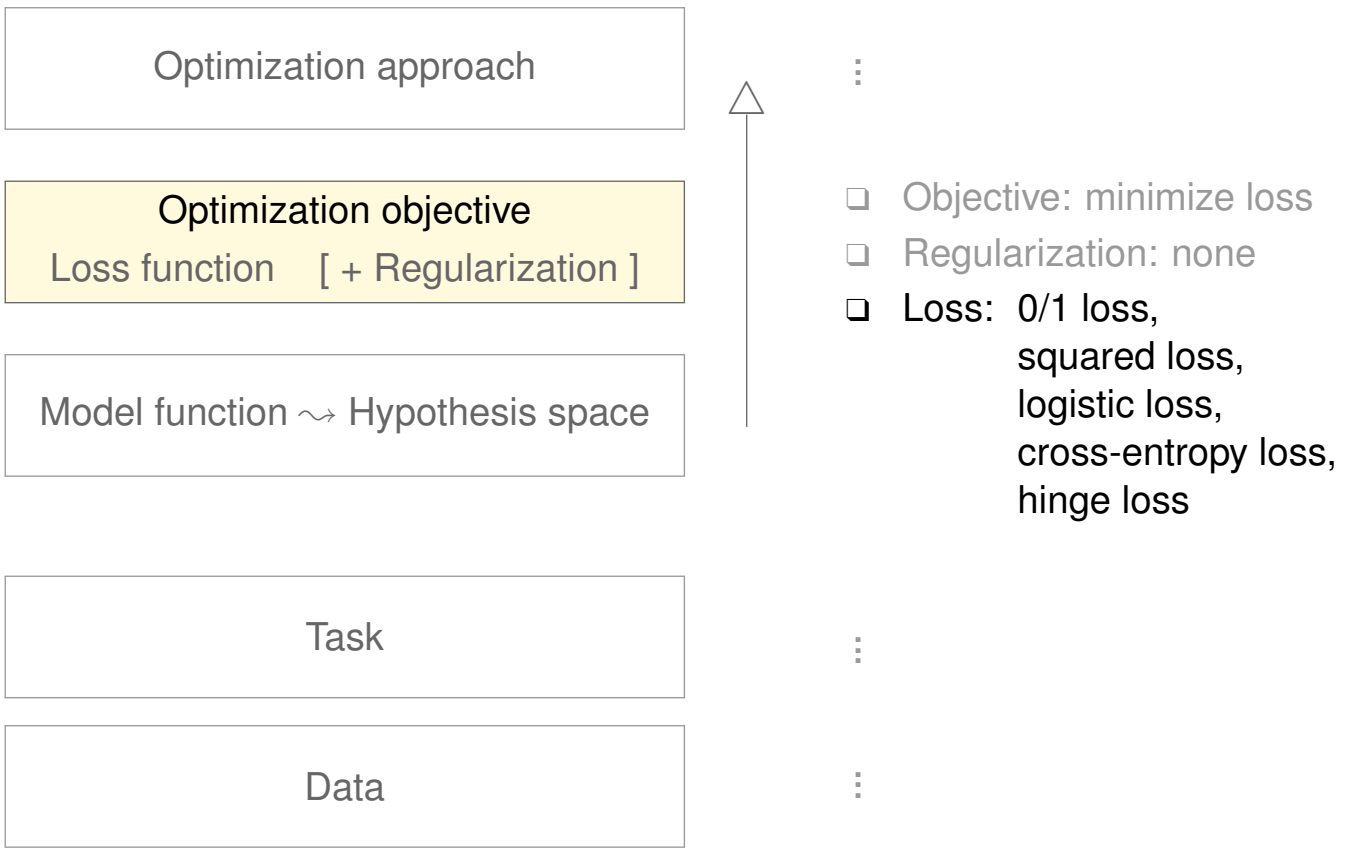
Chapter ML:III

III. Linear Models

- Logistic Regression
- Loss Computation in Detail
- Overfitting
- Regularization
- Gradient Descent in Detail

Loss Computation in Detail

Loss Computation in the Machine Learning Stack



Remarks:

- Given a hypothesis $\mathbf{w}$, its (global) loss, $L(\mathbf{w})$, tells us something about the effectiveness of $\mathbf{w}$. When used as sole criterion (e.g., no regularization is applied) we select from two hypotheses that with the smaller loss. I.e., the most effective hypothesis is found by loss minimization.

  Conversely, we call a function, whose minimization determines the most effective hypothesis, a loss function.

- Loss functions can be distinguished with respect to the problem class they are typically applied to: regression versus classification. Keep in mind that this distinction is not unique since loss functions with continuous output are applied to classification problems as well.

- Furthermore, we distinguish the

  1. *pointwise loss* $l(c, y(\mathbf{x}))$, which is computed for a single $\mathbf{x}$, and the
  2. *global loss* $L(\mathbf{w})$, which accumulates the pointwise losses of all $\mathbf{x} \in X$ for the weight vector $\mathbf{w}$ used in a specific $y(\mathbf{x})$:

$$L(\mathbf{w}) = \sum_{(\mathbf{x},c)\in D} l(c, y(\mathbf{x}))$$

  The pointwise loss is also called per-example loss. [p.268, Goodfellow/Bengio/Courville 2016]

- Instead of “loss” (function, computation) also the terms “error” (function, computation), “cost” (function, computation), or “performance” (function, computation) are used, usually with the same semantics as introduced here. We will use the term “error” for classification problems and the term “loss” for both classification and regression problems.

Loss Computation in Detail

Linear Regression

- The pointwise loss, $l(c, y(\mathbf{x}))$, quantifies the error introduced by some $\mathbf{x}$. The loss depends on a hypothesis $y(\mathbf{x})$ and the true class, c, of $\mathbf{x}$.
- For $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ we define the following pointwise loss functions:
  - 0/1 loss. $l_{0/1}(c, y(\mathbf{x})) = I_{\neq}(c, \text{sign}(y(\mathbf{x}))) = \begin{cases} 0 & \text{if } c = \text{sign}(y(\mathbf{x})) \\ 1 & \text{otherwise} \end{cases}$
  - Squared loss. $l_2(c, y(\mathbf{x})) = (c - y(\mathbf{x}))^2$

Loss Computation in Detail

Linear Regression (continued)

- The pointwise loss, $l(c, y(\mathbf{x}))$, quantifies the error introduced by some $\mathbf{x}$. The loss depends on a hypothesis $y(\mathbf{x})$ and the true class, c, of $\mathbf{x}$.
- For $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ we define the following pointwise loss functions:
  - 0/1 loss. $l_{0/1}(c, y(\mathbf{x})) = I_{\neq}(c, \text{sign}(y(\mathbf{x}))) = \begin{cases} 0 & \text{if } c = \text{sign}(y(\mathbf{x})) \\ 1 & \text{otherwise} \end{cases}$
  - Squared loss. $l_2(c, y(\mathbf{x})) = (c - y(\mathbf{x}))^2$

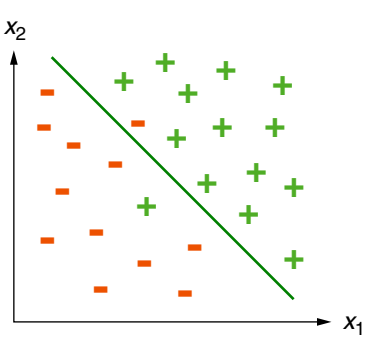
Loss Computation in Detail

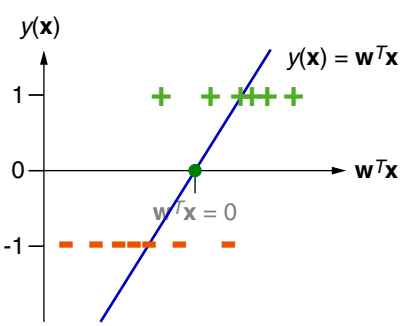
Linear Regression (continued)

- The pointwise loss, $l(c, y(\mathbf{x}))$, quantifies the error introduced by some $\mathbf{x}$. The loss depends on a hypothesis $y(\mathbf{x})$ and the true class, c, of $\mathbf{x}$.
- For $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ we define the following pointwise loss functions:
  - 0/1 loss. $l_{0/1}(c, y(\mathbf{x})) = I_{\neq}(c, \text{sign}(y(\mathbf{x}))) = \begin{cases} 0 & \text{if } c = \text{sign}(y(\mathbf{x})) \\ 1 & \text{otherwise} \end{cases}$
  - Squared loss. $l_2(c, y(\mathbf{x})) = (c - y(\mathbf{x}))^2$

Illustration for a particular $\mathbf{w}$:

Input space: $y(\mathbf{x})$ over hyperplane distance:





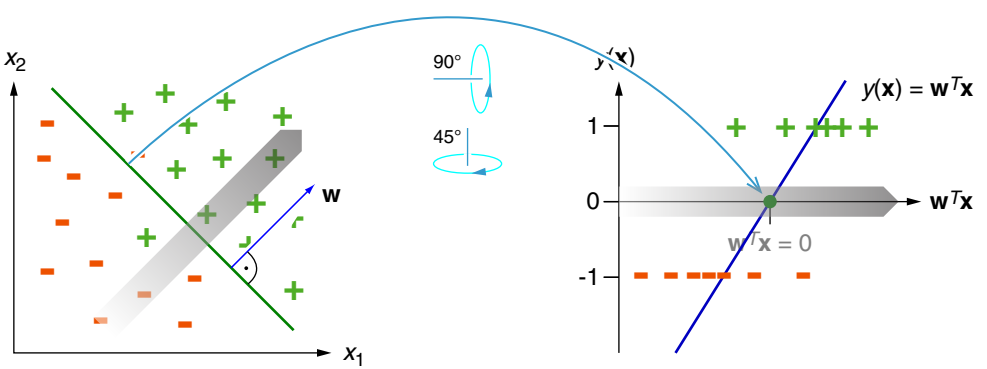
Loss Computation in Detail

Linear Regression (continued)

- The pointwise loss, $l(c, y(\mathbf{x}))$, quantifies the error introduced by some $\mathbf{x}$. The loss depends on a hypothesis $y(\mathbf{x})$ and the true class, c, of $\mathbf{x}$.
- For $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ we define the following pointwise loss functions:
  - 0/1 loss. $l_{0/1}(c, y(\mathbf{x})) = I_{\neq}(c, \text{sign}(y(\mathbf{x}))) = \begin{cases} 0 & \text{if } c = \text{sign}(y(\mathbf{x})) \\ 1 & \text{otherwise} \end{cases}$
  - Squared loss. $l_2(c, y(\mathbf{x})) = (c - y(\mathbf{x}))^2$

Illustration for a particular $\mathbf{w}$:

Input space: $y(\mathbf{x})$ over hyperplane distance:



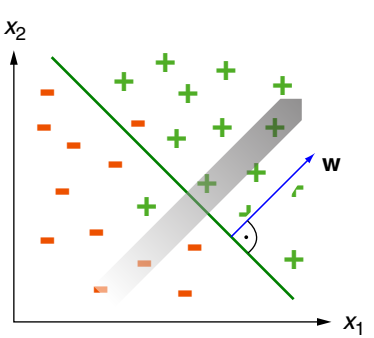
Loss Computation in Detail

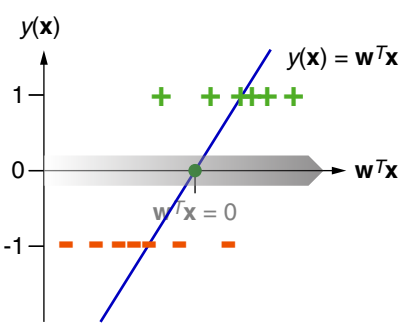
Linear Regression (continued)

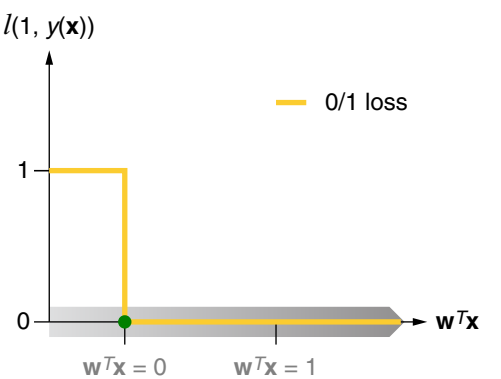
- The pointwise loss, $l(c, y(\mathbf{x}))$, quantifies the error introduced by some $\mathbf{x}$. The loss depends on a hypothesis $y(\mathbf{x})$ and the true class, c, of $\mathbf{x}$.
- For $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ we define the following pointwise loss functions:
  - 0/1 loss. $l_{0/1}(c, y(\mathbf{x})) = I_{\neq}(c, \text{sign}(y(\mathbf{x}))) = \begin{cases} 0 & \text{if } c = \text{sign}(y(\mathbf{x})) \\ 1 & \text{otherwise} \end{cases}$
  - Squared loss. $l_2(c, y(\mathbf{x})) = (c - y(\mathbf{x}))^2$

Illustration for a particular $\mathbf{w}$:

Input space: $y(\mathbf{x})$ over hyperplane distance: Loss over hyperplane distance:







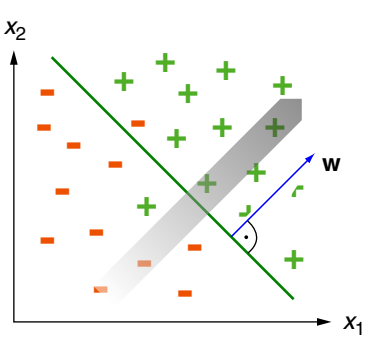
Loss Computation in Detail

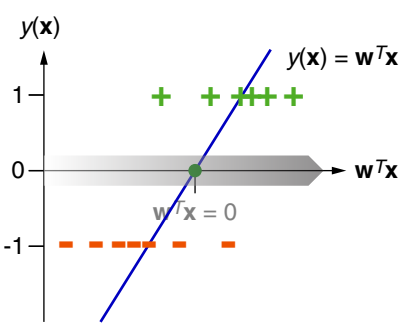
Linear Regression (continued)

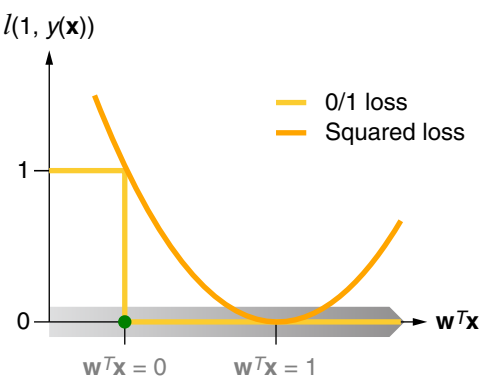
- The pointwise loss, $l(c, y(\mathbf{x}))$, quantifies the error introduced by some $\mathbf{x}$. The loss depends on a hypothesis $y(\mathbf{x})$ and the true class, c, of $\mathbf{x}$.
- For $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ we define the following pointwise loss functions:
  - 0/1 loss. $l_{0/1}(c, y(\mathbf{x})) = I_{\neq}(c, \text{sign}(y(\mathbf{x}))) = \begin{cases} 0 & \text{if } c = \text{sign}(y(\mathbf{x})) \\ 1 & \text{otherwise} \end{cases}$
  - Squared loss. $l_2(c, y(\mathbf{x})) = (c - y(\mathbf{x}))^2$

Illustration for a particular $\mathbf{w}$:

Input space: $y(\mathbf{x})$ over hyperplane distance: Loss over hyperplane distance:







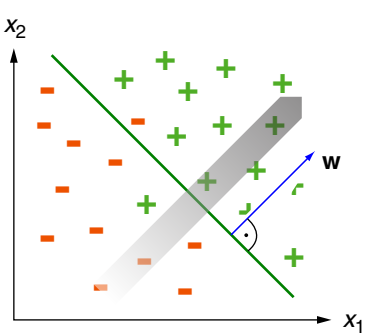
Loss Computation in Detail

Linear Regression (continued)

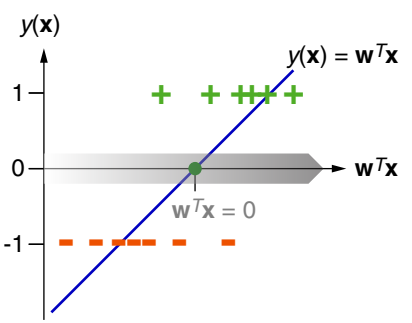
- The pointwise loss, $l(c, y(\mathbf{x}))$, quantifies the error introduced by some $\mathbf{x}$. The loss depends on a hypothesis $y(\mathbf{x})$ and the true class, c, of $\mathbf{x}$.
- For $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ we define the following pointwise loss functions:
  - 0/1 loss. $l_{0/1}(c, y(\mathbf{x})) = I_{\neq}(c, \text{sign}(y(\mathbf{x}))) = \begin{cases} 0 & \text{if } c = \text{sign}(y(\mathbf{x})) \\ 1 & \text{otherwise} \end{cases}$
  - Squared loss. $l_2(c, y(\mathbf{x})) = (c - y(\mathbf{x}))^2$

Illustration for a particular $\mathbf{w}$:

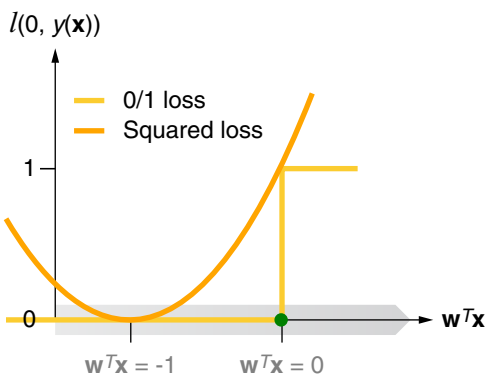
Input space:



$y(\mathbf{x})$ over hyperplane distance:



Loss over hyperplane distance:



Remarks:

- The 0/1 loss computes the misclassification error. Recall in this regard the definition of the true misclassification rate.
- The pointwise squared loss computes the squared residual. The global squared loss, $L_2(\mathbf{w}) = \sum_{(\mathbf{x},c)\in D} l_2(c, y(\mathbf{x}))$, hence computes the residual sum of squares (RSS) related to some $\mathbf{w}$.
- $I_{\neq}$ is an indicator function that returns 1 if its arguments are *unequal* (and 0 if its arguments are equal).
- Recap. We label $y(0)$ with the "positive" class and define sign$(0) = 1$ here.
- Regarding the illustration: $\mathbf{w}^T\mathbf{x}$ is the hyperplane distance in relation to $||\mathbf{w}||$, the length of $\mathbf{w}$. By scaling $\mathbf{w}$ such that $||\mathbf{w}|| = 1$ the hyperplane distance $\mathbf{w}^T\mathbf{x}$ becomes normalized and is also called "geometric distance".

Loss Computation in Detail

Logistic Regression

- The pointwise loss, $l(c, y(\mathbf{x}))$, quantifies the error introduced by some $\mathbf{x}$. The loss depends on a hypothesis $y(\mathbf{x})$ and the true class, c, of $\mathbf{x}$.
- For $y(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$ we define the following pointwise loss functions:
  - 0/1 loss. $l_{0/1}(c, y(\mathbf{x})) = I_{\neq}(c, \text{sign}(y(\mathbf{x}) - 0.5))$ [decision rule]
  - Logistic loss. $l_{\sigma}(c, y(\mathbf{x})) = \begin{cases} -\log(y(\mathbf{x})) & \text{if } c = 1 \\ -\log(1 - y(\mathbf{x})) & \text{if } c = 0 \end{cases}$ [derivation]

Loss Computation in Detail

Logistic Regression (continued)

- The pointwise loss, $l(c, y(\mathbf{x}))$, quantifies the error introduced by some $\mathbf{x}$. The loss depends on a hypothesis $y(\mathbf{x})$ and the true class, c, of $\mathbf{x}$.
- For $y(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$ we define the following pointwise loss functions:
  - 0/1 loss. $l_{0/1}(c, y(\mathbf{x})) = I_{\neq}(c, \text{sign}(y(\mathbf{x}) - 0.5))$ [decision rule]
  - Logistic loss. $l_{\sigma}(c, y(\mathbf{x})) = \begin{cases} -\log(y(\mathbf{x})) & \text{if } c = 1 \\ -\log(1 - y(\mathbf{x})) & \text{if } c = 0 \end{cases}$ [derivation]

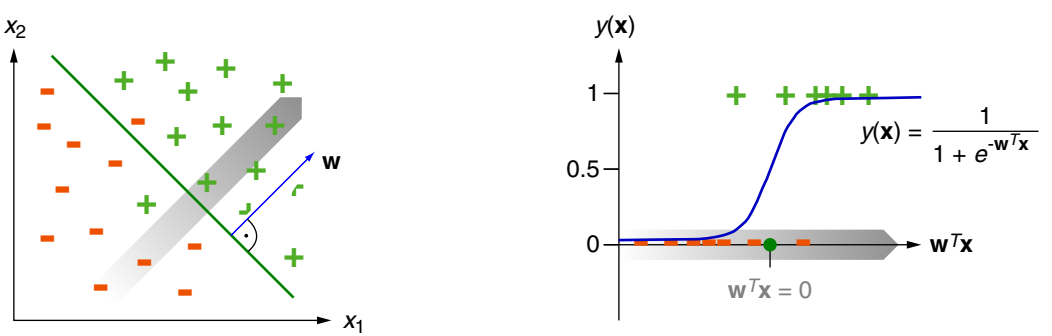
Loss Computation in Detail

Logistic Regression (continued)

- The pointwise loss, $l(c, y(\mathbf{x}))$, quantifies the error introduced by some $\mathbf{x}$. The loss depends on a hypothesis $y(\mathbf{x})$ and the true class, c, of $\mathbf{x}$.
- For $y(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$ we define the following pointwise loss functions:
  - 0/1 loss. $l_{0/1}(c, y(\mathbf{x})) = I_{\neq}(c, \text{sign}(y(\mathbf{x}) - 0.5))$ [decision rule]
  - Logistic loss. $l_{\sigma}(c, y(\mathbf{x})) = \begin{cases} -\log(y(\mathbf{x})) & \text{if } c = 1 \\ -\log(1 - y(\mathbf{x})) & \text{if } c = 0 \end{cases}$ [derivation]

Illustration for a particular $\mathbf{w}$:

Input space: $y(\mathbf{x})$ over hyperplane distance:



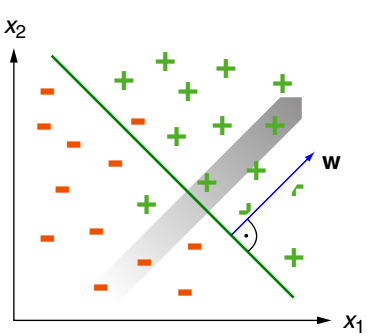
Loss Computation in Detail

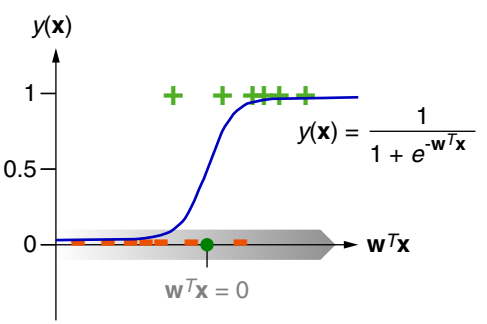
Logistic Regression (continued)

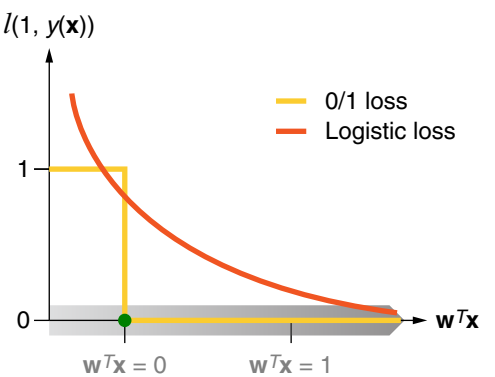
- The pointwise loss, $l(c, y(\mathbf{x}))$, quantifies the error introduced by some $\mathbf{x}$. The loss depends on a hypothesis $y(\mathbf{x})$ and the true class, c, of $\mathbf{x}$.
- For $y(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$ we define the following pointwise loss functions:
  - 0/1 loss. $l_{0/1}(c, y(\mathbf{x})) = I_{\neq}(c, \text{sign}(y(\mathbf{x}) - 0.5))$ [decision rule]
  - Logistic loss. $l_{\sigma}(c, y(\mathbf{x})) = \begin{cases} -\log(y(\mathbf{x})) & \text{if } c = 1 \\ -\log(1 - y(\mathbf{x})) & \text{if } c = 0 \end{cases}$ [derivation]

Illustration for a particular $\mathbf{w}$:

Input space: $y(\mathbf{x})$ over hyperplane distance: Loss over hyperplane distance:







Remarks:

- As before, the 0/1 loss computes the misclassification error.
- The pointwise logistic loss can be rewritten by combining the two cases algebraically:

$$l_\sigma(c, y(\mathbf{x})) \;=\; -c \cdot \log(y(\mathbf{x})) - (1-c) \cdot \log(1-y(\mathbf{x}))$$

  $L_\sigma(\mathbf{w}) = \sum_{(\mathbf{x},c)\in D} l_\sigma(c, y(\mathbf{x}))$ computes the global logistic loss related to some $\mathbf{w}$.
- Recall from the derivation of the logistic loss $L_\sigma(\mathbf{w})$ that its minimization determines $\mathbf{w}_{\text{ML}}$, the most probable hypothesis in $\mathbf{R}^{p+1}$ under the logistic regression model.
- Recap. $I_{\neq}$ is an indicator function that returns 1 if its arguments are *unequal* (and 0 if its arguments are equal).
- Recap. We label $y(0)$ with the "positive" class and define $\text{sign}(0) = 1$ here.

Remarks (different roles of loss functions) :

- Observe that loss functions are employed at two places (in two roles) in an optimization approach:
  1. For the fitting of the data (i.e., the parameter update during regression / optimization / hyperplane search), where a new position of the hyperplane is computed.

     Example: Lines 6+7 in the BGD$_\sigma$ Algorithm.
  2. For the evaluation of a hypothesis' effectiveness, where the proportion of correctly and misclassified examples is analyzed.

     Example: Line 10 in the BGD$_\sigma$ Algorithm.
     General: section Evaluating Effectiveness of part Machine Learning Basics.

  Typically, (1) fitting (optimization) and (2) effectiveness evaluation are done with different loss functions. E.g., logistic regression uses L_σ and $L_{0/1}$ for fitting and evaluation respectively. However, linear regression (not classification) uses RSS (the L_2 loss) for both fitting and evaluation. The basic perceptron learning algorithm uses the misclassification information (the $L_{0/1}$ loss) for both fitting and evaluation.

Chapter ML:III (continued)

III. Linear Models

- Logistic Regression
- Loss Computation in Detail
- Overfitting
- Regularization
- Gradient Descent in Detail

Overfitting

Definition 9 (Overfitting)

Let D be a multiset of examples and let H be a hypothesis space. The hypothesis $h \in H$ is considered to overfit D if an $h' \in H$ with the following property exists:

$$\mathit{Err}(h, D) < \mathit{Err}(h', D) \quad \text{and} \quad \mathit{Err}^*(h) > \mathit{Err}^*(h'),$$

where $\mathit{Err}^*(h)$ denotes the true misclassification rate of h, while $\mathit{Err}(h, D)$ denotes the error of h for D.

[see continuation]

Overfitting

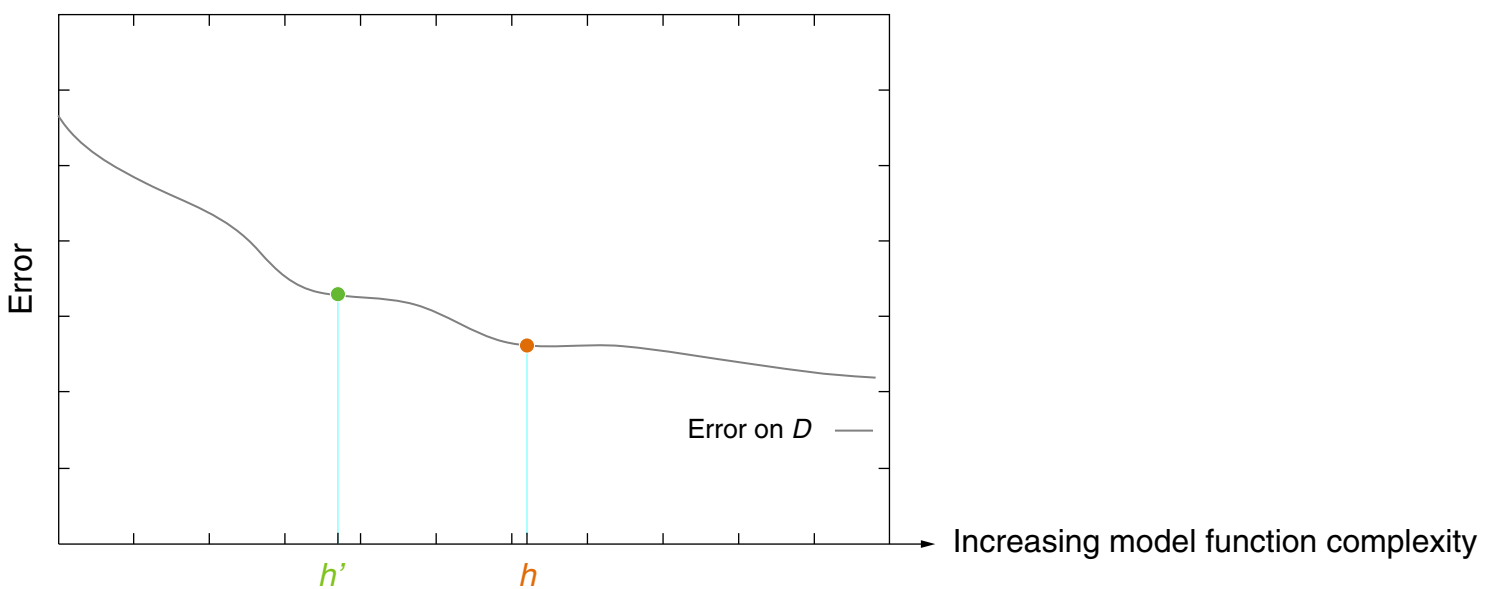
Definition 9 (Overfitting)

Let D be a multiset of examples and let H be a hypothesis space. The hypothesis $h \in H$ is considered to overfit D if an $h' \in H$ with the following property exists:

$$\mathit{Err}(h, D) < \mathit{Err}(h', D) \quad \text{and} \quad \mathit{Err}^*(h) > \mathit{Err}^*(h'),$$

where $\mathit{Err}^*(h)$ denotes the true misclassification rate of h, while $\mathit{Err}(h, D)$ denotes the error of h for D.

[see continuation]



Overfitting

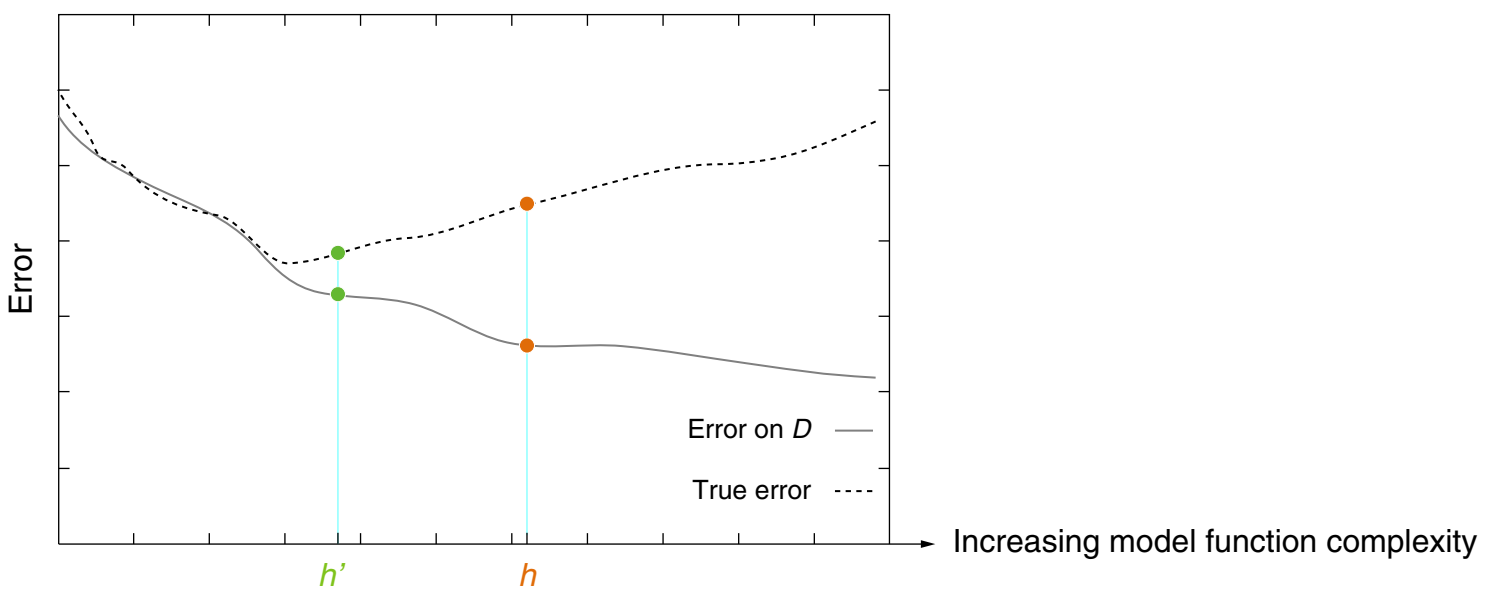
Definition 9 (Overfitting)

Let D be a multiset of examples and let H be a hypothesis space. The hypothesis $h \in H$ is considered to overfit D if an $h' \in H$ with the following property exists:

$$\mathit{Err}(h, D) < \mathit{Err}(h', D) \quad \text{and} \quad \mathit{Err}^*(h) > \mathit{Err}^*(h'),$$

where $\mathit{Err}^*(h)$ denotes the true misclassification rate of h, while $\mathit{Err}(h, D)$ denotes the error of h for D.

[see continuation]



Overfitting

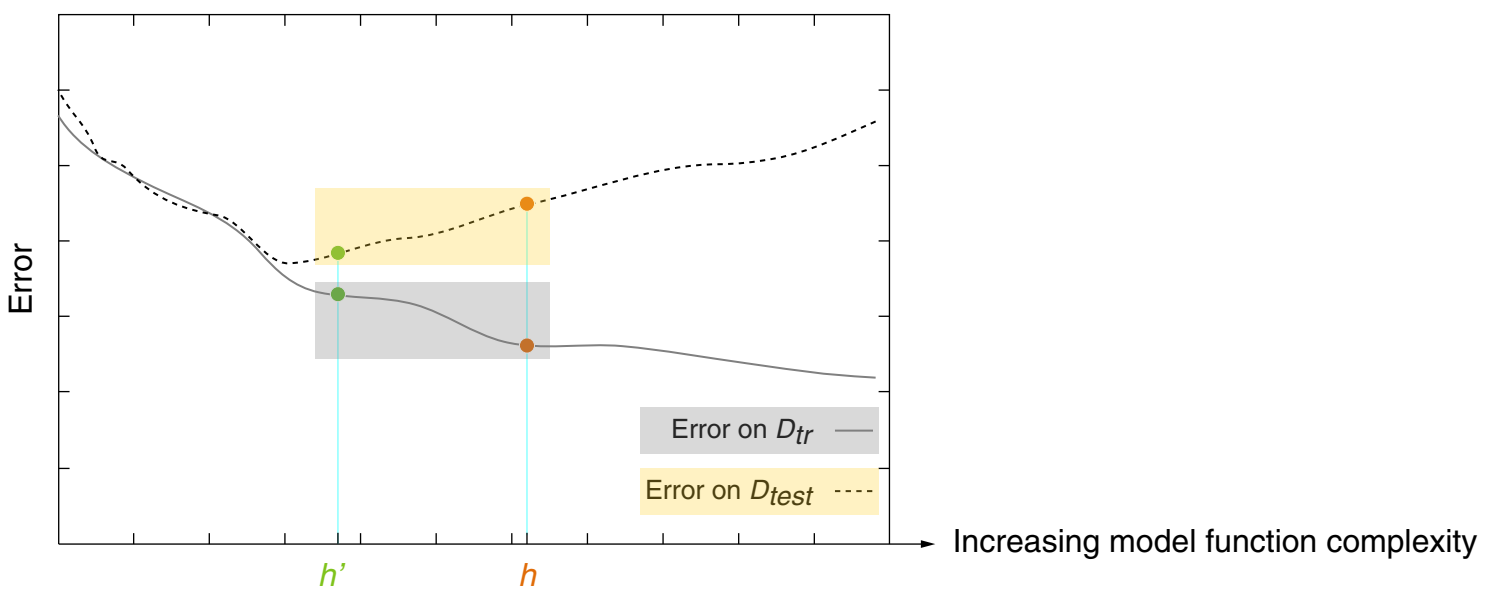
Definition 9 (Overfitting)

Let D be a multiset of examples and let H be a hypothesis space. The hypothesis $h \in H$ is considered to overfit D if an $h' \in H$ with the following property exists:

$$\mathit{Err}(h, D) < \mathit{Err}(h', D) \quad \text{and} \quad \mathit{Err}^*(h) > \mathit{Err}^*(h'),$$

where $\mathit{Err}^*(h)$ denotes the true misclassification rate of h, while $\mathit{Err}(h, D)$ denotes the error of h for D.

[see continuation]



Overfitting

Definition 9 (Overfitting)

Let D be a multiset of examples and let H be a hypothesis space. The hypothesis $h \in H$ is considered to overfit D if an $h' \in H$ with the following property exists:

$$\mathit{Err}(h, D) < \mathit{Err}(h', D) \quad \text{and} \quad \mathit{Err}^*(h) > \mathit{Err}^*(h'),$$

where $\mathit{Err}^*(h)$ denotes the true misclassification rate of h, while $\mathit{Err}(h, D)$ denotes the error of h for D.

[see continuation]

Reasons for overfitting are often rooted in the example set D:

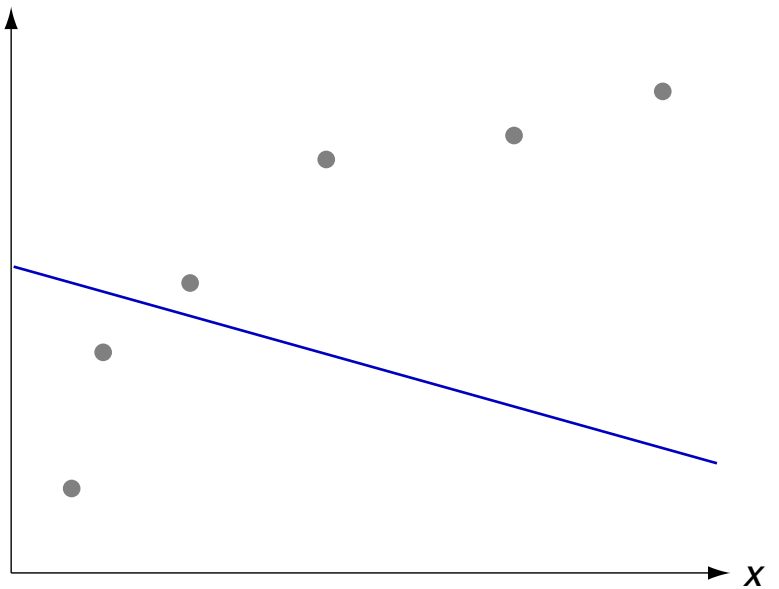
- D is noisy and we "learn noise."
- D is biased and hence not representative.
- D is too small and hence pretends unrealistic data properties.

Overfitting

Example: Linear Regression

Overfitting

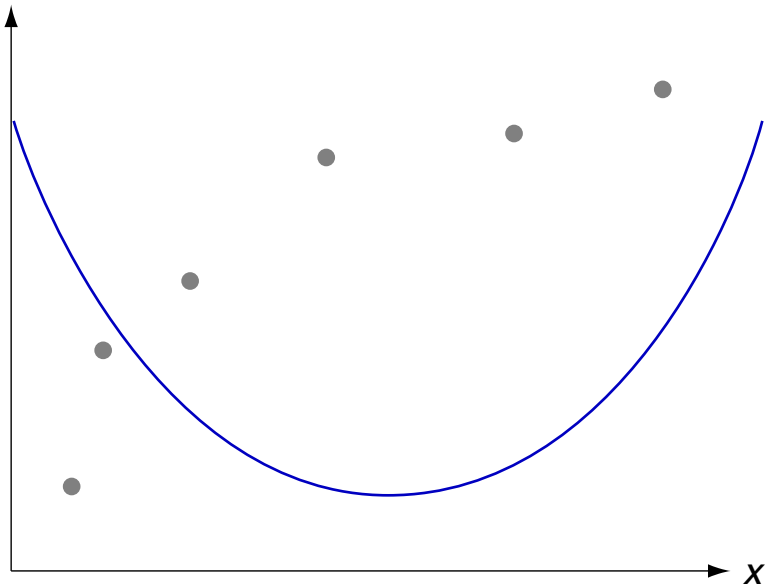
Example: Linear Regression (continued)



(a) $y(x) = w_0 + w_1 \cdot x$

Overfitting

Example: Linear Regression (continued)

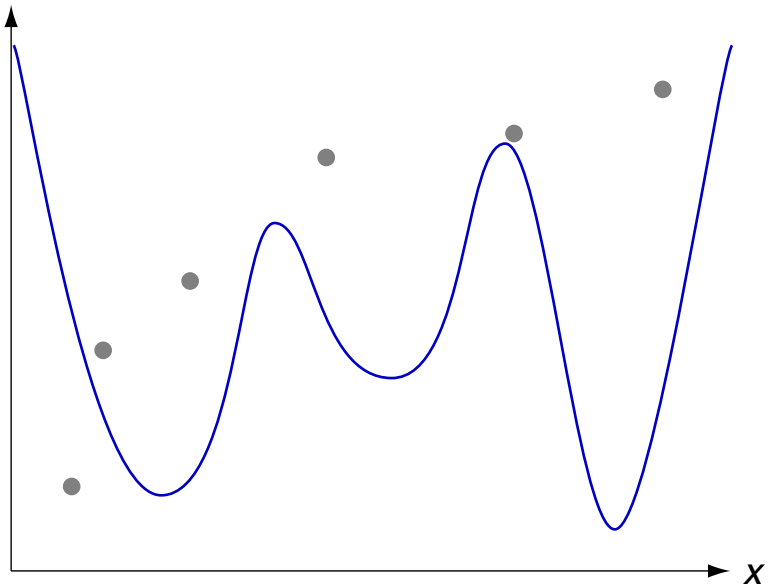


(b) $y(x) = w_0 + w_1 \cdot x + w_2 \cdot x^2$ (basis expansion)

$$y(x) = (w_0 \;\; w_1 \;\; w_2)\begin{pmatrix} 1 \\ x \\ x^2 \end{pmatrix} =: \mathbf{w}^T \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = \mathbf{w}^T\mathbf{x} = y(\mathbf{x}), \quad \text{where } x_0 = 1,\ x_1 = x,\ x_2 = x^2$$

Overfitting

Example: Linear Regression (continued)



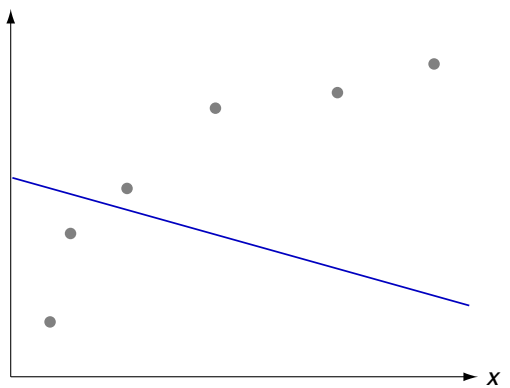
(c) $y(x) = w_0 + \sum_{j=1}^{6} w_j \cdot x^j$ (basis expansion)

$y(x) =: \mathbf{w}^T\mathbf{x} = y(\mathbf{x}),$ where $x_0 = 1,\ x_j = x^j,\ j = 1, \ldots, 6$

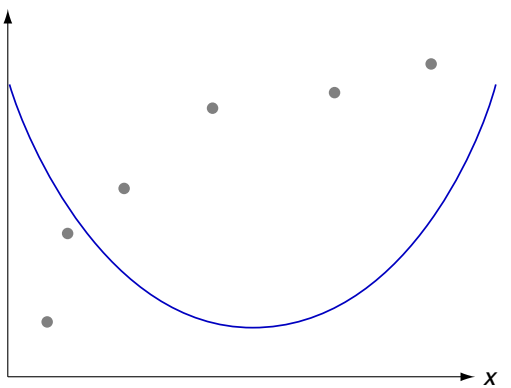
Overfitting

Example: Linear Regression (continued)

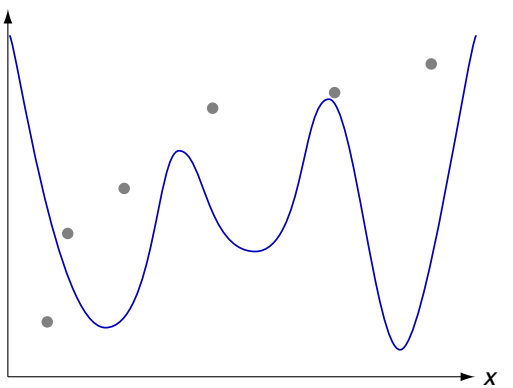
Given the three polynomial model functions of degrees 1, 2, and 6, and a training set D_{tr}, select the function that best fits the data:



(a)



(b)



(c)

Questions:

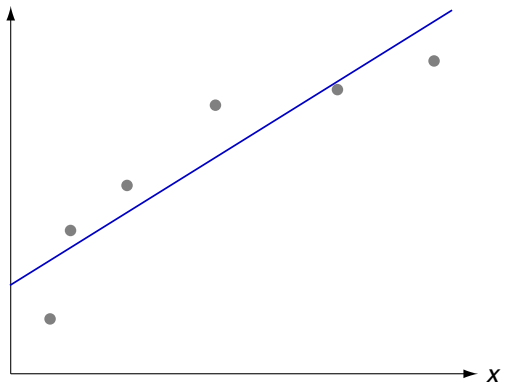
(1) How to choose a suited model function / hypothesis space H?

(2) How to parameterize a model function / pick an element from H?

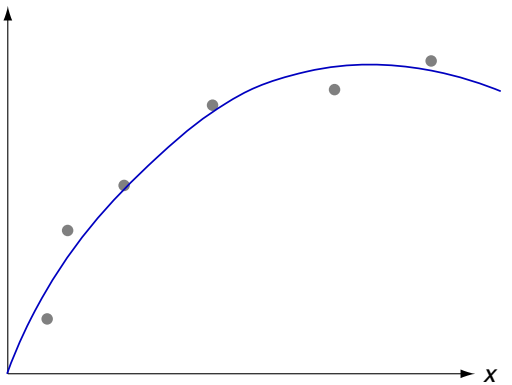
Overfitting

Example: Linear Regression (continued)

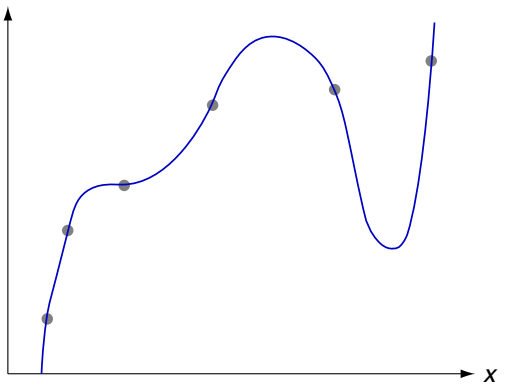
Given the three polynomial model functions of degrees 1, 2, and 6, and a training set D_{tr}, select the function that best fits the data:



(a) $\text{RSS}(\mathbf{w}) \gg 0$



(b) $\text{RSS}(\mathbf{w}) > 0$



(c) $\text{RSS}(\mathbf{w}) = 0$

Questions:

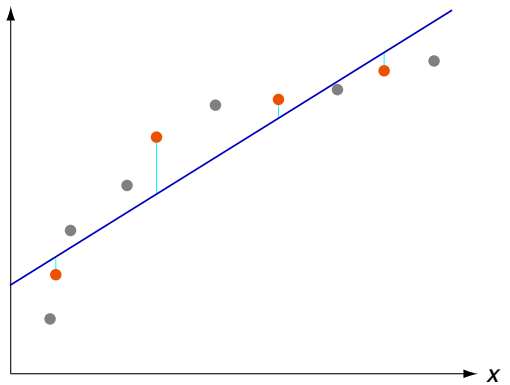
(1) How to choose a suited model function / hypothesis space H?

(2) How to parameterize a model function / pick an element from H?

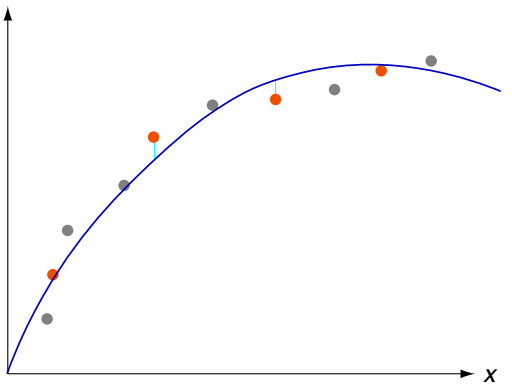
Overfitting

Example: Linear Regression (continued)

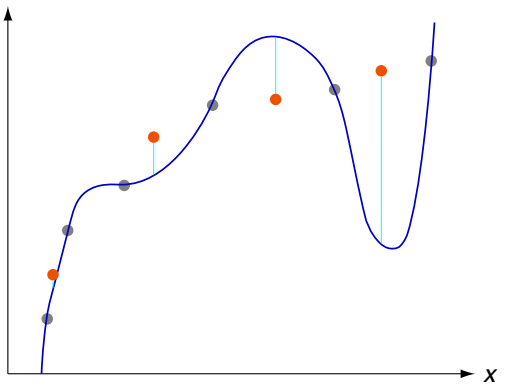
Given the three polynomial model functions of degrees 1, 2, and 6, and a training set D_{tr}, select the function that best fits the data:



(a) $\text{RSS}(\mathbf{w}) \gg 0$



(b) $\text{RSS}(\mathbf{w}) > 0$



(c) $\text{RSS}(\mathbf{w}) \gg 0$

Let D_{test} be a set of test examples.

If $D = D_{tr} \cup D_{test}$ is representative of the real-world population in X, the quadratic model function (b), $y(x) = w_0 + w_1 \cdot x + w_2 \cdot x^2$, is the closest match.

Overfitting

Definition 9 (Overfitting (continued))

Let D be a set of examples and let H be a hypothesis space. The hypothesis $h \in H$ is considered to overfit D if an $h' \in H$ with the following property exists:

$$Err(h, D) < Err(h', D) \quad \text{and} \quad Err^*(h) > Err^*(h'),$$

where $Err^*(h)$ denotes the true misclassification rate of h, while $Err(h, D)$ denotes the error of h for D.

Let $D_{tr} \subset D$ be the training set. Then $Err^*(h)$ can be estimated with a test set $D_{test} \subset D$ where $D_{test} \cap D_{tr} = \emptyset$ [holdout estimation]. The hypothesis $h \in H$ is considered to overfit D if an $h' \in H$ with the following property exists:

$$Err_{tr}(h) < Err_{tr}(h') \quad \text{and} \quad Err(h, D_{test}) > Err(h', D_{test})$$

In particular holds: $Err(h, D_{test}) \gg Err_{tr}(h)$

Overfitting

Mitigation Strategies

How to detect overfitting:

- Visual inspection
  Apply projection or embedding for dimensionalities $p > 3$.
- Validation
  Given a test set, the difference $\mathit{Err}_{\mathit{test}}(y, D_{\mathit{test}}) - \mathit{Err}_{\mathit{tr}}(y)$ is too large.

Overfitting

Mitigation Strategies (continued)

How to detect overfitting:

- Visual inspection
  Apply projection or embedding for dimensionalities $p > 3$.
- Validation
  Given a test set, the difference $\mathit{Err}_{\mathit{test}}(y, D_{\mathit{test}}) - \mathit{Err}_{\mathit{tr}}(y)$ is too large.

How to address overfitting:

- Increase the quantity and / or the quality of the training data D.
  Quantity: More data averages out noise.
  Quality: Omitting “poor examples” allows a better fit, but is problematic though.
- Early stopping of the optimization (training) process.
  Criterion: $\mathit{Err}_{\mathit{test}}(y, D_{\mathit{test}}) - \mathit{Err}_{\mathit{tr}}(y)$ increases with the number of iterations (training time).

Overfitting

Mitigation Strategies (continued)

How to detect overfitting:

- Visual inspection
  Apply projection or embedding for dimensionalities $p > 3$.
- Validation
  Given a test set, the difference $Err_{test}(y, D_{test}) - Err_{tr}(y)$ is too large.

How to address overfitting:

- Increase the quantity and / or the quality of the training data D.
  Quantity: More data averages out noise.
  Quality: Omitting "poor examples" allows a better fit, but is problematic though.
- Early stopping of the optimization (training) process.
  Criterion: $Err_{test}(y, D_{test}) - Err_{tr}(y)$ increases with the number of iterations (training time).
- Regularization: Increase model bias by constraining the hypothesis space.
  (1) Model function: Consider functions of lower complexity / VC dimension. [Wikipedia]
  (2) Hypothesis $\mathbf{w}$: Bound the absolute values of the weights in $\vec{\mathbf{w}}$ of a model function.

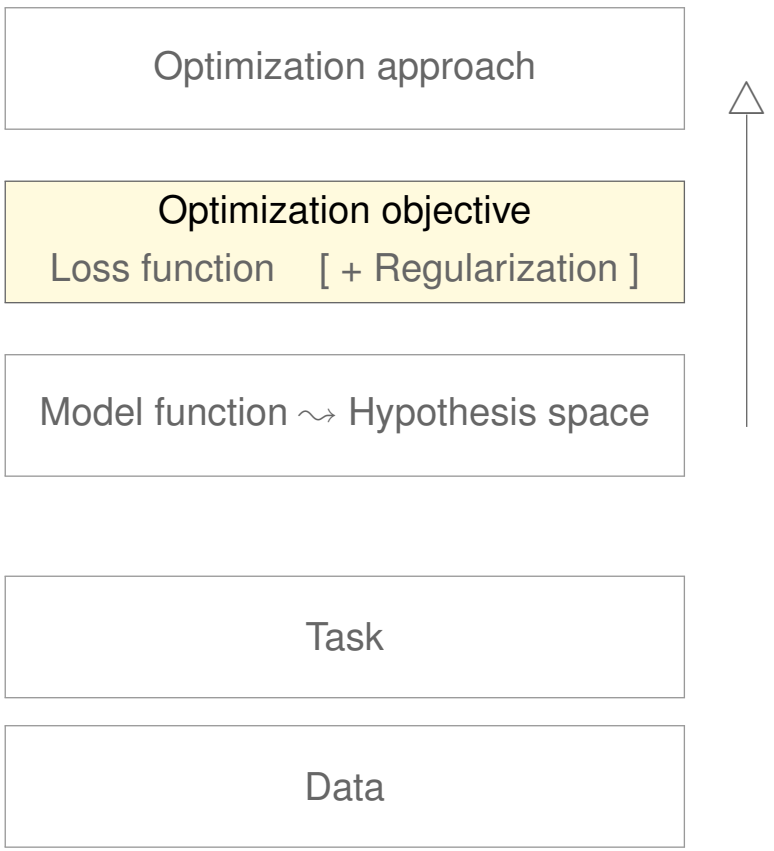
Chapter ML:III (continued)

III. Linear Models

- Logistic Regression
- Loss Computation in Detail
- Overfitting
- Regularization
- Gradient Descent in Detail

Regularization

Regularization in the Machine Learning Stack



- ❑ Objective: minimize objective function
- ❑ Regularization: Euclidean norm, absolute-value norm
- ❑ Loss: $l(c, y(\mathbf{x}))$

Regularization

Bound the Absolute Values of the Weights $\mathbf{w}$

Principle: Add to the loss function (term) a regularization function (term), $R(\mathbf{w})$:

$$\mathcal{L}(\mathbf{w}) \;=\; L(\mathbf{w}) \;+\; \lambda \cdot R(\mathbf{w}), \qquad \ell(\mathbf{w}) \;=\; l(c, y(\mathbf{x})) \;+\; \frac{\lambda}{n} \cdot R(\mathbf{w}),$$

where $\lambda \geq 0$ controls the impact of $R(\mathbf{w})$, $R(\mathbf{w}) \geq 0$.

Regularization

Bound the Absolute Values of the Weights $\mathbf{w}$ (continued)

Principle: Add to the loss function (term) a regularization function (term), $R(\mathbf{w})$:

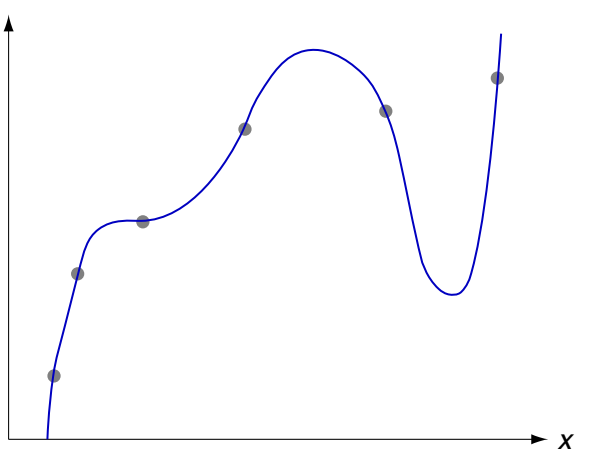
$$\mathcal{L}(\mathbf{w}) \;=\; L(\mathbf{w}) \;+\; \lambda \cdot R(\mathbf{w}), \qquad \ell(\mathbf{w}) \;=\; l(c, y(\mathbf{x})) \;+\; \frac{\lambda}{n} \cdot R(\mathbf{w}),$$

where $\lambda \geq 0$ controls the impact of $R(\mathbf{w})$, $R(\mathbf{w}) \geq 0$.

Example (c) (continued) :

- $y(x) = w_0 + \sum_{j=1}^{6} w_j \cdot x^j$
- $L(\mathbf{w}) = \text{RSS}(\mathbf{w}) = \sum_{i=1}^{n} (y_i - y(x_i))^2$
- $R(\mathbf{w}) = |w_1| + |w_2| + \ldots + |w_6|$
  $\lambda = 0$

$\rightsquigarrow$ $\hat{\mathbf{w}} = (-0.7,\ 15.4,\ -80.6,\ 174.9,\ -99.5,\ -113.7,\ 109.7)^T$



Regularization

Bound the Absolute Values of the Weights $\mathbf{w}$ (continued)

Principle: Add to the loss function (term) a regularization function (term), $R(\mathbf{w})$:

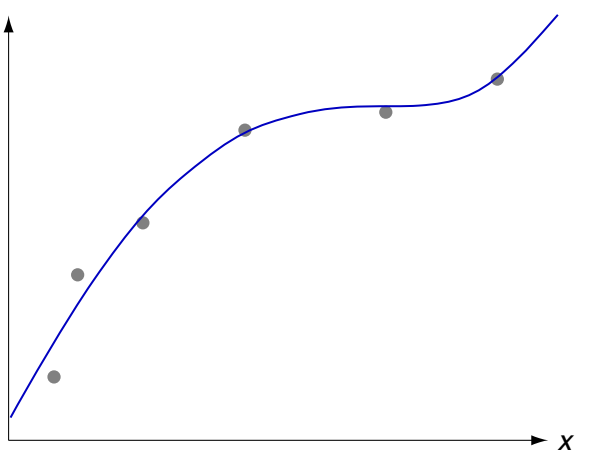
$$\mathcal{L}(\mathbf{w}) \;=\; L(\mathbf{w}) \;+\; \lambda \cdot R(\mathbf{w}), \qquad \ell(\mathbf{w}) \;=\; l(c, y(\mathbf{x})) \;+\; \frac{\lambda}{n} \cdot R(\mathbf{w}),$$

where $\lambda \geq 0$ controls the impact of $R(\mathbf{w})$, $R(\mathbf{w}) \geq 0$.

Example (c) (continued) :

- $y(x) = w_0 + \sum_{j=1}^{6} w_j \cdot x^j$
- $L(\mathbf{w}) = \text{RSS}(\mathbf{w}) = \sum_{i=1}^{n} (y_i - y(x_i))^2$
- $R(\mathbf{w}) = |w_1| + |w_2| + \ldots + |w_6|$
  $\lambda = 0.001$

$\rightsquigarrow$ $\hat{\mathbf{w}} = (0.01,\ 2.0,\ -1.73,\ -0.22,\ 0.0,\ 0.0,\ 0.8)^T$



Regularization

Bound the Absolute Values of the Weights $\mathbf{w}$ (continued)

Principle: Add to the loss function (term) a regularization function (term), $R(\mathbf{w})$:

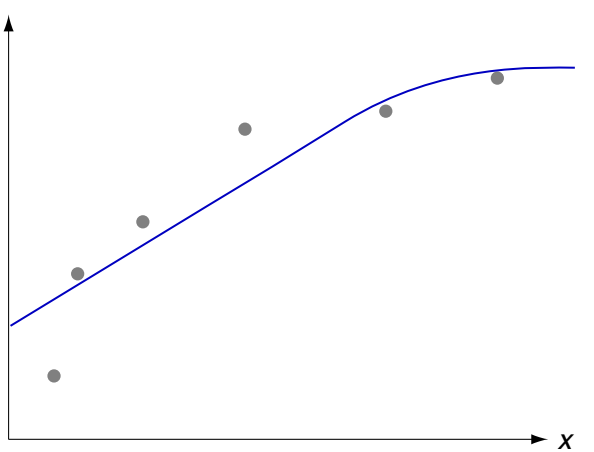
$$\mathcal{L}(\mathbf{w}) \;=\; L(\mathbf{w}) \;+\; \lambda \cdot R(\mathbf{w}), \qquad \ell(\mathbf{w}) \;=\; l(c, y(\mathbf{x})) \;+\; \frac{\lambda}{n} \cdot R(\mathbf{w}),$$

where $\lambda \geq 0$ controls the impact of $R(\mathbf{w})$, $R(\mathbf{w}) \geq 0$.

Example (c) (continued):

- $y(x) = w_0 + \sum_{j=1}^{6} w_j \cdot x^j$
- $L(\mathbf{w}) = \text{RSS}(\mathbf{w}) = \sum_{i=1}^{n} (y_i - y(x_i))^2$
- $R(\mathbf{w}) = |w_1| + |w_2| + \ldots + |w_6|$
  $\lambda = 0.02$

$\rightsquigarrow \quad \hat{\mathbf{w}} = (0.17,\ 0.73,\ 0.0,\ -0.21,\ -0.01,\ -0.01,\ 0.0)^T$



Regularization

Bound the Absolute Values of the Weights $\mathbf{w}$ (continued)

Principle: Add to the loss function (term) a regularization function (term), $R(\mathbf{w})$:

$$\mathcal{L}(\mathbf{w}) \;=\; L(\mathbf{w}) \;+\; \lambda \cdot R(\mathbf{w}), \qquad \ell(\mathbf{w}) \;=\; l(c, y(\mathbf{x})) \;+\; \frac{\lambda}{n} \cdot R(\mathbf{w}),$$

where $\lambda \geq 0$ controls the impact of $R(\mathbf{w})$, $R(\mathbf{w}) \geq 0$.

Observations:

- Model complexity depends (also) on the magnitude of the weights $\mathbf{w}$.
- Minimizing $L(\mathbf{w})$ sets no bounds on the weights $\mathbf{w}$.
- Regularization is achieved with a "counterweight" $\lambda \cdot R(\mathbf{w})$ that grows with $\mathbf{w}$.
- Aside from λ no additional hyperparameter is introduced.

Remarks:

- $\mathcal{L}(\mathbf{w})$ is called (global) “objective function”, “cost function”, or “error function”; $\ell(\mathbf{w})$ is the pointwise counterpart.
- The regularization term constrains the magnitude of the direction vector of the hyperplane, progressively reducing the hyperplane’s steepness as λ increases. The intercept w_0 is adjusted accordingly through minimization of $\mathcal{L}(\mathbf{w})$ but must not be part of the regularization term itself, which would lead to an incorrect solution.
- To denote the difference, we write $\mathbf{w} \equiv (w_0, w_1, \ldots, w_p)^T$ to refer to the entire parameter vector (the actual hypothesis), and $\vec{\mathbf{w}} \equiv (w_1, \ldots, w_p)^T$ for the direction vector excluding w_0.
- About choosing λ:
  - “No black-box procedures for choosing the regularization parameter λ are available, and most likely will never exist.” [Hansen/Hanke 1993]
  - How to calculate the regularization parameter λ in linear regression. [stackoverflow]

Remarks (continued):

- The term “regularization” derives from “regular”, a synonym for “smooth” within the context of model functions. [stackexchange]
- Regularization is applied in settings where the set of examples D is much smaller than the population of real-world objects O. Under the conditions of the Inductive Learning Hypothesis we can infer from D a hypothesis h that generalizes sufficiently well to the entire population—if h is sufficiently simple, stable (wrt. changes in D), and smooth, which can be reached with regularization.

  However, if D covers (nearly) the entire population, minimizing the loss $L(\mathbf{w})$ takes precedence over additional restrictions $R(\mathbf{w})$ regarding the simplicity, the stability, and the smoothness of h.
- The origins of regularization go back to the fields of inverse problems and ill-posed problems. Solving an inverse problem means calculating from a set of observations the causal factors that produced them. [Wikipedia]

  Inverse problems are often ill-posed, where “ill-posedness” is defined as not being “well-posed”. In turn, a mathematical problem is called well-posed if (1) a solution exists, (2) the solution is unique, (3) the solution’s behavior changes continuously with the initial conditions. [Wikipedia]

  Under certain assumptions the problem of learning from examples forms an inverse problem. [deVito 2005]

Regularization

The Vector Norm as Regularization Function

- Ridge regression. $R_{||\vec{\mathbf{w}}||_2^2}(\mathbf{w}) = \sum_{i=1}^{p} w_i^2 = \vec{\mathbf{w}}^T\vec{\mathbf{w}}$
- Lasso regression. $R_{||\vec{\mathbf{w}}||_1}(\mathbf{w}) = \sum_{i=1}^{p} |w_i|$

Remarks:

- Ridge regression predates lasso regression. It is also known as weight decay in machine learning, and with multiple independent discoveries, it is variously known as the Tikhonov-Miller method, the Phillips-Twomey method, the constrained linear inversion method, and the method of linear regularization. [Wikipedia]

- "Lasso" is an acronym for "least absolute shrinkage and selection operator".

  "Ridge" is a metaphor describing plains, shaped like a chain of hills, the highest point of which is not easily to recognize.

- $||\cdot||_k$ denotes the vector norm operator:

$$||\mathbf{x}||_k \equiv \left(\sum_{j=1}^{p} |x_j|^k \right)^{1/k},$$

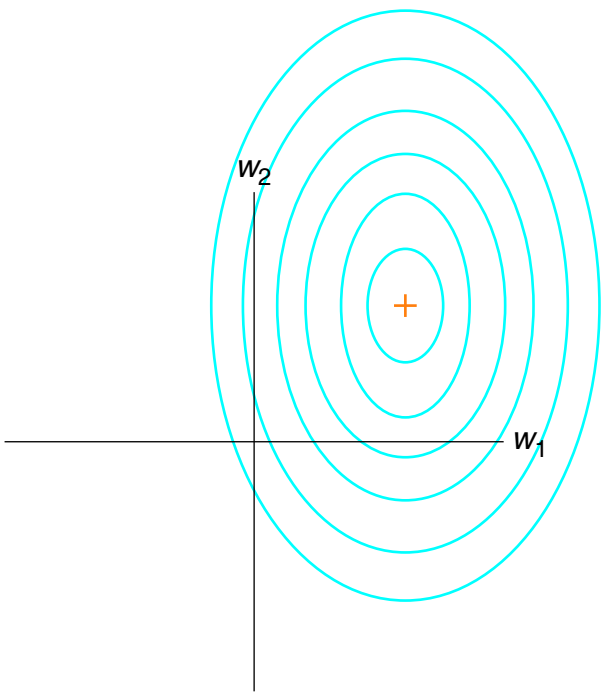
  where $k \in [1, \infty)$ and p is the dimensionality of vector $\mathbf{x}$.

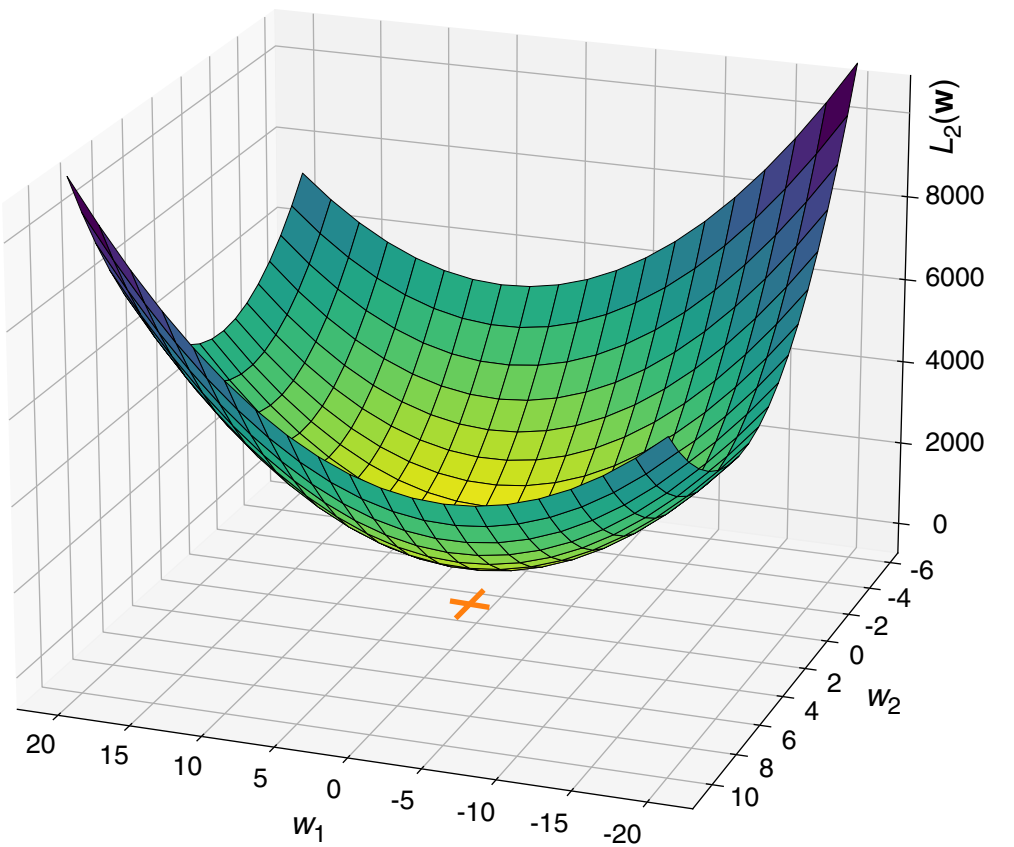
- By convention, $||\cdot||$ (omitting the subscript) refers to the Euclidean norm ($k = 2$).

Regularization

The Vector Norm as Regularization Function (continued)

$$\mathcal{L}(\mathbf{w}) \;=\; L(\mathbf{w})$$

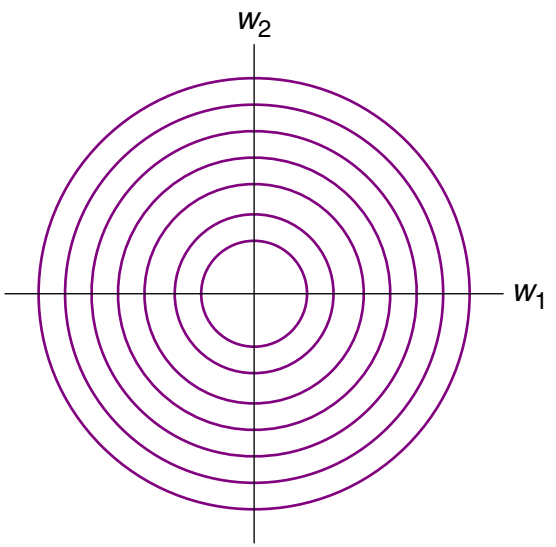


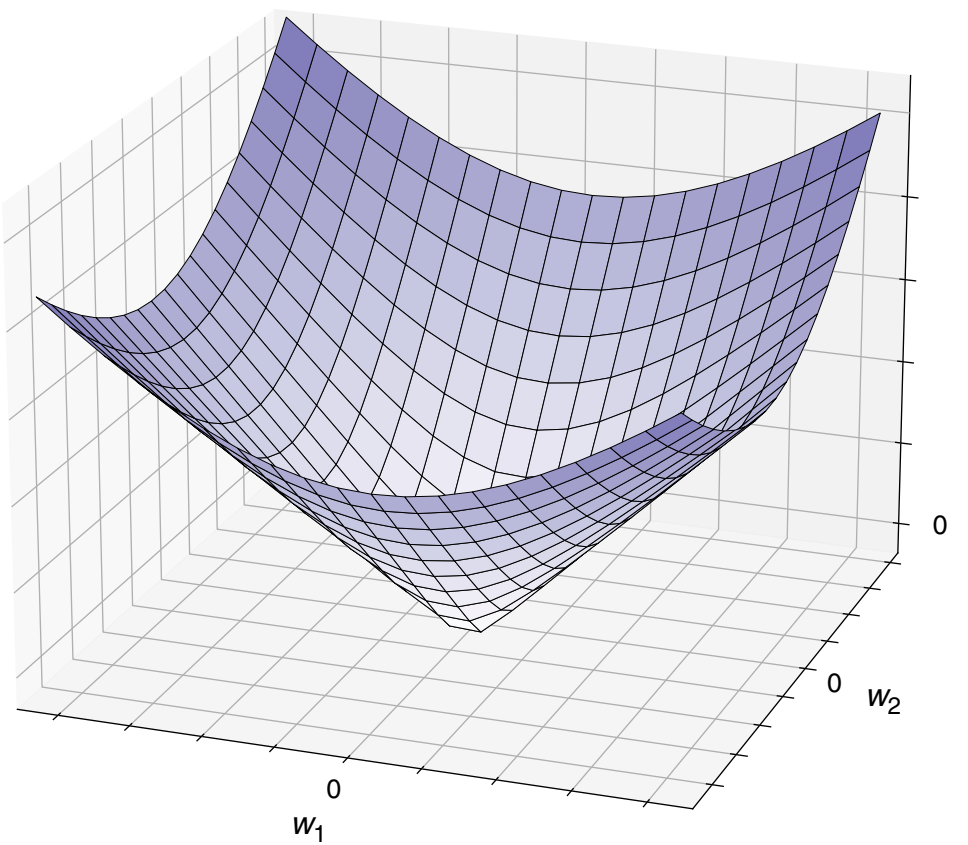


Regularization

The Vector Norm as Regularization Function (continued)

$$R_{||\vec{\mathbf{w}}||_2^2}(\mathbf{w}) = \sum_{i=1}^{p} w_i^2 = \vec{\mathbf{w}}^T \vec{\mathbf{w}}$$

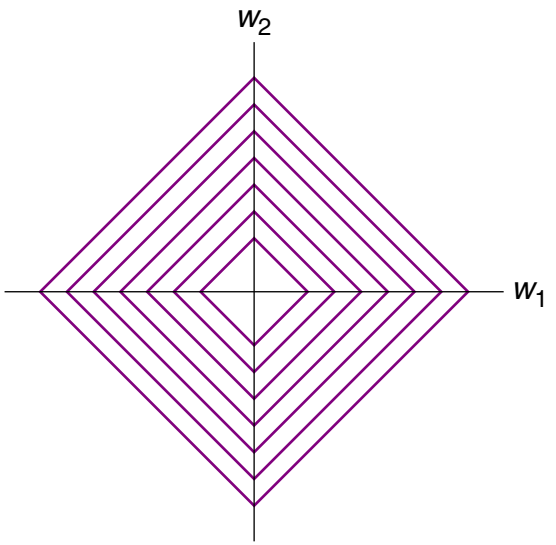


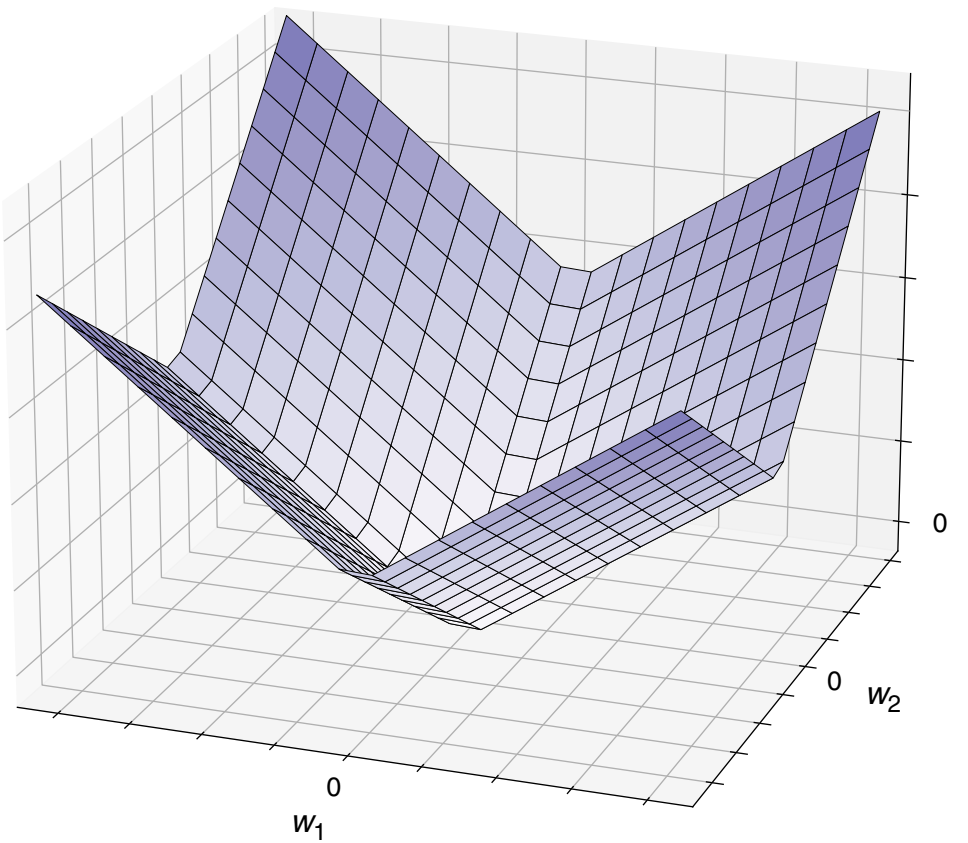


Regularization

The Vector Norm as Regularization Function (continued)

$$R_{||\vec{\mathbf{w}}||_1}(\mathbf{w}) = \sum_{i=1}^{p} |w_i|$$





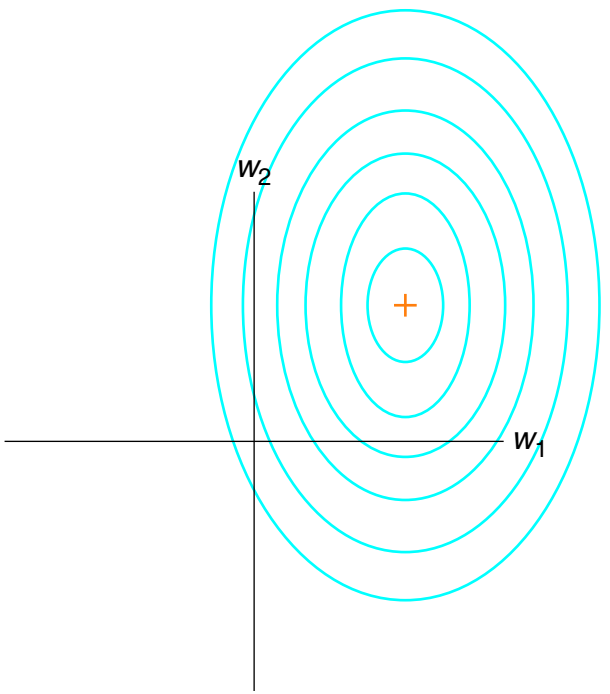
Remarks:

- The exemplified plots of the loss term, $L(\mathbf{w})$, and the regularization term, $R(\mathbf{w})$, are illustrated over the parameter space $\{(w_1, w_2) \mid w_i \in \mathbf{R}\}$ (instead of $\{(w_0, w_1) \mid w_i \in \mathbf{R}\}$) to better emphasize the characteristic difference between ridge regression and lasso regression.
- The contour line plots show two-dimensional projections of the three-dimensional convex loss function (here: RSS) for a given set of example data, as well as of the two regularization functions $R_{||\mathbf{w}||_2^2}$ and $R_{||\mathbf{w}||_1}$, whose shapes do not depend on the data.
- A contour line is a curve along which the respective function has a constant value.

Regularization

The Vector Norm as Regularization Function (continued)

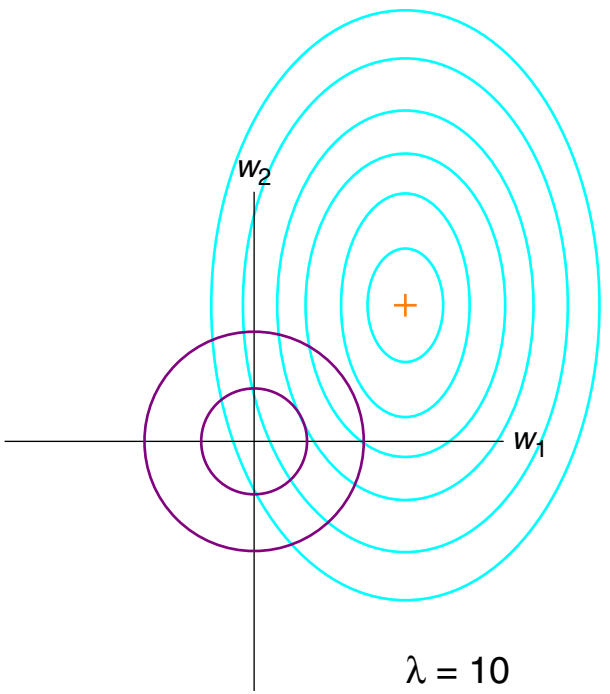
$$\mathcal{L}(\mathbf{w}) = L(\mathbf{w})$$



Regularization

The Vector Norm as Regularization Function (continued)

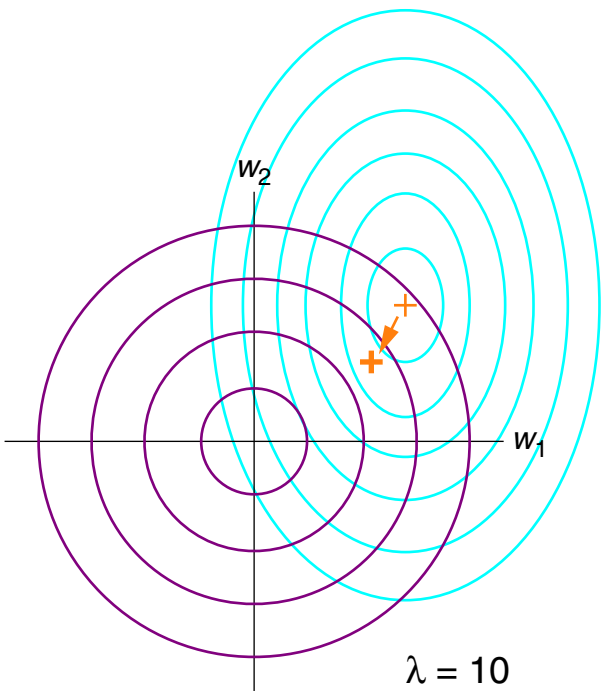
$$\mathcal{L}(\mathbf{w}) = L(\mathbf{w}) + \lambda \cdot R_{||\vec{\mathbf{w}}||_2^2}(\mathbf{w})$$



Regularization

The Vector Norm as Regularization Function (continued)

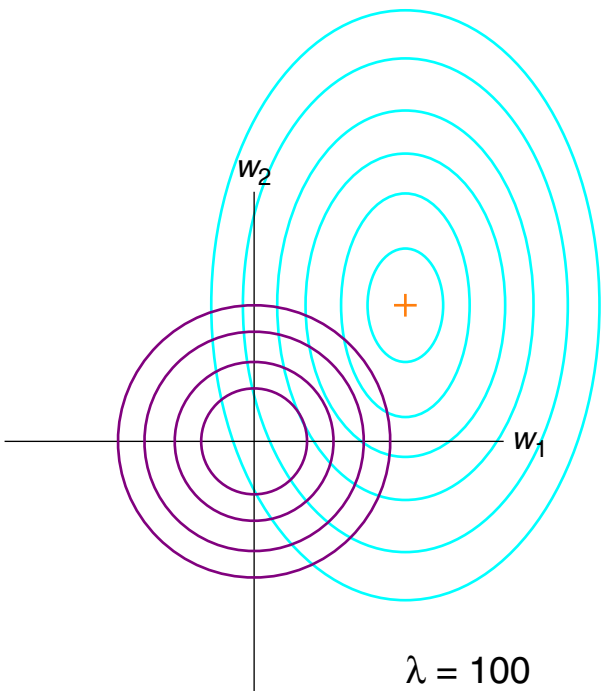
$$\mathcal{L}(\mathbf{w}) = L(\mathbf{w}) + \lambda \cdot R_{||\vec{\mathbf{w}}||_2^2}(\mathbf{w})$$



Regularization

The Vector Norm as Regularization Function (continued)

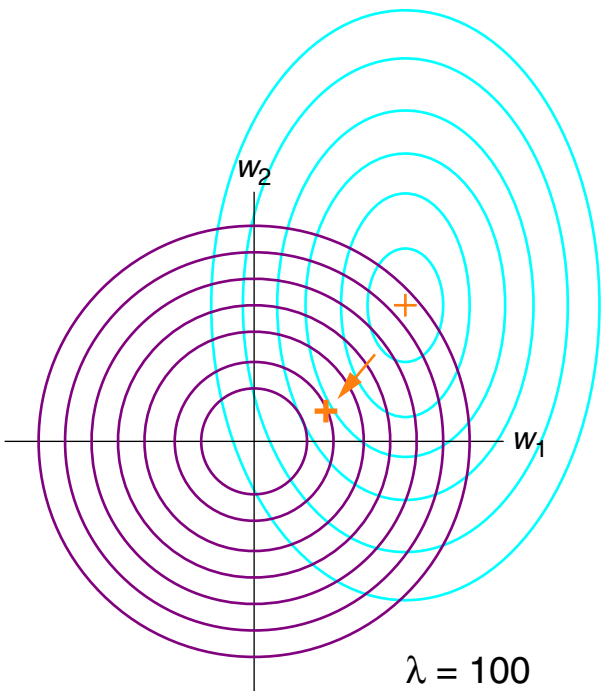
$$\mathcal{L}(\mathbf{w}) = L(\mathbf{w}) + \lambda \cdot R_{||\vec{\mathbf{w}}||_2^2}(\mathbf{w})$$



Regularization

The Vector Norm as Regularization Function (continued)

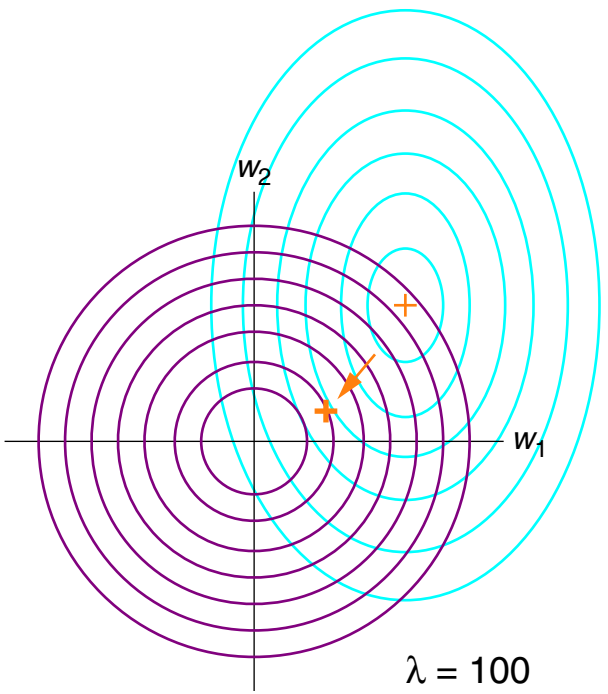
$$\mathcal{L}(\mathbf{w}) = L(\mathbf{w}) + \lambda \cdot R_{||\vec{\mathbf{w}}||_2^2}(\mathbf{w})$$



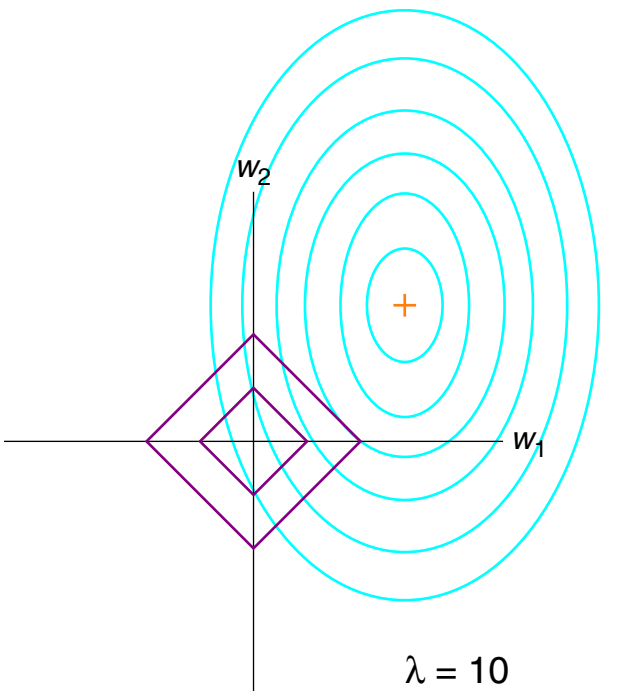
Regularization

The Vector Norm as Regularization Function (continued)

$$\mathcal{L}(\mathbf{w}) = L(\mathbf{w}) + \lambda \cdot R_{||\vec{\mathbf{w}}||_2^2}(\mathbf{w})$$



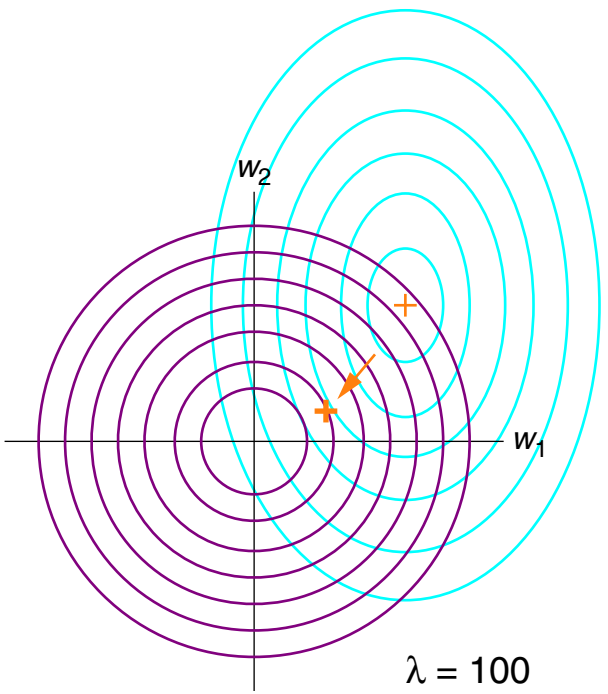
$$\mathcal{L}(\mathbf{w}) = L(\mathbf{w}) + \lambda \cdot R_{||\vec{\mathbf{w}}||_1}(\mathbf{w})$$



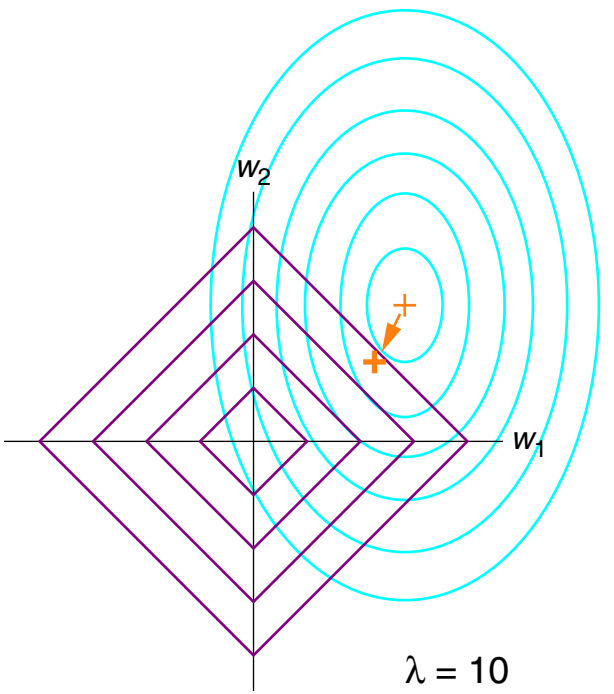
Regularization

The Vector Norm as Regularization Function (continued)

$$\mathcal{L}(\mathbf{w}) = L(\mathbf{w}) + \lambda \cdot R_{||\vec{\mathbf{w}}||_2^2}(\mathbf{w})$$



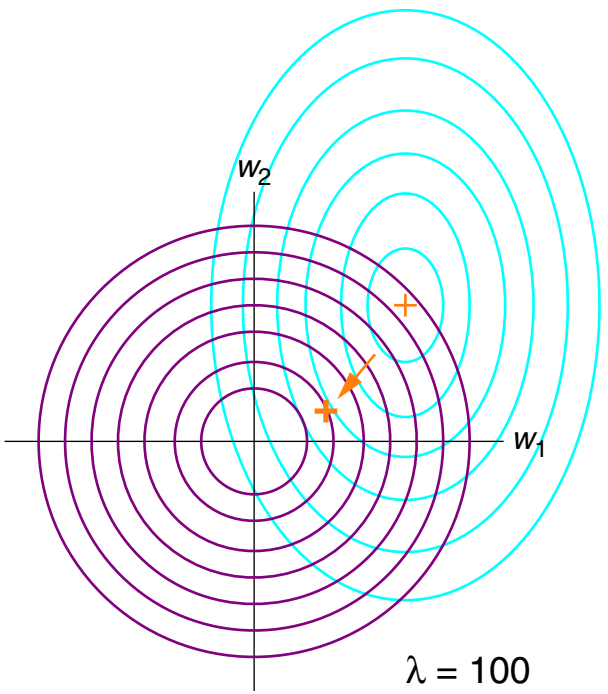
$$\mathcal{L}(\mathbf{w}) = L(\mathbf{w}) + \lambda \cdot R_{||\vec{\mathbf{w}}||_1}(\mathbf{w})$$



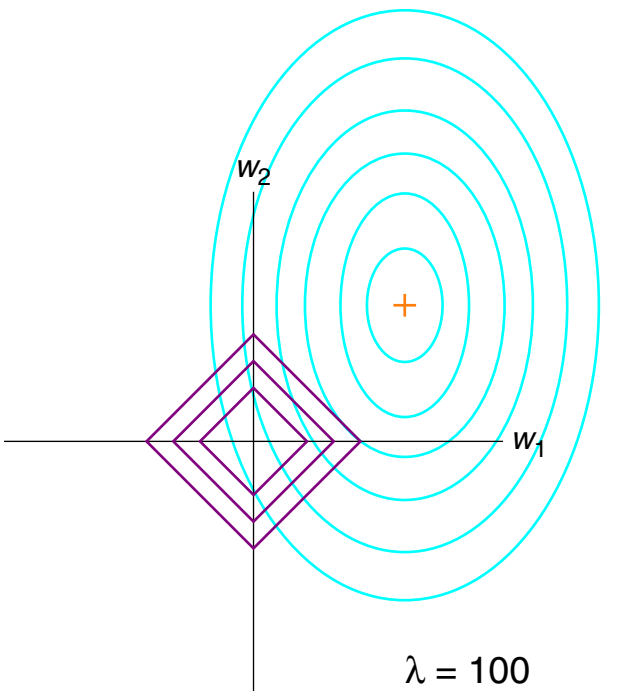
Regularization

The Vector Norm as Regularization Function (continued)

$$\mathcal{L}(\mathbf{w}) = L(\mathbf{w}) + \lambda \cdot R_{||\vec{\mathbf{w}}||_2^2}(\mathbf{w})$$



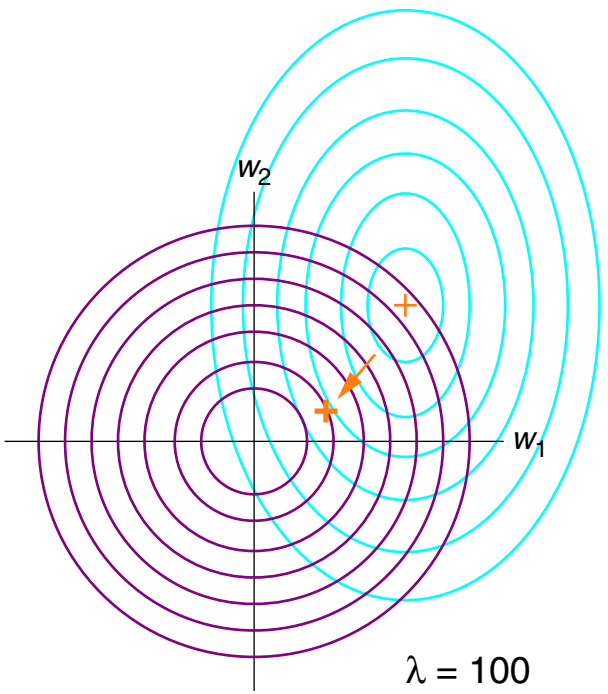
$$\mathcal{L}(\mathbf{w}) = L(\mathbf{w}) + \lambda \cdot R_{||\vec{\mathbf{w}}||_1}(\mathbf{w})$$



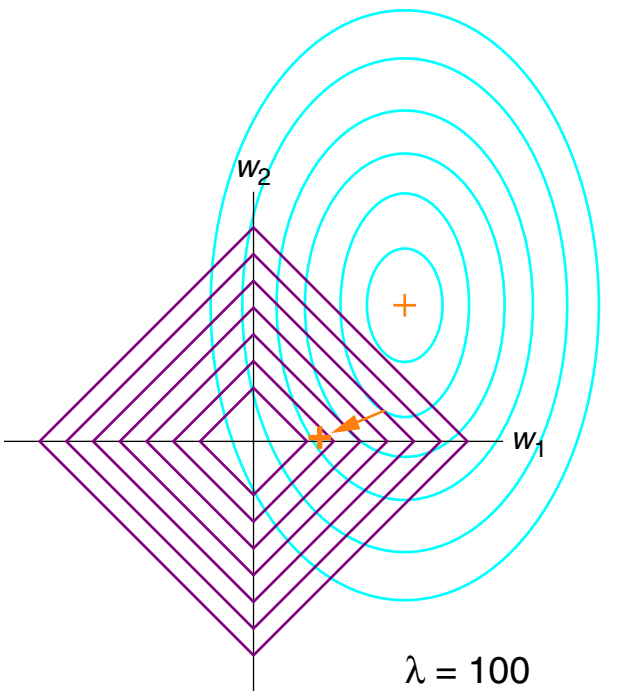
Regularization

The Vector Norm as Regularization Function (continued)

$$\mathcal{L}(\mathbf{w}) = L(\mathbf{w}) + \lambda \cdot R_{||\vec{\mathbf{w}}||_2^2}(\mathbf{w})$$



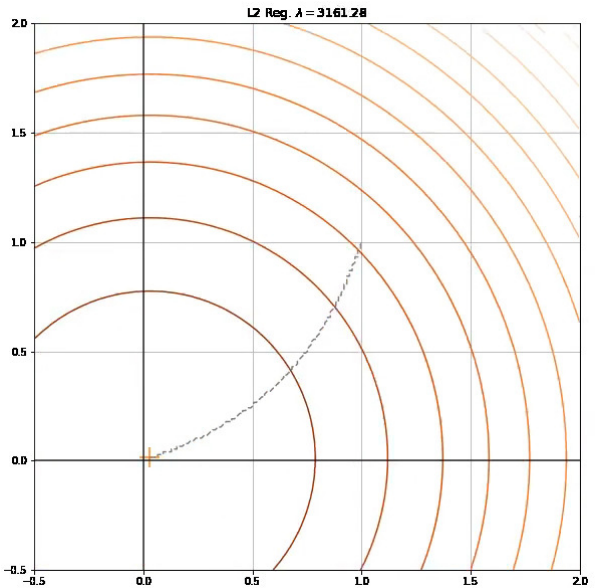
$$\mathcal{L}(\mathbf{w}) = L(\mathbf{w}) + \lambda \cdot R_{||\vec{\mathbf{w}}||_1}(\mathbf{w})$$



Regularization

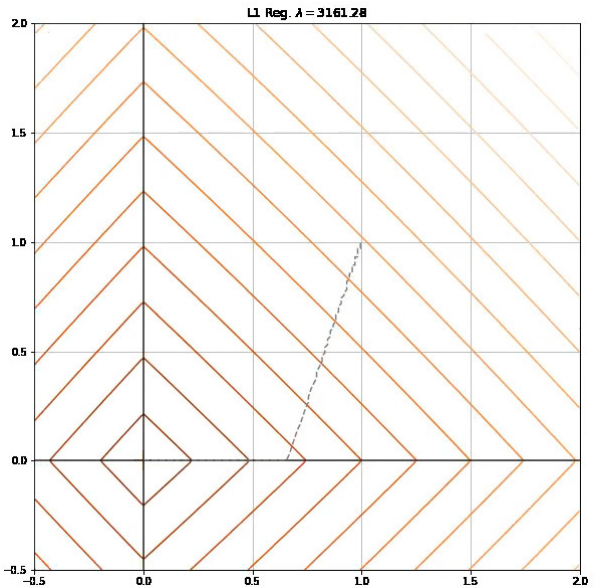
The Vector Norm as Regularization Function (continued)

$$\mathcal{L}(\mathbf{w}) = L(\mathbf{w}) + \lambda \cdot R_{||\vec{\mathbf{w}}||_2^2}(\mathbf{w})$$



[animation]

$$\mathcal{L}(\mathbf{w}) = L(\mathbf{w}) + \lambda \cdot R_{||\vec{\mathbf{w}}||_1}(\mathbf{w})$$



[animation]

The animations show superimposed contourlines. The choice of R determines the trajectory the minimum takes towards the origin as a function of λ. [stackexchange]

Remarks:

- The exemplified loss function is minimal at the cross. Without regularization, the weights associated with the minimum will be the result of a linear regression. By adding the regularization term $\lambda \cdot R(\mathbf{w})$ with $\lambda > 0$, the joint minimum of the two functions is found closer to the origin of the parameter space than the minimum of the loss function.
- The choice of λ determines how much closer the joint minimum is to the origin of the parameter space; the higher, the closer, and thus the smaller the parameters $\mathbf{w}$.
- The minimum of $\mathcal{L}(\mathbf{w})$ is on a tangent point between a contour line of $L(\mathbf{w})$ and a contour line of $R(\mathbf{w})$. Barring exceptional cases, the minimum of $\mathcal{L}(\mathbf{w})$ (the sum of global loss and regularization) is unique, even if the minimum of $L(\mathbf{w})$ (the global loss) is non-unique.
- A key difference between ridge ($R_{||\vec{\mathbf{w}}||_2^2}$) and lasso ($R_{||\vec{\mathbf{w}}||_1}$) regression is that, with lasso regression, parameters can be reduced to zero, eliminating the corresponding feature from the model function.

  With ridge regression, the influence of all parameters will be reduced "uniformly." In particular, a parameter will be reduced to zero if and only if the minimum of the loss function is found on that parameter's axis.

Regularization

Regularized Linear Regression [linear regression]

- Given $\mathbf{x}$, predict a real-valued output under a linear model function:

$$y(\mathbf{x}) = w_0 + \sum_{j=1}^{p} w_j \cdot x_j$$

- Vector notation with $x_0 = 1$ and $\mathbf{w} = (w_0, w_1, \ldots, w_p)^T$:

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

Regularization

Regularized Linear Regression (continued) [linear regression]

- Given $\mathbf{x}$, predict a real-valued output under a linear model function:

$$y(\mathbf{x}) = w_0 + \sum_{j=1}^{p} w_j \cdot x_j$$

- Vector notation with $x_0 = 1$ and $\mathbf{w} = (w_0, w_1, \ldots, w_p)^T$:

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

- Given $\mathbf{x}_1, \ldots, \mathbf{x}_n$, assess goodness of fit of the objective function:

$$\mathcal{L}(\mathbf{w}) \;=\; \mathsf{RSS}(\mathbf{w}) + \lambda \cdot R_{||\vec{\mathbf{w}}||_\mathbf{2}^\mathbf{2}}(\mathbf{w}) \;=\; \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \cdot \vec{\mathbf{w}}^T \vec{\mathbf{w}} \qquad (1)$$

Regularization

Regularized Linear Regression (continued) [linear regression]

- Given $\mathbf{x}$, predict a real-valued output under a linear model function:

$$y(\mathbf{x}) = w_0 + \sum_{j=1}^{p} w_j \cdot x_j$$

- Vector notation with $x_0 = 1$ and $\mathbf{w} = (w_0, w_1, \ldots, w_p)^T$:

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

- Given $\mathbf{x}_1, \ldots, \mathbf{x}_n$, assess goodness of fit of the objective function:

$$\mathcal{L}(\mathbf{w}) \;=\; \text{RSS}(\mathbf{w}) + \lambda \cdot R_{||\vec{\mathbf{w}}||_2^2}(\mathbf{w}) \;=\; \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \cdot \vec{\mathbf{w}}^T \vec{\mathbf{w}} \tag{1}$$

- Estimate optimum $\mathbf{w}$ by minimizing the residual sum of squares:

$$\hat{\mathbf{w}} \;=\; \underset{\mathbf{w} \in \mathbf{R}^{p+1}}{\text{argmin}} \; \mathcal{L}(\mathbf{w}) \tag{2}$$

Regularization

Regularized Linear Regression (continued) [linear regression]

- Let X denote the $n \times (p+1)$ matrix, where row i is $(1 \;\; \mathbf{x}_i^T)$ with $(\mathbf{x}_i, y_i) \in D$.

  Let $\mathbf{y}$ denote the n-vector of outputs in the training set D.

$\rightsquigarrow \quad \mathcal{L}(\mathbf{w}) \;=\; (\mathbf{y} - X\mathbf{w})^T(\mathbf{y} - X\mathbf{w}) \;+\; \lambda \cdot \vec{\mathbf{w}}^T\vec{\mathbf{w}}$

Regularization

Regularized Linear Regression (continued) [linear regression]

- Let X denote the $n \times (p+1)$ matrix, where row i is $(1 \;\; \mathbf{x}_i^T)$ with $(\mathbf{x}_i, y_i) \in D$.

  Let $\mathbf{y}$ denote the n-vector of outputs in the training set D.

$$\rightsquigarrow \quad \mathcal{L}(\mathbf{w}) \;=\; (\mathbf{y} - X\mathbf{w})^T(\mathbf{y} - X\mathbf{w}) \;+\; \lambda \cdot \vec{\mathbf{w}}^T\vec{\mathbf{w}}$$

- Minimize $\mathcal{L}(\mathbf{w})$ via a direct method:

$$\begin{aligned}
\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} &= -2X^T(\mathbf{y} - X\mathbf{w}) \;+\; 2\lambda \cdot \begin{pmatrix} 0 \\ \vec{\mathbf{w}} \end{pmatrix} \;=\; 0 \\
X^T(\mathbf{y} - X\mathbf{w}) - \lambda \cdot \begin{pmatrix} 0 \\ \vec{\mathbf{w}} \end{pmatrix} &= 0 \\
\Leftrightarrow \quad \left(X^TX + \lambda \cdot \mathrm{diag}(0, 1, \ldots, 1)\right) \mathbf{w} &= X^T\mathbf{y} \\
\Leftrightarrow \quad \mathbf{w} &= \left(X^TX + \mathrm{diag}(0, \lambda, \ldots, \lambda) \right)^{-1} X^T\mathbf{y}
\end{aligned}$$

Regularization

Regularized Linear Regression (continued) [linear regression]

- Let X denote the $n \times (p+1)$ matrix, where row i is $(1 \;\; \mathbf{x}_i^T)$ with $(\mathbf{x}_i, y_i) \in D$.

  Let $\mathbf{y}$ denote the n-vector of outputs in the training set D.

$$\rightsquigarrow \quad \mathcal{L}(\mathbf{w}) \;=\; (\mathbf{y} - X\mathbf{w})^T(\mathbf{y} - X\mathbf{w}) \;+\; \lambda \cdot \vec{\mathbf{w}}^T\vec{\mathbf{w}}$$

- Minimize $\mathcal{L}(\mathbf{w})$ via a direct method:

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} \;=\; -2X^T(\mathbf{y} - X\mathbf{w}) \;+\; 2\lambda \cdot \begin{pmatrix} 0 \\ \vec{\mathbf{w}} \end{pmatrix} \;=\; 0$$

$$X^T(\mathbf{y} - X\mathbf{w}) - \lambda \cdot \begin{pmatrix} 0 \\ \vec{\mathbf{w}} \end{pmatrix} \;=\; 0$$

$$\Leftrightarrow \quad \left(X^TX + \lambda \cdot \mathrm{diag}(0, 1, \ldots, 1)\right) \mathbf{w} \;=\; X^T\mathbf{y} \qquad \text{Normal equations.}$$

$$\Leftrightarrow \quad \mathbf{w} \;=\; \Big(\underbrace{X^TX + \mathrm{diag}(0, \lambda, \ldots, \lambda)}_{\text{Conditioning the moment matrix } X^TX \quad \text{[Wikipedia 1, 2, 3]}} \Big)^{-1} X^T\mathbf{y} \quad \text{If } \lambda > 0.$$

Regularization

Regularized Linear Regression (continued) [linear regression]

- Let X denote the $n \times (p+1)$ matrix, where row i is $(1 \;\; \mathbf{x}_i^T)$ with $(\mathbf{x}_i, y_i) \in D$.

  Let $\mathbf{y}$ denote the n-vector of outputs in the training set D.

$$\rightsquigarrow \quad \mathcal{L}(\mathbf{w}) \;=\; (\mathbf{y} - X\mathbf{w})^T(\mathbf{y} - X\mathbf{w}) \;+\; \lambda \cdot \vec{\mathbf{w}}^T\vec{\mathbf{w}}$$

- Minimize $\mathcal{L}(\mathbf{w})$ via a direct method:

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} \;=\; -2X^T(\mathbf{y} - X\mathbf{w}) \;+\; 2\lambda \cdot \begin{pmatrix} 0 \\ \vec{\mathbf{w}} \end{pmatrix} \;=\; 0$$

$$X^T(\mathbf{y} - X\mathbf{w}) - \lambda \cdot \begin{pmatrix} 0 \\ \vec{\mathbf{w}} \end{pmatrix} \;=\; 0$$

$$\Leftrightarrow \quad \left(X^TX + \lambda \cdot \mathrm{diag}(0, 1, \ldots, 1)\right) \mathbf{w} \;=\; X^T\mathbf{y} \qquad \text{Normal equations.}$$

$$\Leftrightarrow \quad \hat{\mathbf{w}} \equiv \mathbf{w} \;=\; \underbrace{\left(X^TX + \mathrm{diag}(0, \lambda, \ldots, \lambda) \right)^{-1}}_{\text{Conditioning the moment matrix } X^TX \quad \text{[Wikipedia 1, 2, 3]}} X^T\mathbf{y} \qquad \text{If } \lambda > 0.$$

$$\hat{y}(\mathbf{x}_i) \;=\; \hat{\mathbf{w}}^T\, \mathbf{x}_i \qquad \text{Regression function with least squares estimator } \hat{\mathbf{w}}.$$

Chapter ML:III (continued)

III. Linear Models

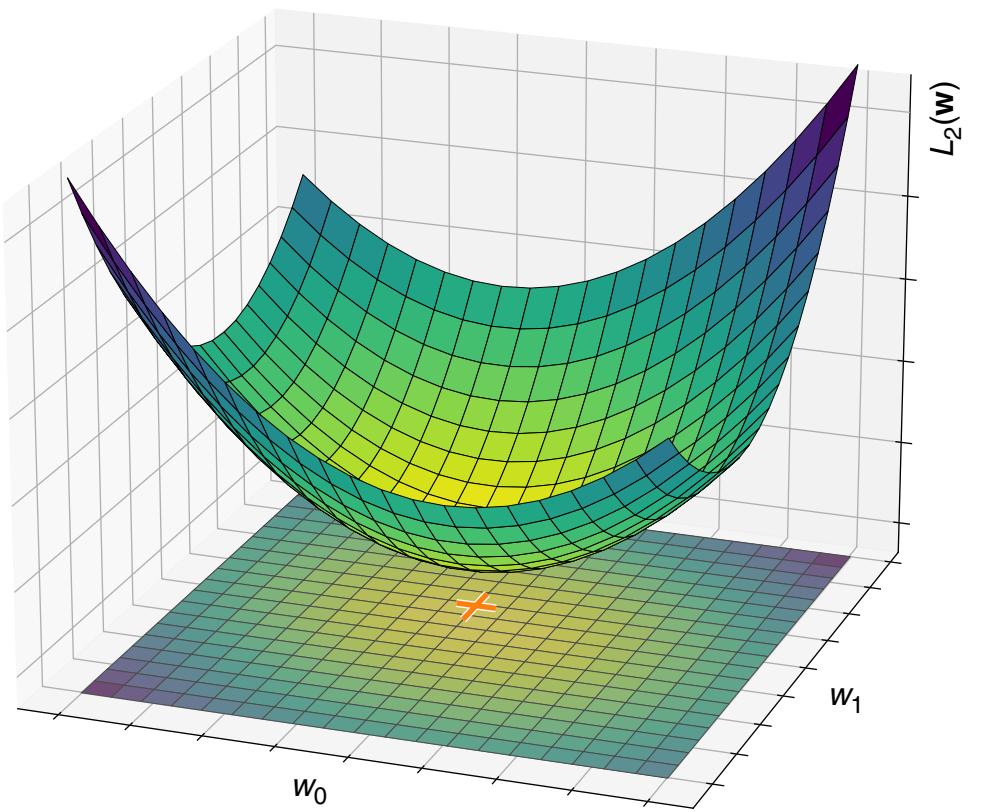
- Logistic Regression
- Loss Computation in Detail
- Overfitting
- Regularization
- Gradient Descent in Detail

Gradient Descent in Detail

Principle

Gradient descent, GD, is a first-order iterative optimization algorithm for finding a local extremum of a differentiable function f. [Wikipedia]

In our algorithms, f is the global loss function, L, or some objective function, $\mathcal{L}$.

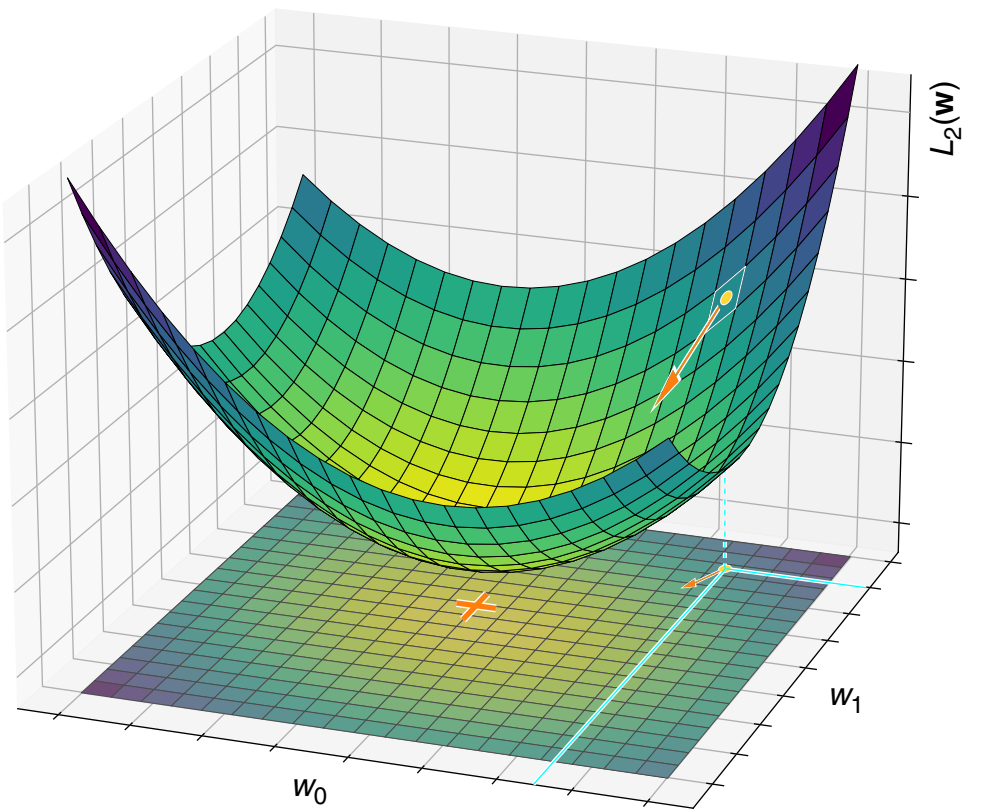


Gradient Descent in Detail

Principle (continued)

Gradient descent, GD, is a first-order iterative optimization algorithm for finding a local extremum of a differentiable function f. [Wikipedia]

In our algorithms, f is the global loss function, L, or some objective function, $\mathcal{L}$.

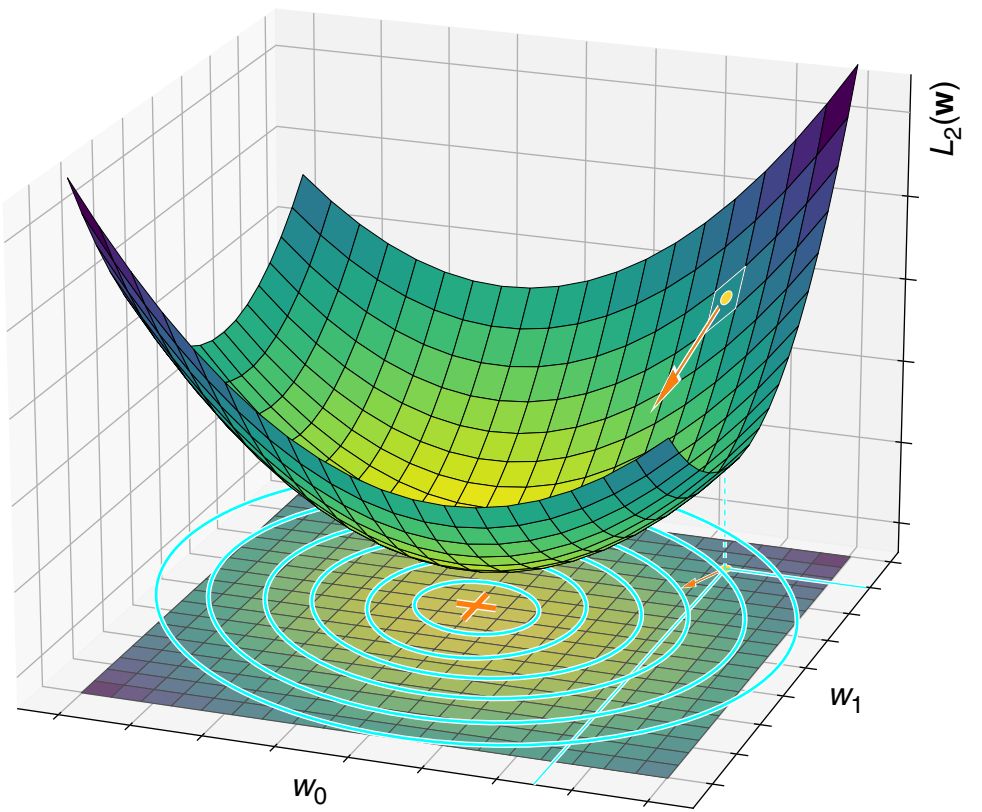


Gradient Descent in Detail

Principle (continued)

Gradient descent, GD, is a first-order iterative optimization algorithm for finding a local extremum of a differentiable function f. [Wikipedia]

In our algorithms, f is the global loss function, L, or some objective function, $\mathcal{L}$.



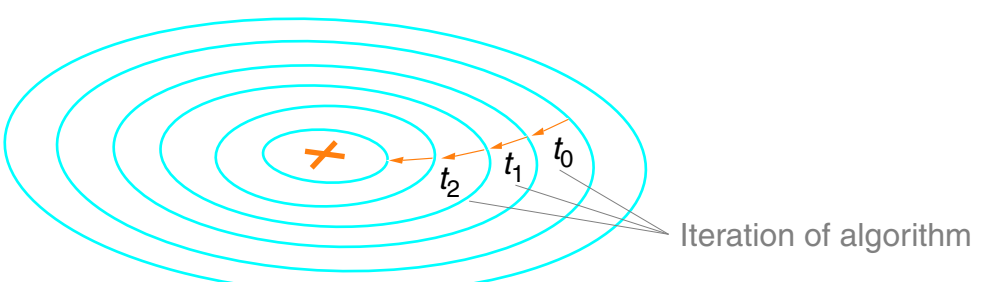
Gradient Descent in Detail

Principle (continued)

Gradient descent, GD, is a first-order iterative optimization algorithm for finding a local extremum of a differentiable function f. [Wikipedia]

In our algorithms, f is the global loss function, L, or some objective function, $\mathcal{L}$.

- The gradient ∇f of a differentiable function f of several variables is a vector whose components are the partial derivatives of f. (simplified definition)
- The gradient of a function is the direction of steepest ascent or descent.
- Gradient *ascent* means stepping in the direction of the gradient.
- Likewise, gradient *descent* means stepping in the opposite direction of the gradient; it will lead to a local minimum of that function.



Remarks:

- In machine learning the GD principle is applied to take the direction of steepest descent for various loss and objective functions. In this section we will discuss gradient descent in the following hypothesis search settings:

  (1) linear regression + squared loss

  (2) linear regression + 0/1 loss

  (3) logistic regression + logistic loss + regularization

  In section Multilayer Perceptron of part Neural Networks the GD principle is applied in the form of the backpropagation mechanism to tackle search settings with unconstrained hypotheses forms:

  (4) multilayer perceptron with single hidden layer and k-dimensional output + squared loss

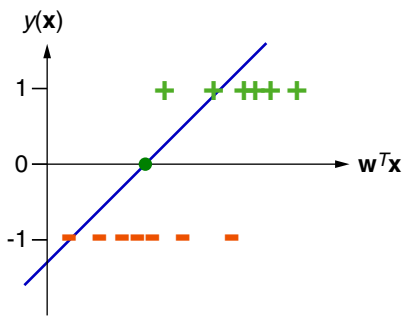
  (5) multilayer perceptron with d hidden layers and k-dimensional output + squared loss

- Recall that by the method of steepest (= gradient) descent the determination of the global optimum can be guaranteed for convex functions only.

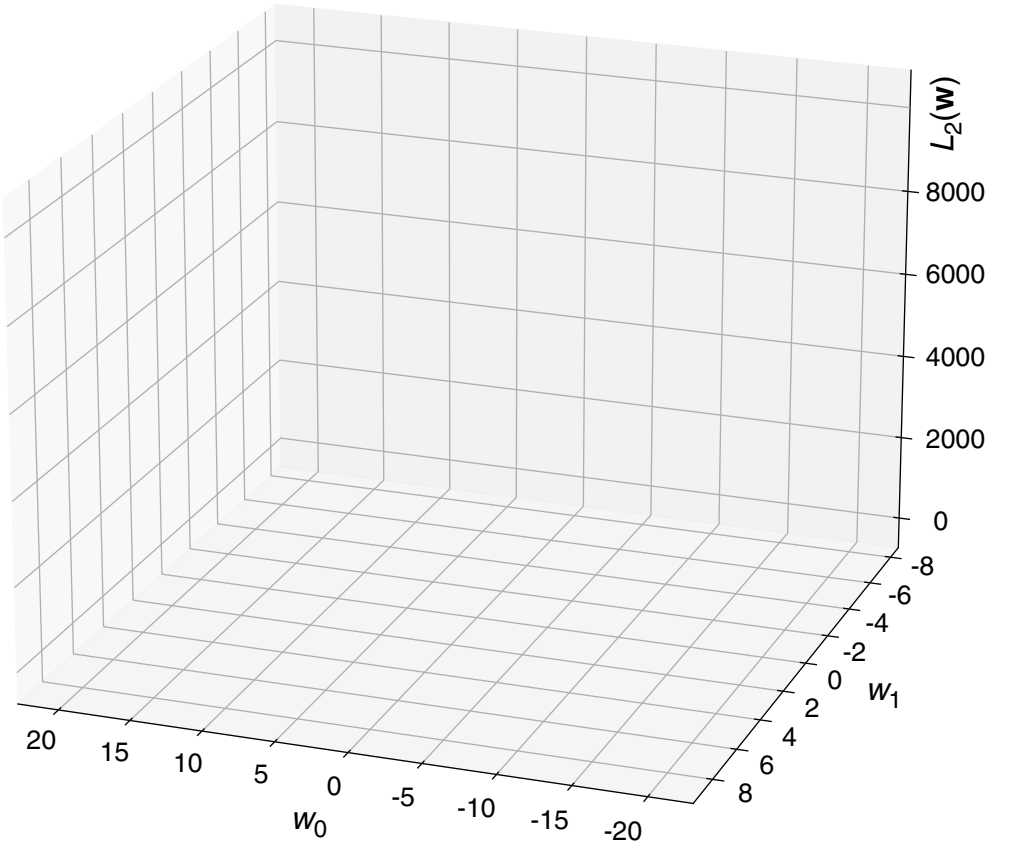
  The (loss or objective) functions considered in the settings (1) and (3) are convex; the (loss or objective) functions considered in the settings (2), (4), and (5) are typically non-convex.

Gradient Descent in Detail

(1) Linear Regression + Squared Loss

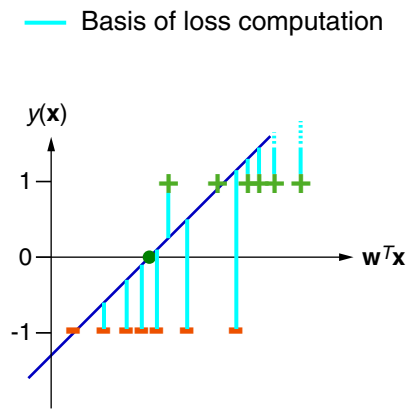


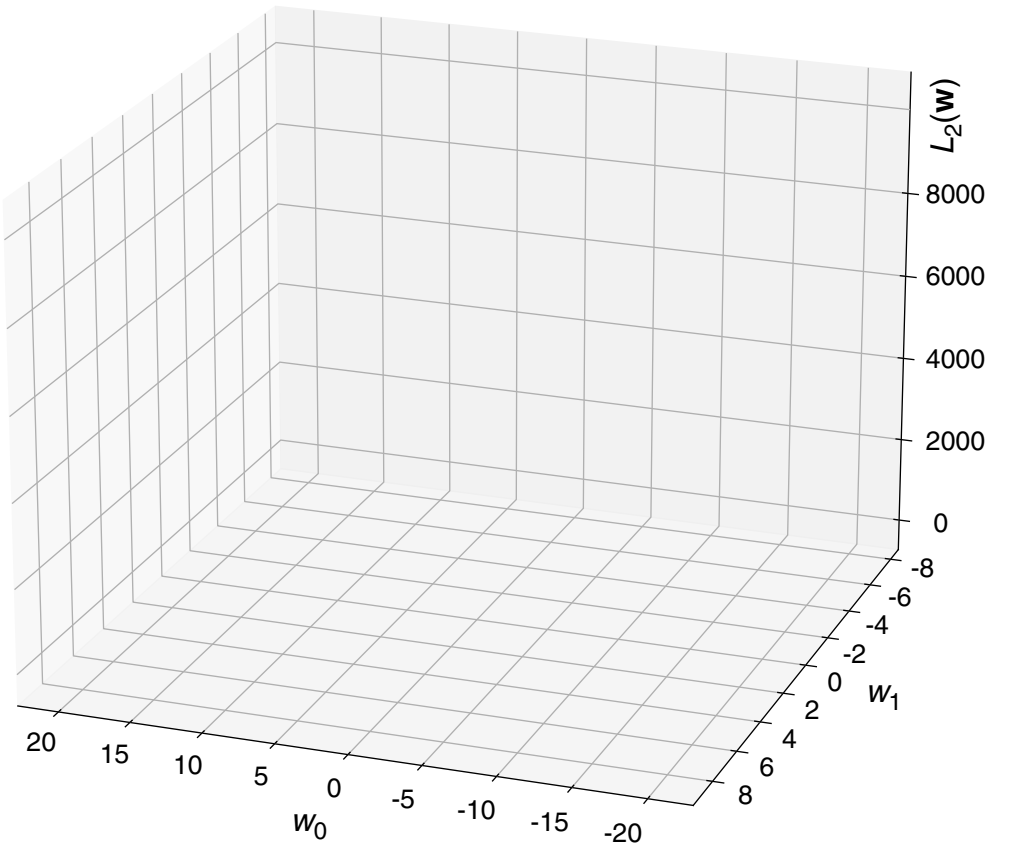
$$y(\mathbf{x}) \overset{(\star)}{=} \mathbf{w}^T\mathbf{x}$$



Gradient Descent in Detail

(1) Linear Regression + Squared Loss (continued)





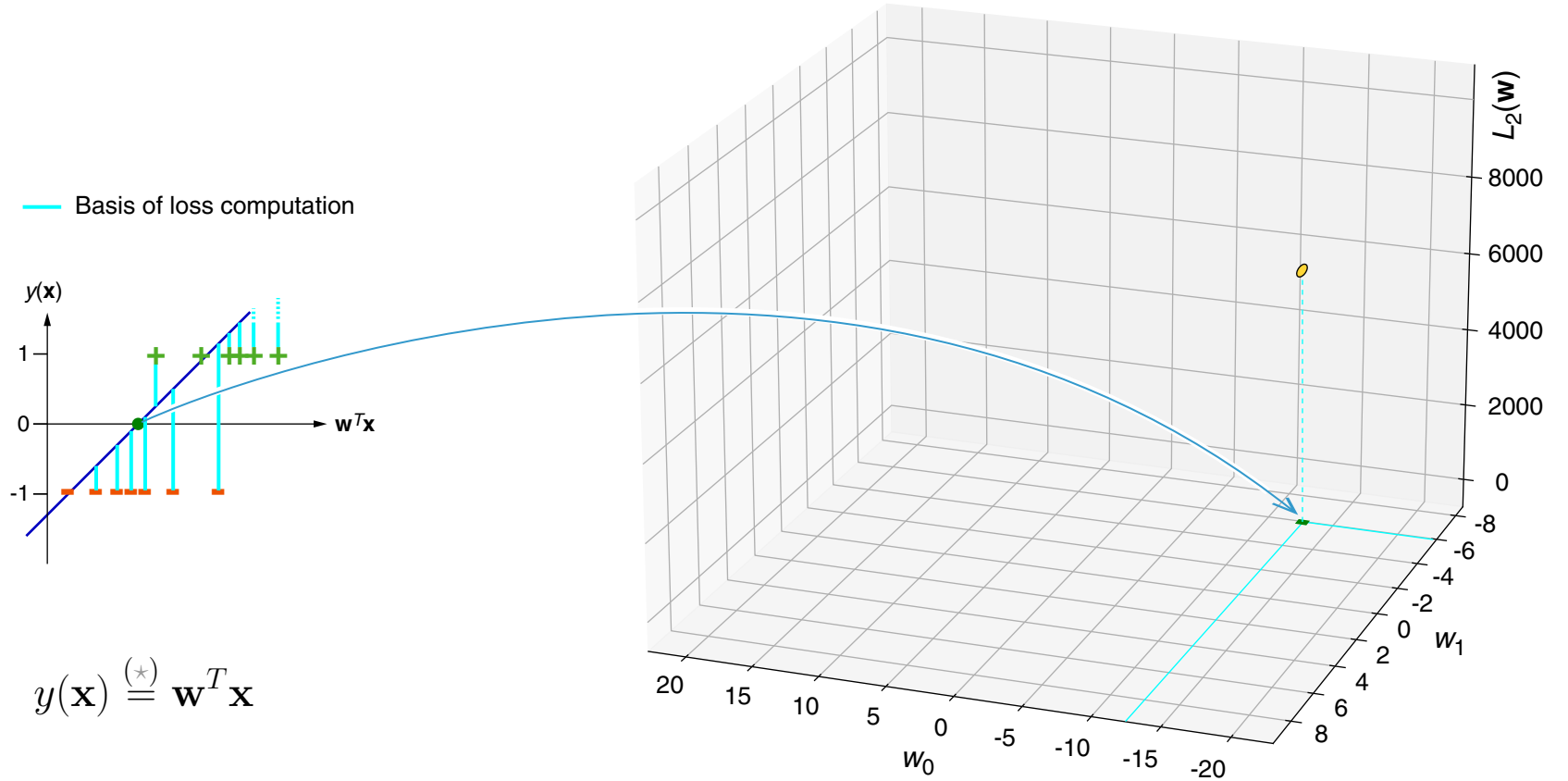
$$y(\mathbf{x}) \overset{(\star)}{=} \mathbf{w}^T\mathbf{x}$$

$$L_2(\mathbf{w}) = \frac{1}{2}\cdot \mathsf{RSS}(\mathbf{w}) = \frac{1}{2}\cdot\sum_{(\mathbf{x},c)\in D}(c - y(\mathbf{x}))^2$$

[pointwise squared loss]

Gradient Descent in Detail

(1) Linear Regression + Squared Loss (continued)



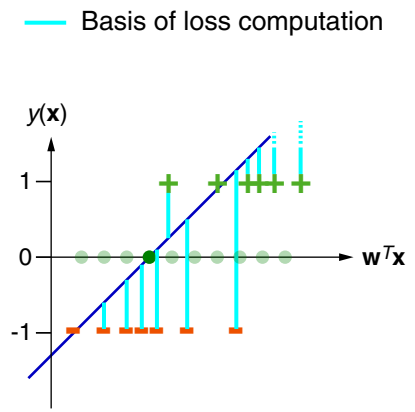
$$y(\mathbf{x}) \overset{(\star)}{=} \mathbf{w}^T\mathbf{x}$$

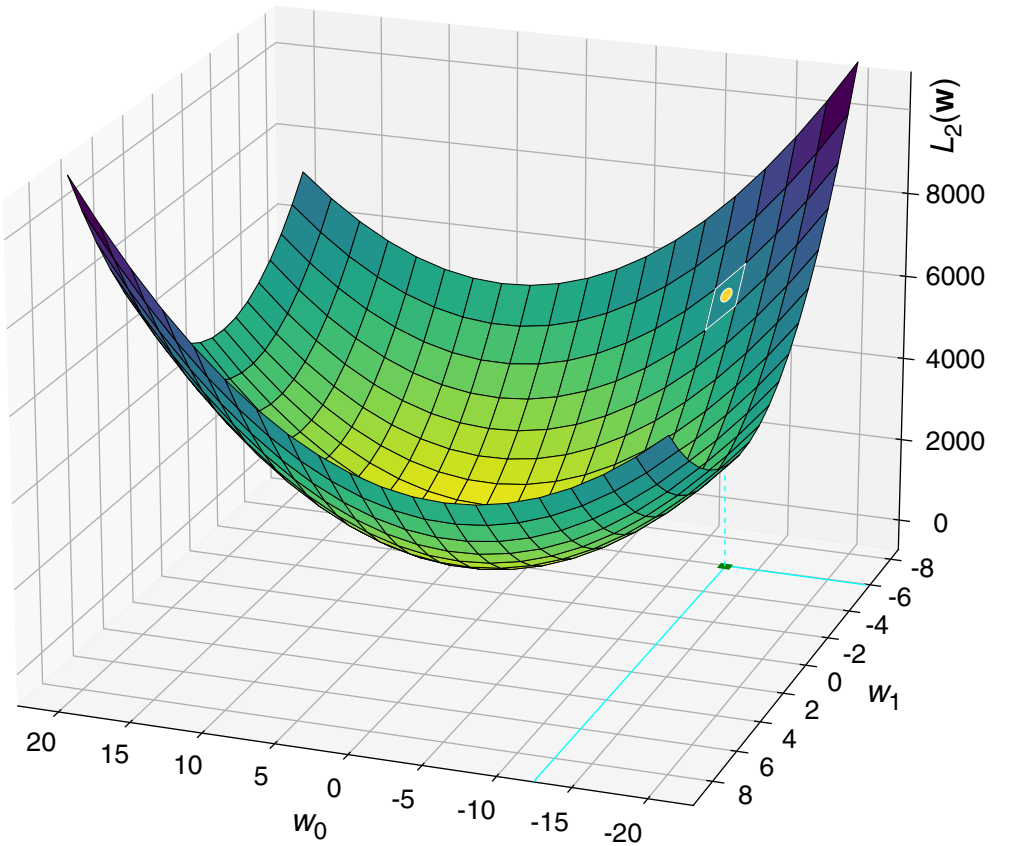
$$L_2(\mathbf{w}) = \frac{1}{2}\cdot \mathsf{RSS}(\mathbf{w}) = \frac{1}{2}\cdot\sum_{(\mathbf{x},c)\in D}(c - y(\mathbf{x}))^2$$

[pointwise squared loss]

Gradient Descent in Detail

(1) Linear Regression + Squared Loss (continued)





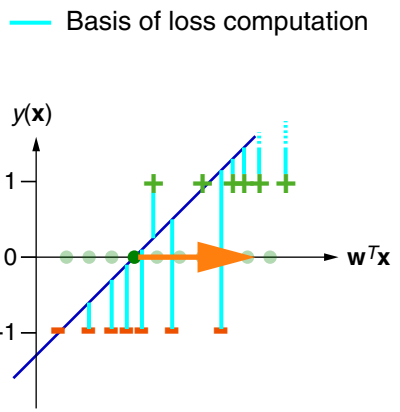
$$y(\mathbf{x}) \overset{(\star)}{=} \mathbf{w}^T\mathbf{x}$$

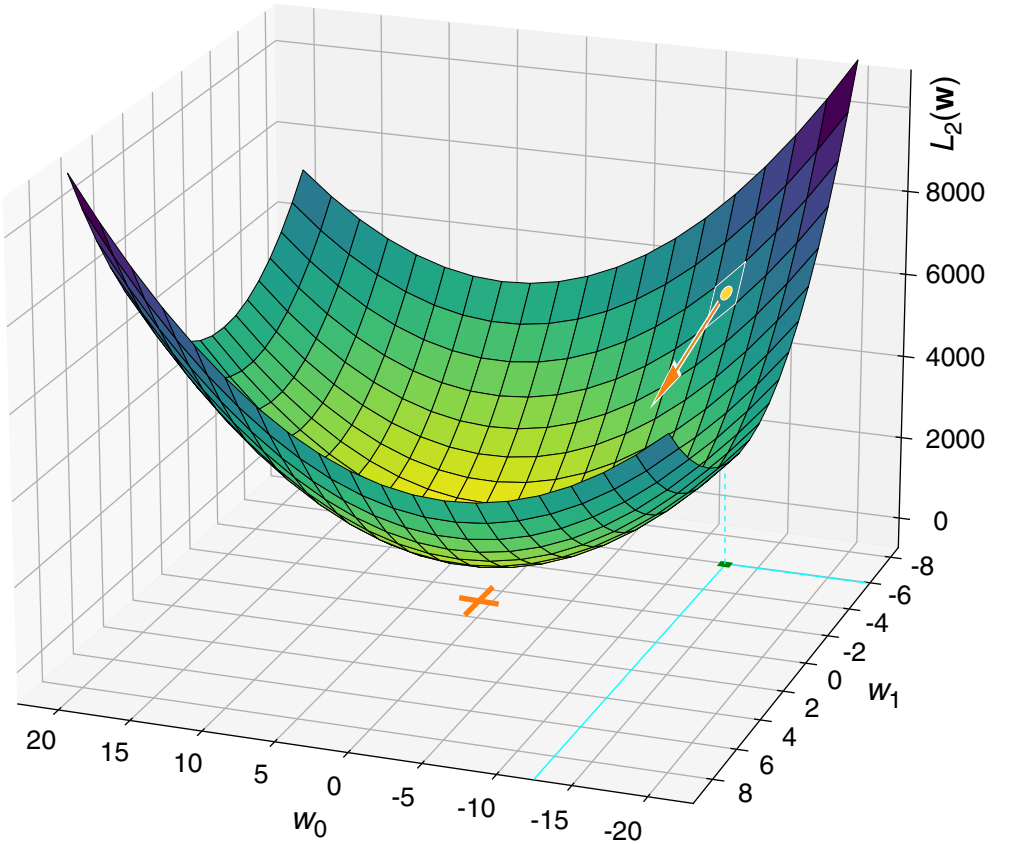
$$L_2(\mathbf{w}) = \frac{1}{2}\cdot \mathsf{RSS}(\mathbf{w}) = \frac{1}{2}\cdot\sum_{(\mathbf{x},c)\in D}(c - y(\mathbf{x}))^2$$

[pointwise squared loss]

Gradient Descent in Detail

(1) Linear Regression + Squared Loss (continued)





$$y(\mathbf{x}) \overset{(\star)}{=} \mathbf{w}^T\mathbf{x}$$

$$L_2(\mathbf{w}) = \frac{1}{2}\cdot \mathsf{RSS}(\mathbf{w}) = \frac{1}{2}\cdot\sum_{(\mathbf{x},c)\in D}(c - y(\mathbf{x}))^2$$

[pointwise squared loss]

$$\nabla L_2(\mathbf{w}) = \left(\frac{\partial L_2(\mathbf{w})}{\partial w_0}, \frac{\partial L_2(\mathbf{w})}{\partial w_1}, \cdots, \frac{\partial L_2(\mathbf{w})}{\partial w_p}\right)^T$$

Gradient Descent in Detail

(1) Linear Regression + Squared Loss (continued)

Update of weight vector $\mathbf{w}$: (BGD algorithm, Line 9)

$$\mathbf{w} = \mathbf{w} + \Delta\mathbf{w},$$

using the gradient of the loss function $L_2(\mathbf{w})$ to take steepest descent:

$$\Delta\mathbf{w} \;=\; -\eta \cdot \nabla L_2(\mathbf{w})$$

Gradient Descent in Detail

(1) Linear Regression + Squared Loss (continued)

Update of weight vector $\mathbf{w}$: (BGD algorithm, Line 9)

$$\mathbf{w} = \mathbf{w} + \Delta\mathbf{w},$$

using the gradient of the loss function $L_2(\mathbf{w})$ to take steepest descent:

$$\begin{aligned}
\Delta\mathbf{w} &= -\eta \cdot \nabla L_2(\mathbf{w}) \\
&= -\eta \cdot \left(\frac{\partial L_2(\mathbf{w})}{\partial w_0}, \frac{\partial L_2(\mathbf{w})}{\partial w_1}, \cdots, \frac{\partial L_2(\mathbf{w})}{\partial w_p} \right)^T \\
&\vdots \\
&= \eta \cdot \sum_{(\mathbf{x},c)\in D} (c - \mathbf{w}^T\mathbf{x}) \cdot \mathbf{x}
\end{aligned}$$

Remarks (derivation of $\nabla L_2(\mathbf{w})$):

- Consider the partial derivative for a parameter $w_j,\ j = 0, \ldots, p$:

$$
\begin{aligned}
\frac{\partial}{\partial w_j} L_2(\mathbf{w}) &= \frac{\partial}{\partial w_j} \frac{1}{2} \cdot \sum_{(\mathbf{x},c) \in D} (c - y(\mathbf{x}))^2 \\
&= \frac{1}{2} \cdot \sum_{(\mathbf{x},c) \in D} \frac{\partial}{\partial w_j} (c - y(\mathbf{x}))^2 \\
&\overset{(1)}{=} \sum_{(\mathbf{x},c) \in D} (c - y(\mathbf{x})) \cdot \frac{\partial}{\partial w_j} (c - y(\mathbf{x})) \\
&= \sum_{(\mathbf{x},c) \in D} (c - \mathbf{w}^T \mathbf{x}) \cdot \frac{\partial}{\partial w_j} (c - \mathbf{w}^T \mathbf{x}) \quad \text{// } j\text{-th summand depends on } w_j. \\
&= \sum_{(\mathbf{x},c) \in D} (c - \mathbf{w}^T \mathbf{x}) \cdot (-x_j) \\
&= - \sum_{(\mathbf{x},c) \in D} (c - \mathbf{w}^T \mathbf{x}) \cdot x_j
\end{aligned}
$$

- Plugging the results for $\frac{\partial}{\partial w_j} L_2(\mathbf{w})$ into $-\eta \cdot (\ldots)^T$ yields the update formula for $\Delta \mathbf{w}$.

- Hints:

(1) Chain rule with $\frac{d}{dz}(g(z))^2 = 2 \cdot g(z) \cdot \frac{d}{dz} g(z)$

Gradient Descent in Detail

The BGD Algorithm

[algorithms: LMS BGD_σ PT BGD IGD]

Algorithm:	BGD	Batch Gradient Descent
Input:	D	Multiset of examples $(\mathbf{x}, c)$ with $\mathbf{x} \in \mathbf{R}^p$, $c \in \{-1, 1\}$.
	η	Learning rate, a small positive constant.
Output:	$\mathbf{w}$	Weight vector from $\mathbf{R}^{p+1}$. (= hypothesis)

BGD(D, η)

1. *initialize_random_weights*$(\mathbf{w})$, $t = 0$
2. **REPEAT**
3. $t = t + 1$, $\Delta\mathbf{w} = 0$
4. FOREACH $(\mathbf{x}, c) \in D$ DO
5. $y(\mathbf{x}) \overset{(*)}{=} \mathbf{w}^T\mathbf{x}$
6. $\delta = c - y(\mathbf{x})$
7. $\Delta\mathbf{w} \overset{(*)}{=} \Delta\mathbf{w} + \eta \cdot \delta \cdot \mathbf{x}$ // $-\delta \cdot \mathbf{x}$ is the derivative of $l_2(c, y(\mathbf{x}))$ wrt. $\mathbf{w}$.
8. ENDDO
9. $\mathbf{w} = \mathbf{w} + \Delta\mathbf{w}$ // $\Delta\mathbf{w} = -\eta \cdot \nabla L_2(\mathbf{w})$
10. **UNTIL**(*convergence*$(D, y(\cdot), t)$)
11. *return*$(\mathbf{w})$

Gradient Descent in Detail

The BGD Algorithm [algorithms: LMS BGD_σ PT BGD IGD]

Algorithm:	BGD	Batch Gradient Descent
Input:	D	Multiset of examples $(\mathbf{x}, c)$ with $\mathbf{x} \in \mathbf{R}^p$, $c \in \{-1, 1\}$.
	η	Learning rate, a small positive constant.
Output:	$\mathbf{w}$	Weight vector from $\mathbf{R}^{p+1}$. (= hypothesis)

BGD(D, η)

1. *initialize_random_weights*$(\mathbf{w})$, $t = 0$
2. **REPEAT**
3. $t = t + 1$, $\Delta\mathbf{w} = 0$
4. FOREACH $(\mathbf{x}, c) \in D$ DO
5. $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ Model function evaluation.
6. $\delta = c - y(\mathbf{x})$ Calculation of residual.
7. $\Delta\mathbf{w} = \Delta\mathbf{w} + \eta \cdot \delta \cdot \mathbf{x}$ Calculation of derivative, accumulate for entire D.
8. ENDDO
9. $\mathbf{w} = \mathbf{w} + \Delta\mathbf{w}$ Parameter vector update = one gradient step down.
10. **UNTIL**(*convergence*$(D, y(\cdot), t)$)
11. *return*$(\mathbf{w})$

Remarks:

($\star$) Recap. We consider the feature vector $\mathbf{x}$ in its extended form when used as operand in a scalar product with the weight vector, $\mathbf{w}^T\mathbf{x}$, and consequently, when noted as argument of the model function, $y(\mathbf{x})$. I.e., $\mathbf{x} = (1, x_1, \ldots, x_p)^T \in \mathbf{R}^{p+1}$, and $x_0 = 1$.

- Recap. Each BGD iteration "`REPEAT ... UNTIL`"
  1. computes the direction of steepest loss descent as $-\nabla L_2(\mathbf{w}_t) = \sum_{(\mathbf{x},c)\in D} (c - y_t(\mathbf{x})) \cdot \mathbf{x}$, and
  2. updates $\mathbf{w}_t$ by taking a step of length η in this direction.
- Recap. The function *convergence*$(\cdot)$ can analyze the global squared loss, $L_2(\mathbf{w}_t)$, or the norm of the loss gradient, $||\nabla L_2(\mathbf{w}_t)||$, and compare it to a small positive bound ε. Consider in this regard the vectors of observed and computed classes, $D|_c$ and $y(D|_{\mathbf{x}})$ respectively. In addition, the function may check via t an upper bound on the number of iterations.

Remarks: (continued)

- The basic gradient descent is a first-order optimization method, and its speed of convergence may be considered unsatisfactory. Even when taking the optimal step size η at each iteration, it has only a linear rate of convergence. [Meza 2010]

  More advanced numerical algorithms to tackle the optimization (search for minimum L_2) are listed below but are not treated in detail here:

  - Newton-Raphson algorithm
  - BFGS algorithm (Broyden-Fletcher-Goldfarb-Shanno)
  - L-BFGS algorithm (limited memory BFGS)
  - conjugate gradient method

Gradient Descent in Detail

Global Loss versus Pointwise Loss

The weight adaptation of the BGD algorithm computes in each iteration the global loss, i.e., the loss of *all* examples in D ("batch gradient descent").

The (squared) loss with regard to a *single* example $(\mathbf{x}, c) \in D$, also called pointwise loss is given as:

$$l_2(c, y(\mathbf{x})) \;=\; \frac{1}{2}\,(c - \mathbf{w}^T\mathbf{x})^2 \qquad \text{[global squared loss]}$$

The respective weight adaptation computes canonically as follows:

$$\Delta\mathbf{w} \;=\; \eta \cdot (c - \mathbf{w}^T\mathbf{x}) \cdot \mathbf{x} \qquad \text{[IGD algorithm]}$$

Remarks:

- The adaptation rule for single example, $\Delta\mathbf{w} = \eta \cdot (c - \mathbf{w}^T\mathbf{x}) \cdot \mathbf{x}$, is known under different names:
  - delta rule
  - Widrow-Hoff rule
  - adaline rule
  - least mean squares (LMS) rule
- The delta rule gives rise to the IGD algorithm (incremental gradient descent), which is introduced in the following. Moreover, the delta rule forms the basis of the backpropagation algorithm.

Gradient Descent in Detail

The IGD Algorithm

[algorithms: LMS BGD_σ PT BGD IGD]

Algorithm:	IGD	Incremental Gradient Descent
Input:	D	Multiset of examples $(\mathbf{x}, c)$ with $\mathbf{x} \in \mathbf{R}^p$, $c \in \{-1, 1\}$.
	η	Learning rate, a small positive constant.
Output:	$\mathbf{w}$	Weight vector from $\mathbf{R}^{p+1}$. (= hypothesis)

IGD(D, η)

1. *initialize_random_weights*$(\mathbf{w})$, $t = 0$
2. **REPEAT**
3. $\quad t = t + 1$
4. $\quad$ FOREACH $(\mathbf{x}, c) \in D$ DO
5. $\quad\quad y(\mathbf{x}) \overset{(\star)}{=} \mathbf{w}^T\mathbf{x}$
6. $\quad\quad \delta = c - y(\mathbf{x})$
7. $\quad\quad \Delta\mathbf{w} \overset{(\star)}{=} \eta \cdot \delta \cdot \mathbf{x}$ // $-\delta \cdot \mathbf{x}$ is the derivative of $l_2(c, y(\mathbf{x}))$ wrt. $\mathbf{w}$.
8. $\quad\quad \mathbf{w} = \mathbf{w} + \Delta\mathbf{w}$
9. $\quad$ ENDDO
10. **UNTIL**(*convergence*$(D, y(\,\cdot\,), t)$)
11. *return*$(\mathbf{w})$

Gradient Descent in Detail

The IGD Algorithm

[algorithms: LMS BGD$_\sigma$ PT BGD IGD]

Algorithm:	IGD	Incremental Gradient Descent
Input:	D	Multiset of examples $(\mathbf{x}, c)$ with $\mathbf{x} \in \mathbf{R}^p$, $c \in \{-1, 1\}$.
	η	Learning rate, a small positive constant.
Output:	$\mathbf{w}$	Weight vector from $\mathbf{R}^{p+1}$. (= hypothesis)

IGD(D, η)

1. *initialize_random_weights*$(\mathbf{w})$, $t = 0$
2. **REPEAT**
3. $t = t + 1$
4. FOREACH $(\mathbf{x}, c) \in D$ DO
5. $y(\mathbf{x}) \overset{(1)}{=} \mathbf{w}^T \mathbf{x}$ — Model function evaluation.
6. $\delta = c - y(\mathbf{x})$ — Calculation of residual.
7. $\Delta \mathbf{w} \overset{(2)}{=} \eta \cdot \delta \cdot \mathbf{x}$ // $-\delta \cdot \frac{\partial}{\partial \mathbf{w}} l_2(c, y(\mathbf{x}))$ wrt. $\mathbf{w}$. — Calculation of derivative.
8. $\mathbf{w} = \mathbf{w} + \Delta \mathbf{w}$ — Parameter vector update = one gradient step down.
9. ENDDO
10. **UNTIL**(*convergence*$(D, y(\cdot), t)$)
11. *return*$(\mathbf{w})$

Remarks (IGD) :

- The sequence of incremental weight adaptations approximates the gradient descent of the batch approach. If η is chosen sufficiently small this approximation can be done at arbitrary precision.
- The computation of the global loss, $L_2(\mathbf{w})$, of batch gradient descent enables larger weight adaptation steps.
- Compared to batch gradient descent, the example-based weight adaptation of incremental gradient descent can better avoid getting stuck in a local minimum of the loss function.
- When, as is done here, the residual sum of squares, RSS, is chosen as loss function, the incremental gradient descent algorithm is very similar to the LMS algorithm.
- A related method to incremental gradient descent is *stochastic* gradient descent, SGD, which estimates the gradient from a randomly selected subset of the data.

Remarks (recap. different roles of loss functions) :

- Observe that loss functions are employed at two places (in two roles) in an optimization approach:
  1. For the fitting of the data (i.e., the parameter update during regression / optimization / hyperplane search), where a new position of the hyperplane is computed.

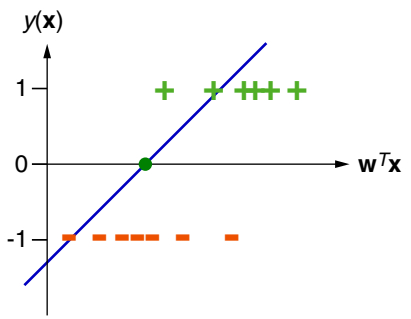
     Example: Lines 6+7 in the BGD Algorithm and the IGD Algorithm.
  2. For the evaluation of a hypothesis' effectiveness, where the portion of correctly and misclassified examples is analyzed.

     Example: Line 10 in the BGD Algorithm and the IGD Algorithm.
     General: section Evaluating Effectiveness of part Machine Learning Basics.

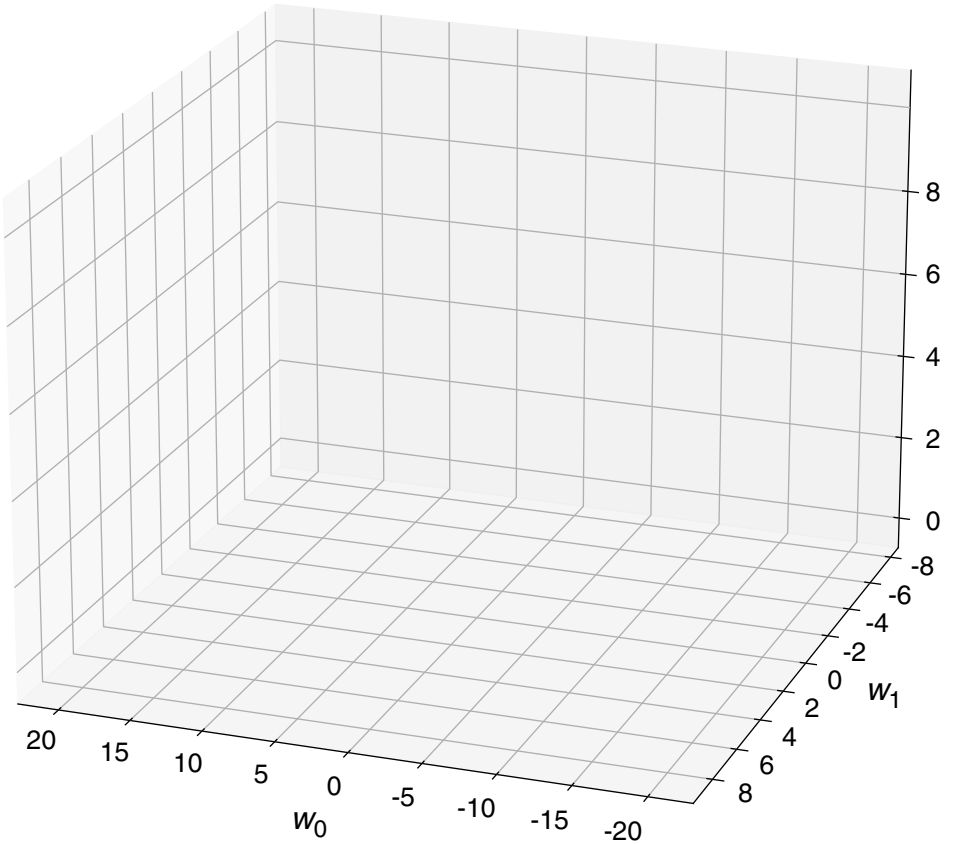
  Typically, fitting (optimization) (1.) and effectiveness evaluation (2.) are done with different loss functions. E.g., logistic regression uses L_σ and $L_{0/1}$ for fitting and evaluation respectively. However, linear regression (not classification) uses RSS (the L_2 loss) for both fitting and evaluation. The basic perceptron learning algorithm uses the misclassification information (the $L_{0/1}$ loss) for both fitting and evaluation.

Gradient Descent in Detail

(2) Linear Regression + 0/1 Loss

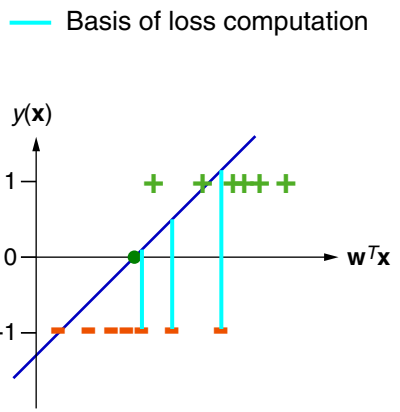


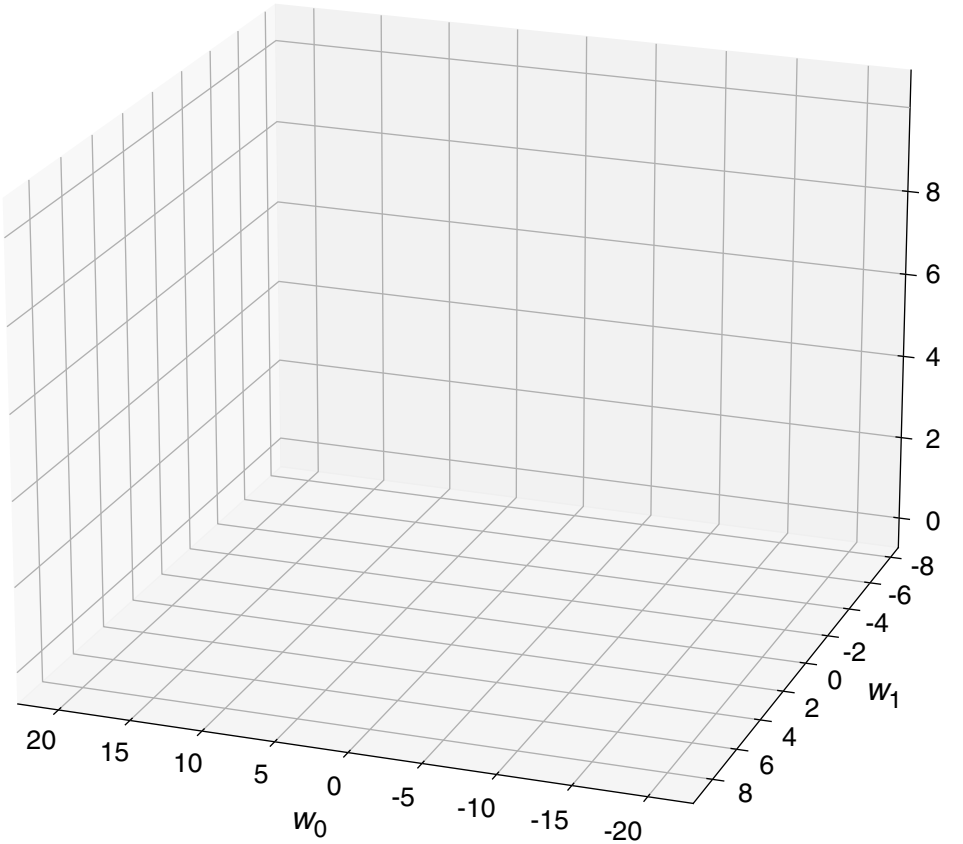
$$y(\mathbf{x}) \overset{(\star)}{=} \mathbf{w}^T\mathbf{x}$$



Gradient Descent in Detail

(2) Linear Regression + 0/1 Loss (continued)



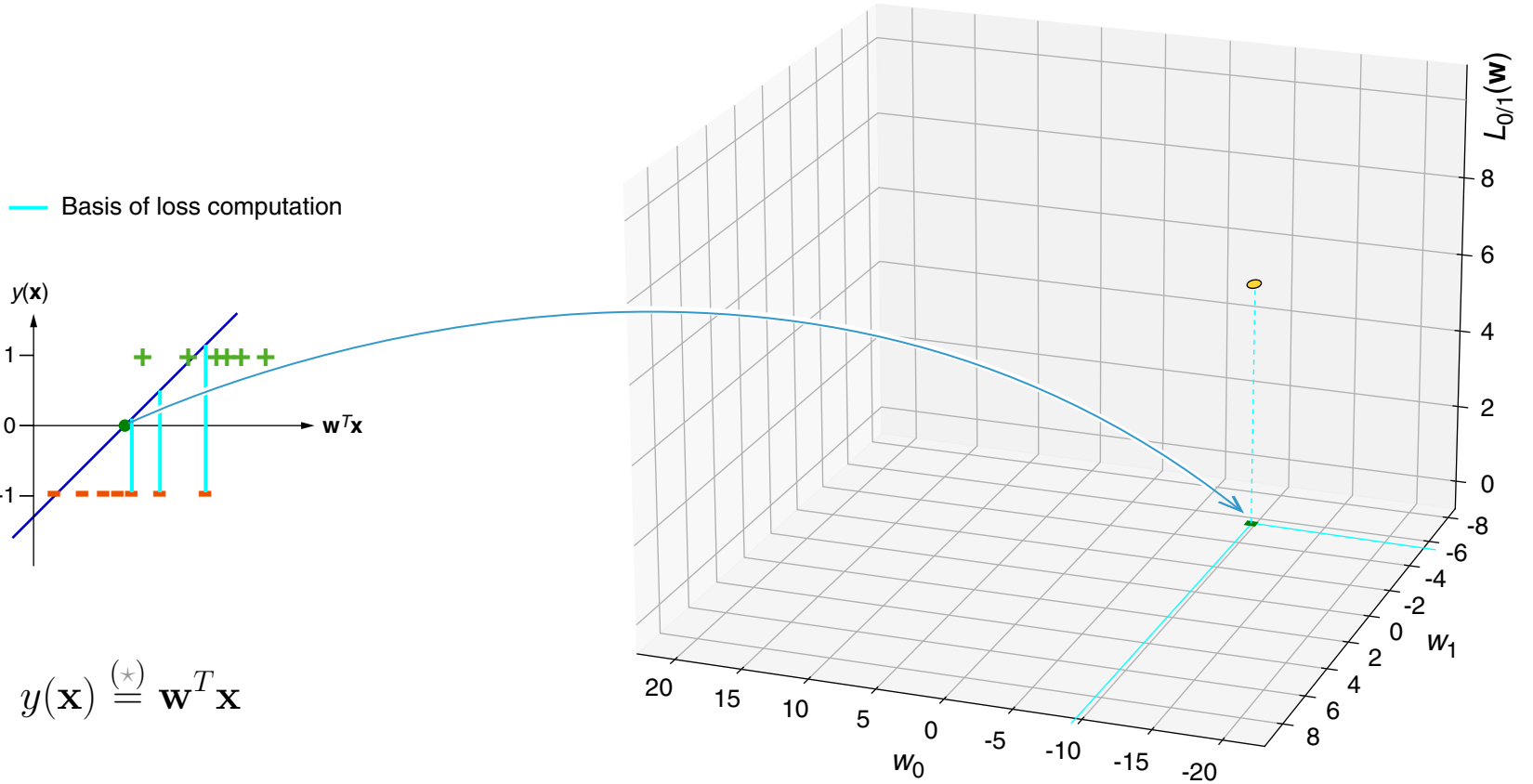


$$y(\mathbf{x}) \overset{(\star)}{=} \mathbf{w}^T\mathbf{x}$$

$$L_{0/1}(\mathbf{w}) = \sum_{(\mathbf{x},c)\in D} I_{\neq}(c, \text{sign}(y(\mathbf{x}))) = \sum_{(\mathbf{x},c)\in D} \frac{1}{2}\cdot(c - \text{sign}(\mathbf{w}^T\mathbf{x}))$$ [pointwise 0/1 loss]

Gradient Descent in Detail

(2) Linear Regression + 0/1 Loss (continued)

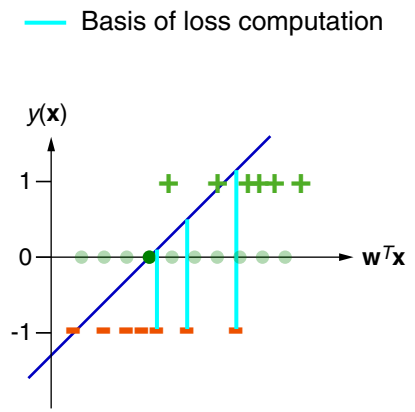


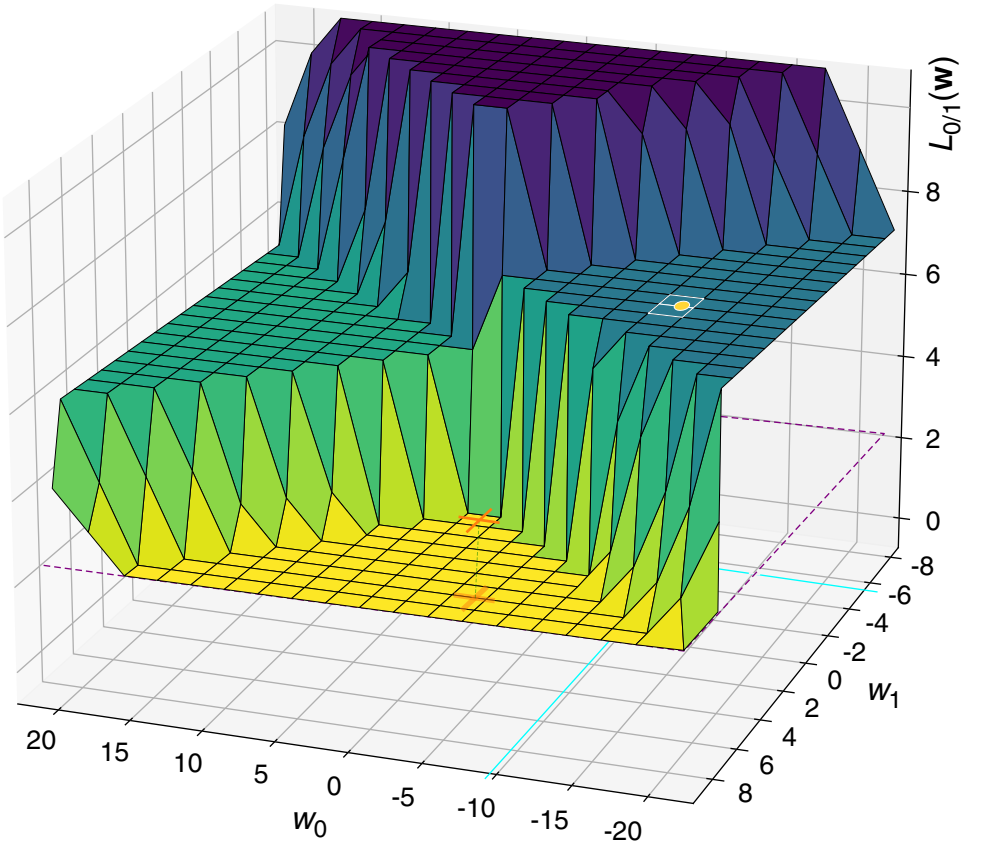
$$y(\mathbf{x}) \overset{(\star)}{=} \mathbf{w}^T\mathbf{x}$$

$$L_{0/1}(\mathbf{w}) = \sum_{(\mathbf{x},c)\in D} I_{\neq}(c, \text{sign}(y(\mathbf{x}))) = \sum_{(\mathbf{x},c)\in D} \frac{1}{2}\cdot(c - \text{sign}(\mathbf{w}^T\mathbf{x}))$$ [pointwise 0/1 loss]

Gradient Descent in Detail

(2) Linear Regression + 0/1 Loss (continued)





$$y(\mathbf{x}) \overset{(\star)}{=} \mathbf{w}^T\mathbf{x}$$

$$L_{0/1}(\mathbf{w}) = \sum_{(\mathbf{x},c)\in D} I_{\neq}(c, \text{sign}(y(\mathbf{x}))) = \sum_{(\mathbf{x},c)\in D} \frac{1}{2} \cdot (c - \text{sign}(\mathbf{w}^T\mathbf{x}))$$ [pointwise 0/1 loss]

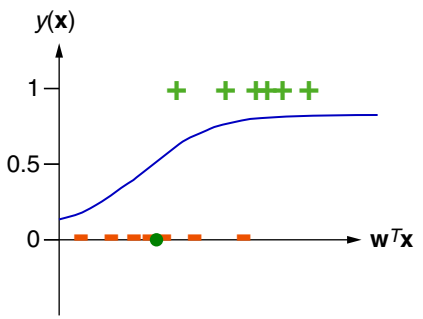
$\nabla L_{0/1}(\mathbf{w})$ cannot be expressed as a differentiable function.

Remarks:

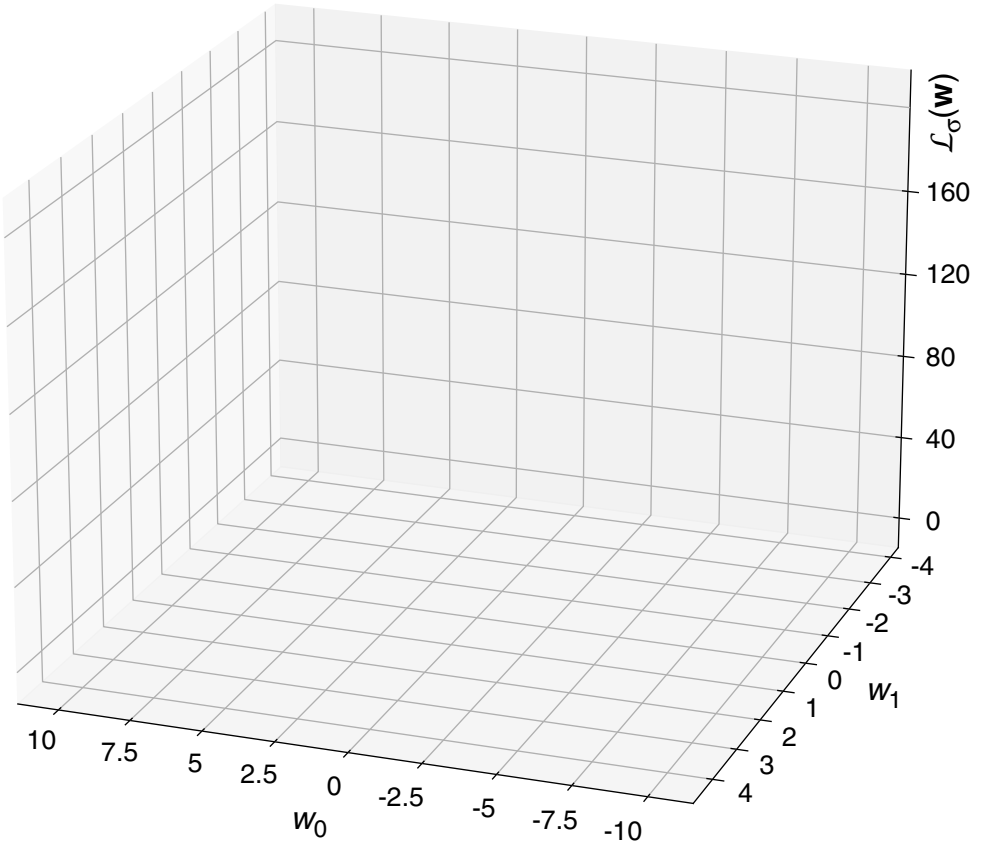
- Since $\nabla L_{0/1}(\mathbf{w})$ is not a differentiable function the gradient descent method cannot be applied to determine its minimum.
- Recap. $I_{\neq}$ is an indicator function that returns 1 if its arguments are *unequal* (and 0 if its arguments are equal).
- Recap. We label $y(0)$ with the "positive" class and define sign$(0) = 1$ here.

Gradient Descent in Detail

(3) Logistic Regression + Logistic Loss + Regularization

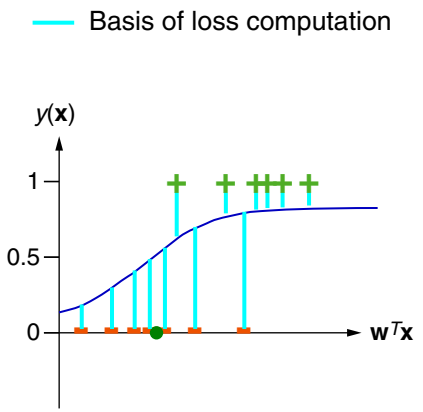


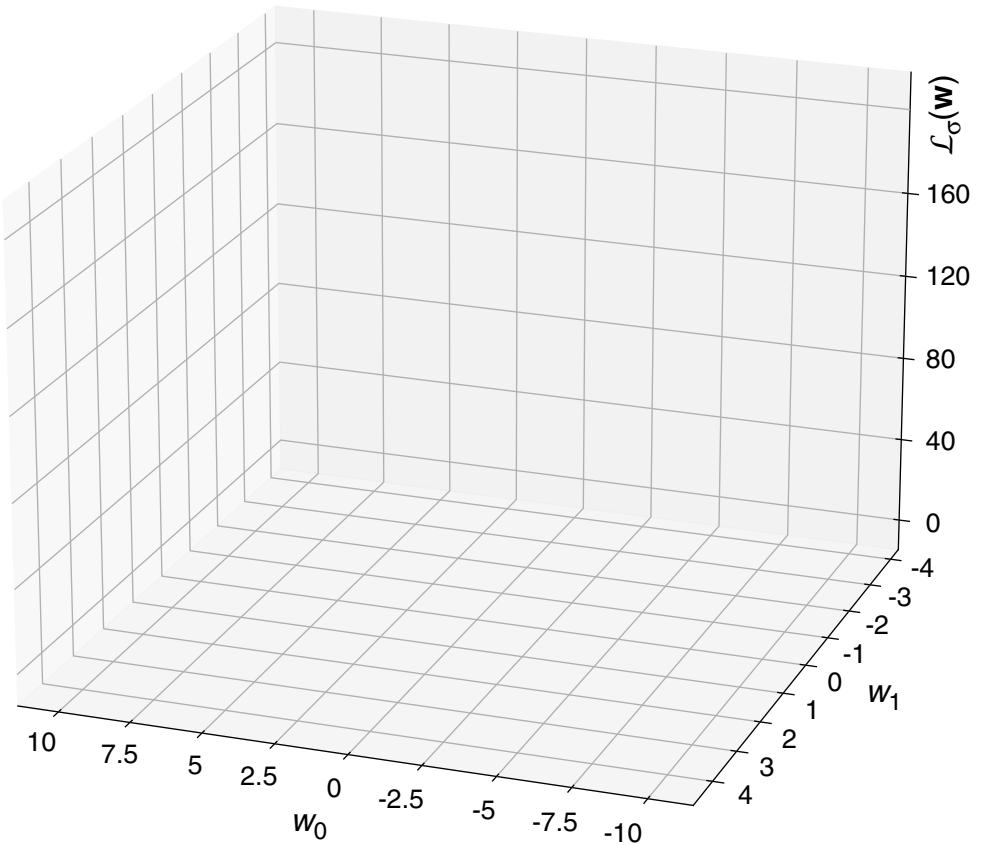
$$y(\mathbf{x}) \overset{(\star)}{=} \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$$



Gradient Descent in Detail

(3) Logistic Regression + Logistic Loss + Regularization (continued)





$$y(\mathbf{x}) \overset{(\star)}{=} \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$$

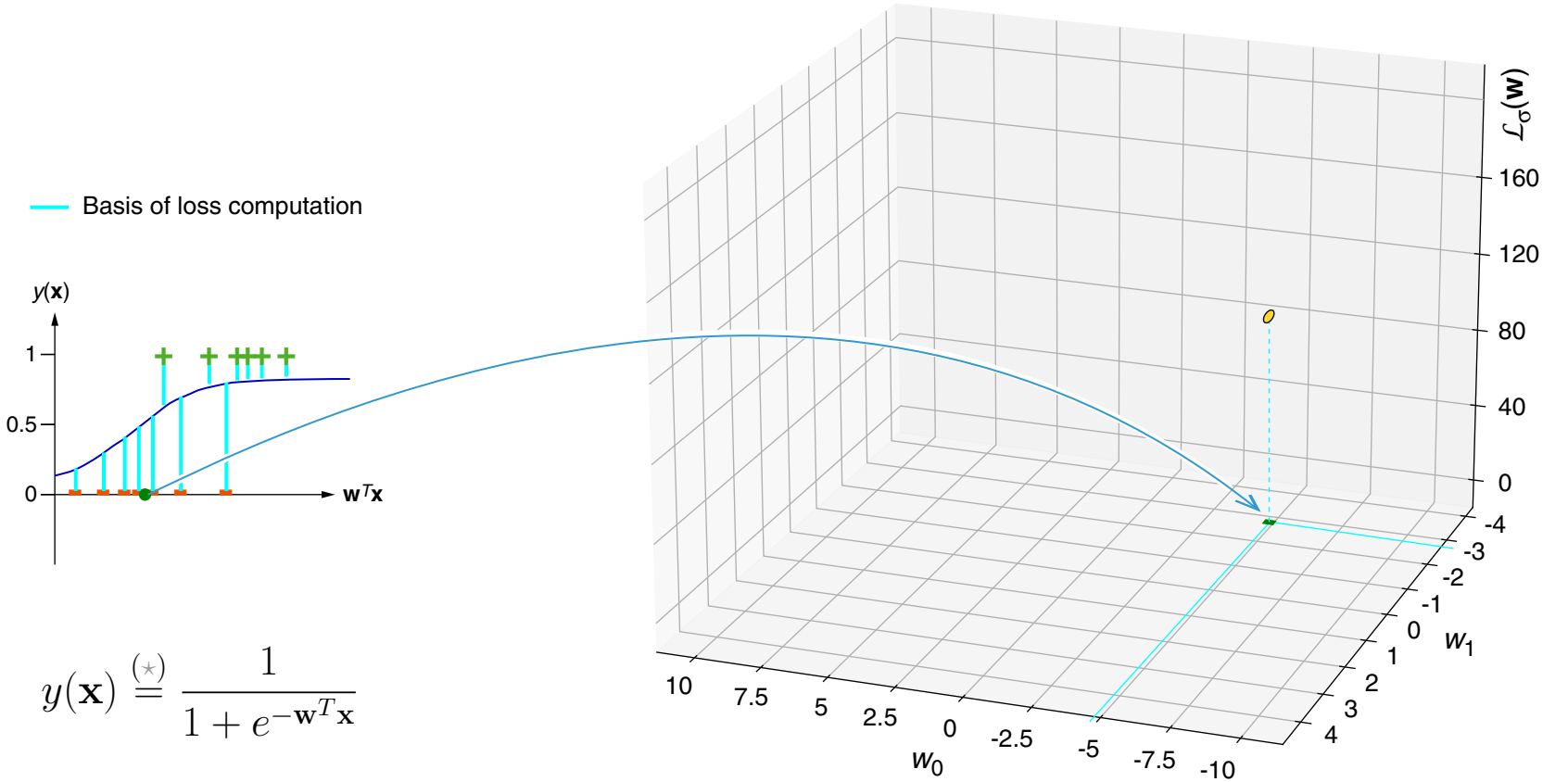
$$\mathcal{L}_\sigma(\mathbf{w}) \;=\; L_\sigma + \lambda \cdot R_{\|\vec{\mathbf{w}}\|_2^2} \;=\; \sum_{(\mathbf{x},c)\in D} l_\sigma(c, y(\mathbf{x})) + \lambda \cdot \vec{\mathbf{w}}^T\vec{\mathbf{w}}$$

[definitions: L_σ, l_σ, $R_{\|\vec{\mathbf{w}}\|_2^2}$]

$$= \sum_{(\mathbf{x},c)\in D} -c \cdot \log(y(\mathbf{x})) - (1-c) \cdot \log(1-y(\mathbf{x})) + \lambda \cdot \vec{\mathbf{w}}^T\vec{\mathbf{w}}$$

Gradient Descent in Detail

(3) Logistic Regression + Logistic Loss + Regularization (continued)



$$y(\mathbf{x}) \overset{(\star)}{=} \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$$

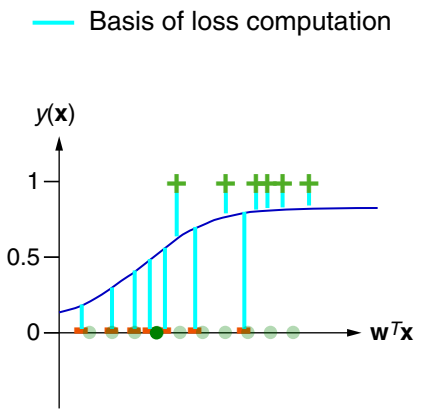
$$\mathcal{L}_\sigma(\mathbf{w}) \;=\; L_\sigma + \lambda \cdot R_{\|\vec{\mathbf{w}}\|_2^2} \;=\; \sum_{(\mathbf{x},c)\in D} l_\sigma(c, y(\mathbf{x})) + \lambda \cdot \vec{\mathbf{w}}^T\vec{\mathbf{w}}$$

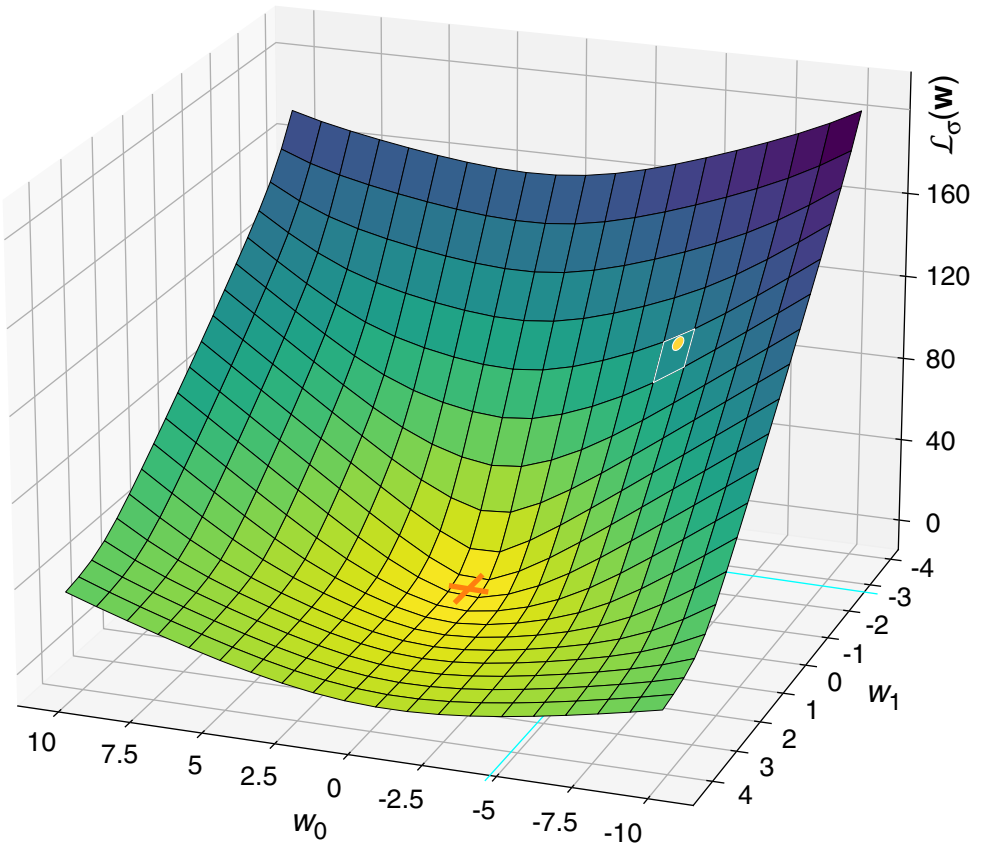
[definitions: L_σ, l_σ, $R_{\|\vec{\mathbf{w}}\|_2^2}$]

$$= \sum_{(\mathbf{x},c)\in D} -c \cdot \log(y(\mathbf{x})) - (1-c) \cdot \log(1-y(\mathbf{x})) + \lambda \cdot \vec{\mathbf{w}}^T\vec{\mathbf{w}}$$

Gradient Descent in Detail

(3) Logistic Regression + Logistic Loss + Regularization (continued)





$$y(\mathbf{x}) \overset{(\star)}{=} \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$$

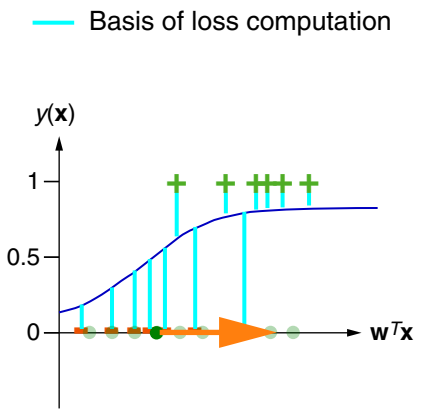
$$\mathcal{L}_\sigma(\mathbf{w}) \;=\; L_\sigma + \lambda \cdot R_{\|\vec{\mathbf{w}}\|_2^2} \;=\; \sum_{(\mathbf{x},c)\in D} l_\sigma(c, y(\mathbf{x})) + \lambda \cdot \vec{\mathbf{w}}^T\vec{\mathbf{w}}$$

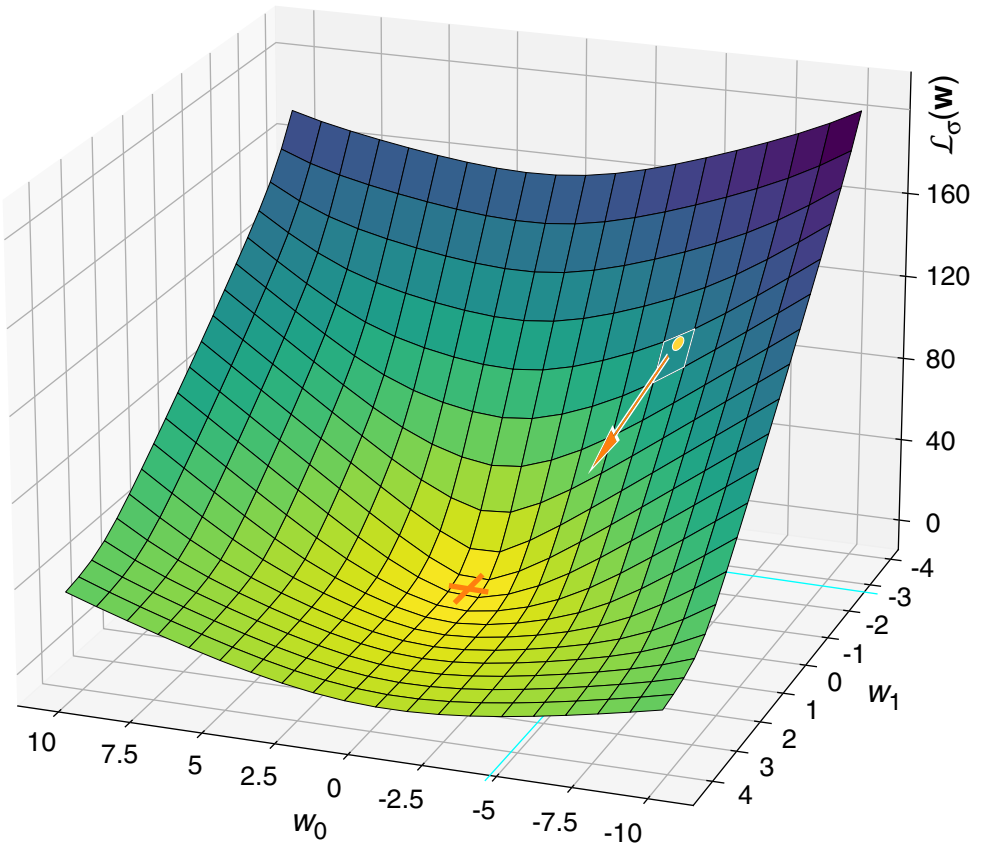
[definitions: L_σ, l_σ, $R_{\|\vec{\mathbf{w}}\|_2^2}$]

$$= \sum_{(\mathbf{x},c)\in D} -c \cdot \log(y(\mathbf{x})) - (1-c) \cdot \log(1-y(\mathbf{x})) + \lambda \cdot \vec{\mathbf{w}}^T\vec{\mathbf{w}}$$

Gradient Descent in Detail

(3) Logistic Regression + Logistic Loss + Regularization (continued)





$$y(\mathbf{x}) \overset{(\star)}{=} \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$$

$$\mathcal{L}_\sigma(\mathbf{w}) \;=\; L_\sigma + \lambda \cdot R_{\|\vec{\mathbf{w}}\|_2^2} \;=\; \sum_{(\mathbf{x},c)\in D} l_\sigma(c, y(\mathbf{x})) + \lambda \cdot \vec{\mathbf{w}}^T\vec{\mathbf{w}}$$

[definitions: L_σ, l_σ, $R_{\|\vec{\mathbf{w}}\|_2^2}$]

$$\nabla\mathcal{L}_\sigma(\mathbf{w}) \;=\; \left(\frac{\partial\mathcal{L}_\sigma(\mathbf{w})}{\partial w_0}, \frac{\partial\mathcal{L}_\sigma(\mathbf{w})}{\partial w_1}, \ldots, \frac{\partial\mathcal{L}_\sigma(\mathbf{w})}{\partial w_p}\right)^T$$

Gradient Descent in Detail

(3) Logistic Regression + Logistic Loss + Regularization (continued)

Update of weight vector $\mathbf{w}$: (BGD$_\sigma$ algorithm, Line 9)

$$\mathbf{w} = \mathbf{w} + \Delta\mathbf{w},$$

using the gradient of the objective function $\mathcal{L}_\sigma(\mathbf{w})$ to take steepest descent:

$$\Delta\mathbf{w} \;=\; -\eta \cdot \nabla\mathcal{L}_\sigma(\mathbf{w})$$

Gradient Descent in Detail

(3) Logistic Regression + Logistic Loss + Regularization (continued)

Update of weight vector $\mathbf{w}$: (BGD$_\sigma$ algorithm, Line 9)

$$\mathbf{w} = \mathbf{w} + \Delta\mathbf{w},$$

using the gradient of the objective function $\mathcal{L}_\sigma(\mathbf{w})$ to take steepest descent:

$$\begin{aligned}
\Delta\mathbf{w} &= -\eta \cdot \nabla\mathcal{L}_\sigma(\mathbf{w}) \\
&= -\eta \cdot \left(\frac{\partial\mathcal{L}_\sigma(\mathbf{w})}{\partial w_0}, \frac{\partial\mathcal{L}_\sigma(\mathbf{w})}{\partial w_1}, \cdots, \frac{\partial\mathcal{L}_\sigma(\mathbf{w})}{\partial w_p} \right)^T \\
&\vdots \\
&= \eta \cdot \sum_{(\mathbf{x},c)\in D} (c - \sigma(\mathbf{w}^T\mathbf{x})) \cdot \mathbf{x} \quad - \quad \eta \cdot 2\lambda \cdot \begin{pmatrix} 0 \\ \vec{\mathbf{w}} \end{pmatrix}
\end{aligned}$$

Remarks:

- Recap. Distinguish between the p-dimensional direction vector $\vec{\mathbf{w}} = (w_1, \ldots, w_p)^T$, and the $(p+1)$-dimensional hypothesis $\mathbf{w} = (w_0, w_1, \ldots, w_p)^T$.
- The BGD variant BGD$_\sigma$ has already been introduced in section Logistic Regression of part Linear Models. However, none of the algorithms presented so far consider the update term $\eta \cdot 2\lambda \cdot \binom{0}{\vec{\mathbf{w}}}$ for the ridge regression regularization constraint, $\lambda \cdot \vec{\mathbf{w}}^T \vec{\mathbf{w}}$, which has been derived now and which may be added to the respective algorithms as follows:

[BGD$_\sigma$, BGD] Line 9: $\mathbf{w} = \mathbf{w} + \Delta\mathbf{w} - \eta \cdot 2\lambda \cdot \binom{0}{\vec{\mathbf{w}}}$

[IGD] Line 8: $\mathbf{w} = \mathbf{w} + \Delta\mathbf{w} - \eta \cdot 2\frac{\lambda}{|D|} \cdot \binom{0}{\vec{\mathbf{w}}}$

Remarks (derivation of $\nabla\mathcal{L}_\sigma(\mathbf{w})$):

- Consider the partial derivative for a parameter $w_j,\ j = 0, \ldots, p$:

$$\begin{aligned}
\frac{\partial}{\partial w_j}\mathcal{L}_\sigma(\mathbf{w}) &= \frac{\partial}{\partial w_j}L_\sigma(\mathbf{w}) \ + \ \frac{\partial}{\partial w_j}\lambda \cdot R_{\|\vec{\mathbf{w}}\|_2^2}(\mathbf{w}) \\
&= \sum_{(\mathbf{x},c)\in D} \frac{\partial}{\partial w_j} l_\sigma(c, y(\mathbf{x})) \ + \ \lambda \cdot \frac{\partial}{\partial w_j}\vec{\mathbf{w}}^T\vec{\mathbf{w}} \\
&= \sum_{(\mathbf{x},c)\in D} \frac{\partial}{\partial w_j} \left[-c \cdot \log(\sigma(\mathbf{w}^T\mathbf{x})) - (1-c)\cdot\log(1-\sigma(\mathbf{w}^T\mathbf{x}))\right] \ + \ \lambda \cdot \frac{\partial}{\partial w_j}\sum_{i=1}^{p} w_i^2 \\
&= \sum_{(\mathbf{x},c)\in D} \left[-c \cdot \frac{\partial}{\partial w_j}\log(\sigma(\mathbf{w}^T\mathbf{x})) - (1-c)\cdot\frac{\partial}{\partial w_j}\log(1-\sigma(\mathbf{w}^T\mathbf{x}))\right] \overset{(1)}{+} \ \lambda \cdot \frac{\partial}{\partial w_j} w_j^2 \\
&\overset{(2)}{=} \sum_{(\mathbf{x},c)\in D} \left[-c \cdot \frac{1}{\sigma(\mathbf{w}^T\mathbf{x})} \cdot \frac{\partial}{\partial w_j}\sigma(\mathbf{w}^T\mathbf{x}) \right. \\
&\qquad \left. - (1-c)\cdot\frac{1}{1-\sigma(\mathbf{w}^T\mathbf{x})}\cdot\left(-\frac{\partial}{\partial w_j}\sigma(\mathbf{w}^T\mathbf{x})\right)\right] \overset{(1)}{+} \ 2\lambda \cdot w_j \\
&= \hookrightarrow \text{ p. 153}
\end{aligned}$$

Remarks (derivation of $\nabla \mathcal{L}_\sigma(\mathbf{w})$): (continued)

$$\overset{(3)}{=} \sum_{(\mathbf{x},c)\in D} \Bigg[-c \cdot \frac{1}{\sigma(\mathbf{w}^T\mathbf{x})} \cdot \sigma(\mathbf{w}^T\mathbf{x}) \cdot (1-\sigma(\mathbf{w}^T\mathbf{x})) \cdot \frac{\partial}{\partial w_j}\mathbf{w}^T\mathbf{x} - (1-c) \cdot \frac{1}{1-\sigma(\mathbf{w}^T\mathbf{x})} \cdot (-1) \cdot \sigma(\mathbf{w}^T\mathbf{x}) \cdot (1-\sigma(\mathbf{w}^T\mathbf{x})) \cdot \frac{\partial}{\partial w_j}\mathbf{w}^T\mathbf{x} \Bigg] \overset{(1)}{+} 2\lambda \cdot w_j$$

$$= \sum_{(\mathbf{x},c)\in D} -c \cdot (1-\sigma(\mathbf{w}^T\mathbf{x})) \cdot x_j + (1-c) \cdot \sigma(\mathbf{w}^T\mathbf{x}) \cdot x_j \overset{(1)}{+} 2\lambda \cdot w_j$$

$$= -\sum_{(\mathbf{x},c)\in D} (c - \sigma(\mathbf{w}^T\mathbf{x})) \cdot x_j \overset{(1)}{+} 2\lambda \cdot w_j$$

- Plugging the results for $\frac{\partial}{\partial w_j}\mathcal{L}_\sigma(\mathbf{w})$ into $-\eta \cdot (\ldots)^T$ yields the update formula for $\Delta\mathbf{w}$.
- Hints:
  - (1) Since $\vec{\mathbf{w}} \equiv (w_1, \ldots, w_p)^T$ the right summand is defined as 0 for $w_j = w_0$.
  - (2) Chain rule with $\frac{d}{dz}\log(z) = \frac{1}{z}$
  - (3) Chain rule with $\frac{d}{dz}\sigma(z) = \sigma(z) \cdot (1-\sigma(z))$, where $\sigma(z) = \frac{1}{1+e^{-z}}$