# Chapter ML:III

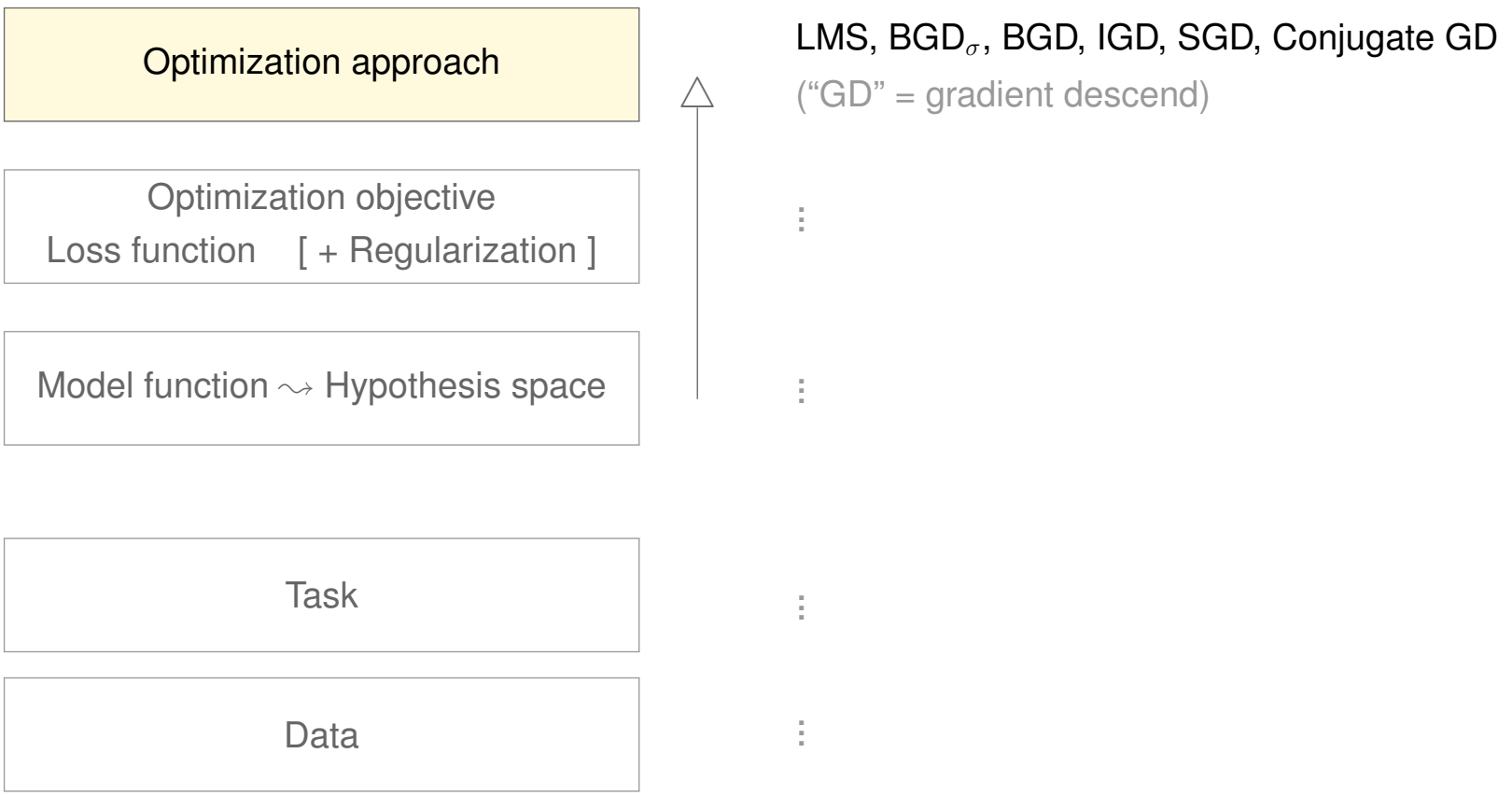
## III. Linear Models

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

# Gradient Descent in Detail

## ML Stack: Gradient Descend

[ML stack: LMS, log. regression, loss comp., regularization, GD ]

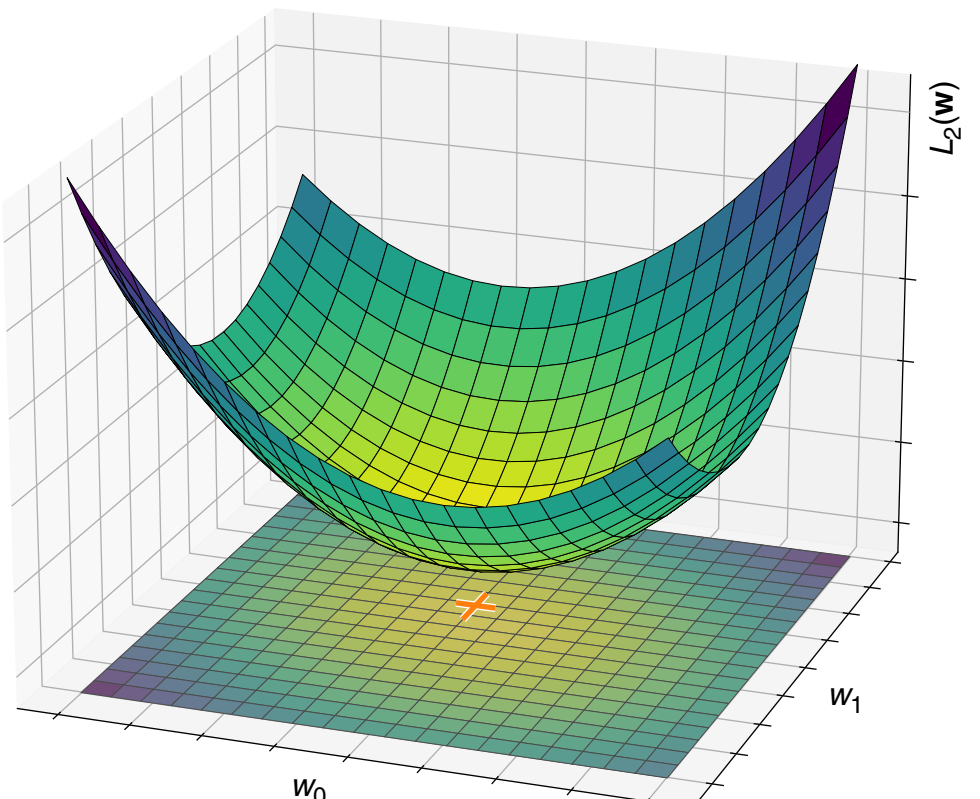


# Gradient Descent in Detail

## 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, $\mathcal{L}$.

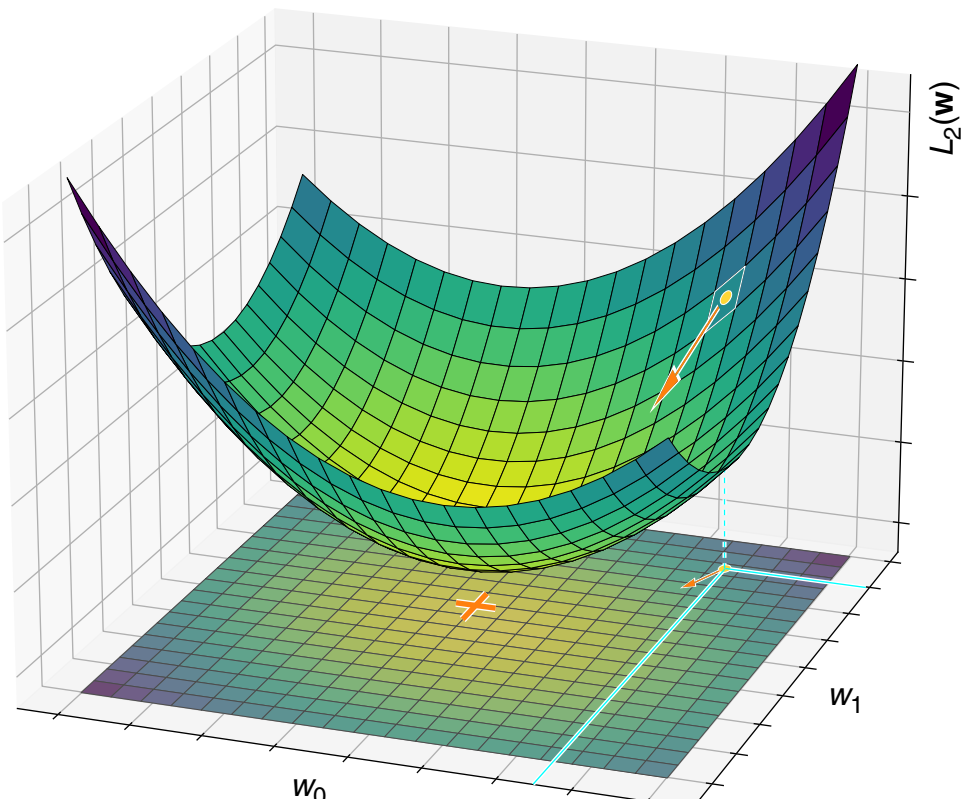


# Gradient Descent in Detail

## Principle

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, $\mathcal{L}$.

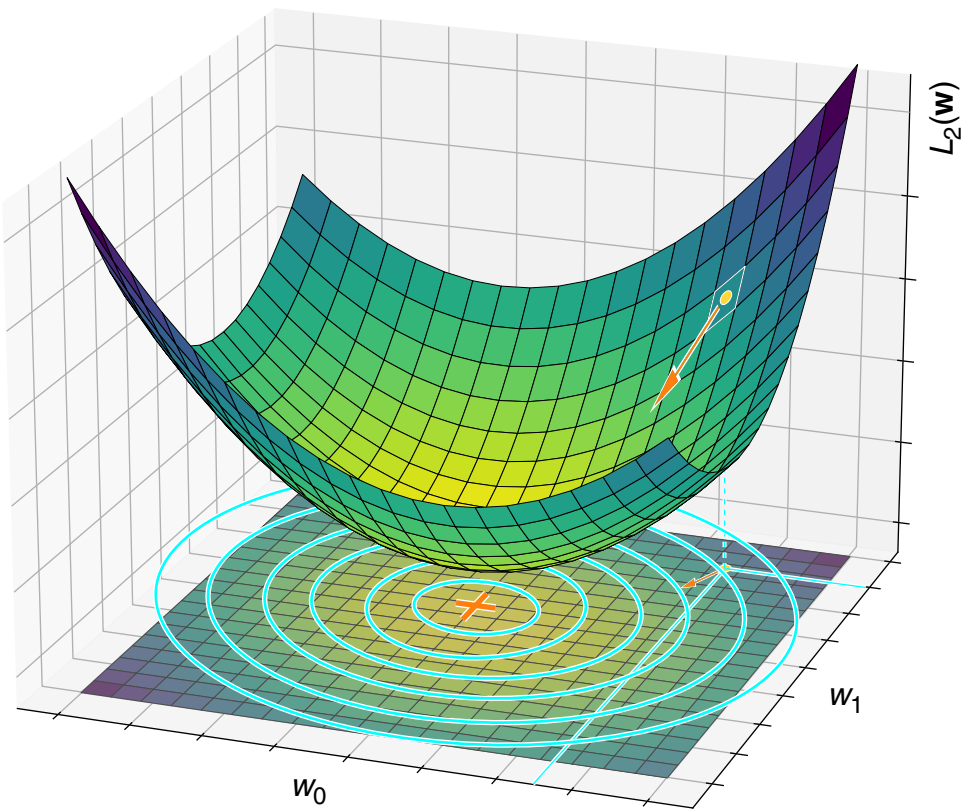


# Gradient Descent in Detail

## Principle

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, $\mathcal{L}$.



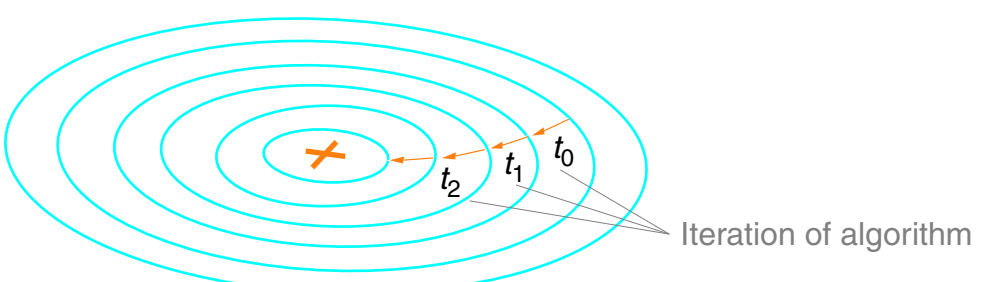
# Gradient Descent in Detail

## Principle

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 $\nabla f$ 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

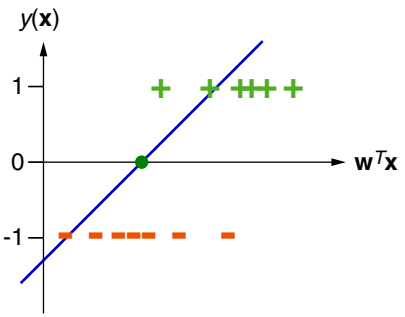
  (5) multilayer perceptron with $d$ hidden layers and $k$-dimensional output + squared loss

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

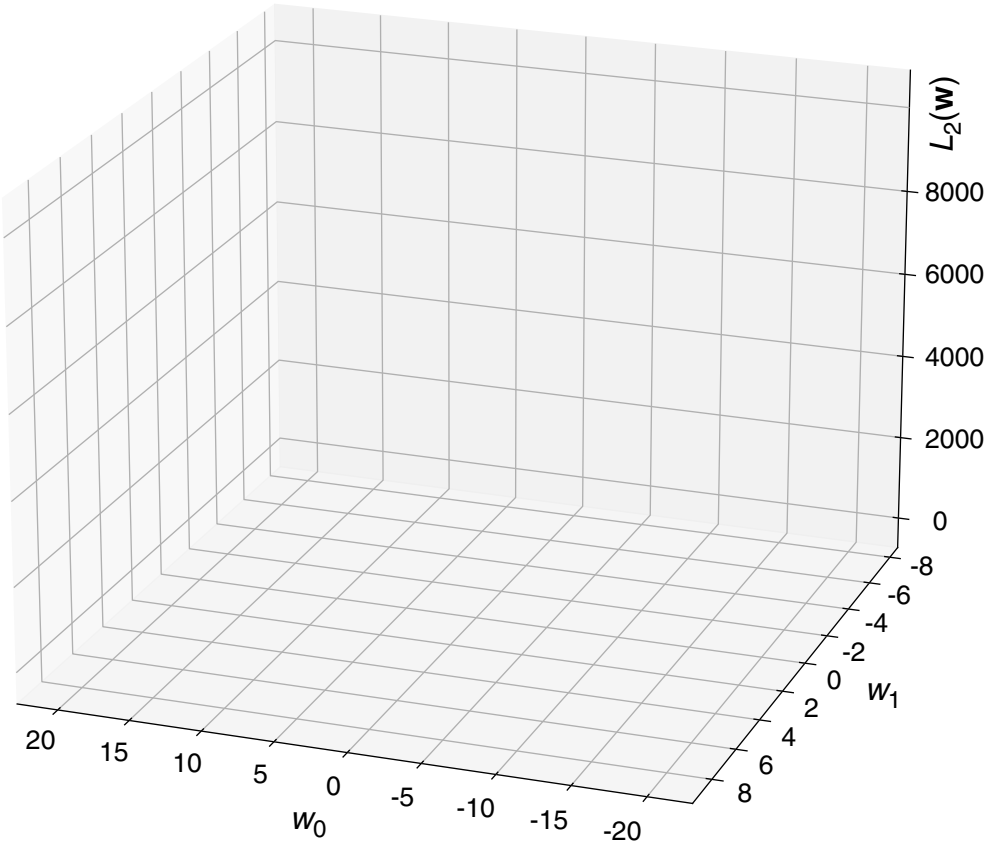
  The (loss or objective) functions considered in the settings (1) and (3) are convex; the (loss or objective) functions considered in the settings (2), (4), and (5) are typically non-convex.

# Gradient Descent in Detail

## (1) Linear Regression + Squared Loss

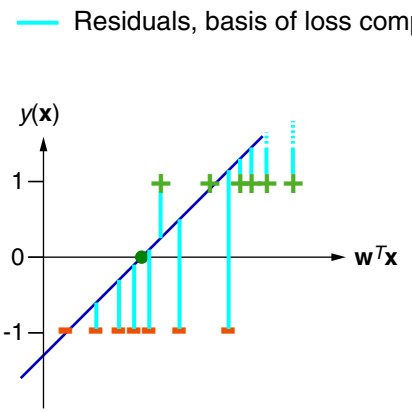


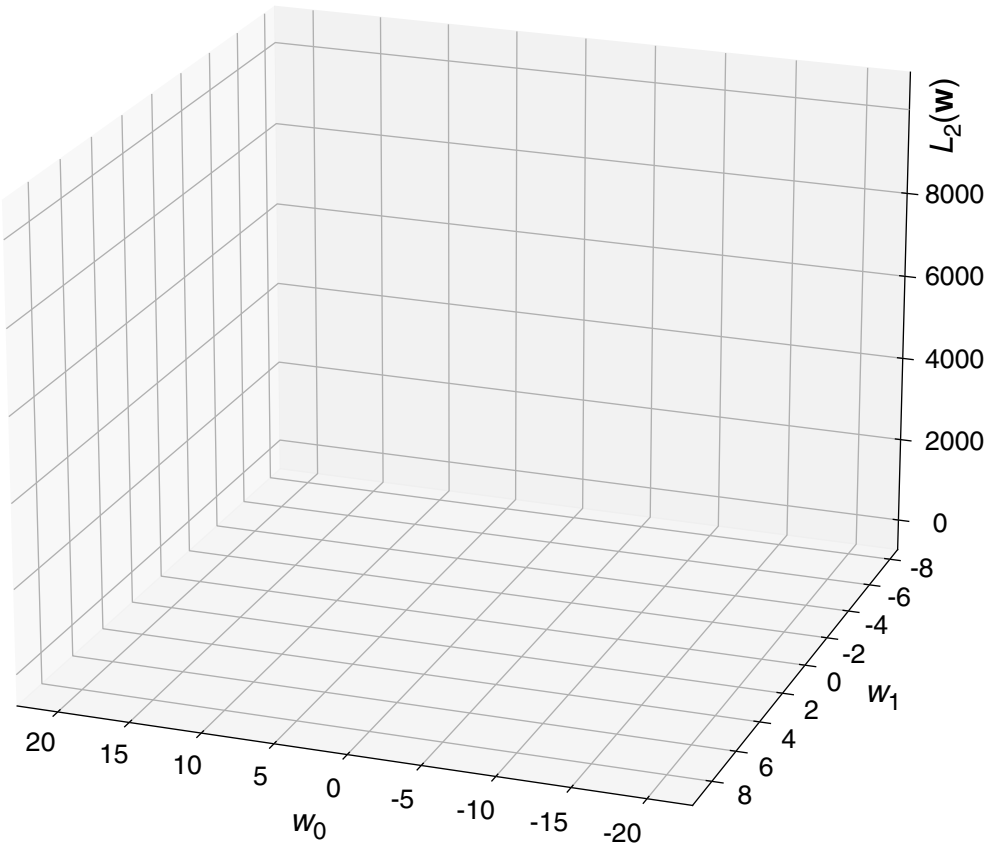

$$y(\mathbf{x}) \overset{(\star)}{=} \mathbf{w}^T\mathbf{x}$$



# Gradient Descent in Detail

## (1) Linear Regression + Squared Loss



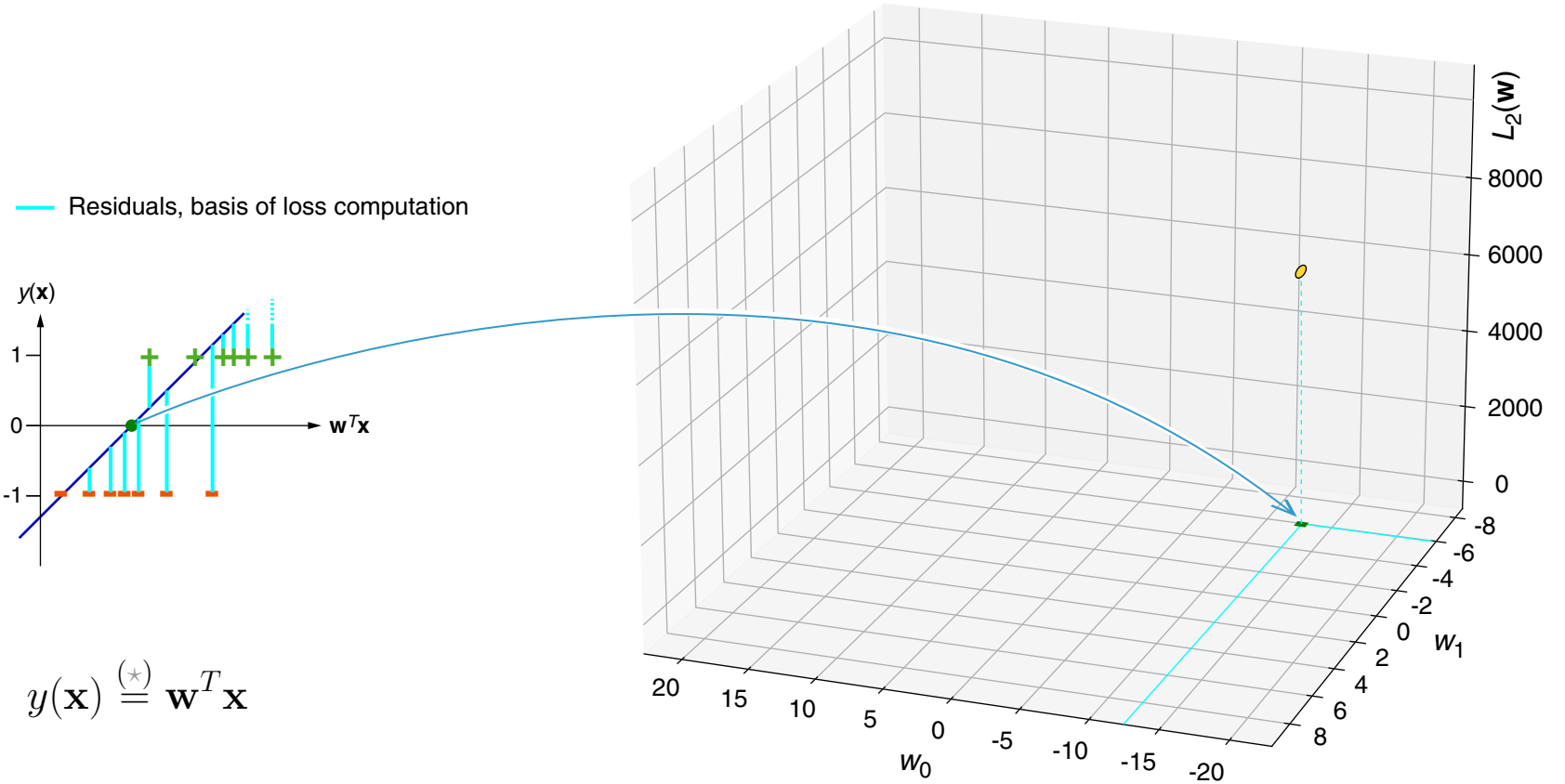

$$y(\mathbf{x}) \overset{(\star)}{=} \mathbf{w}^T\mathbf{x}$$

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

[pointwise squared loss]

# Gradient Descent in Detail

## (1) Linear Regression + Squared Loss



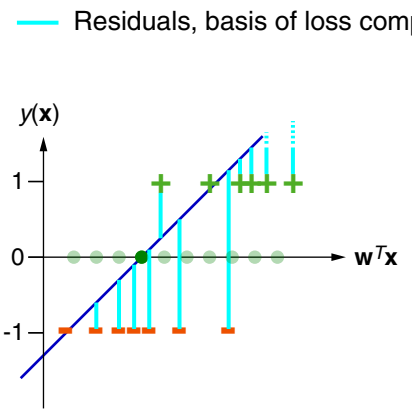

$$y(\mathbf{x}) \overset{(\star)}{=} \mathbf{w}^T\mathbf{x}$$

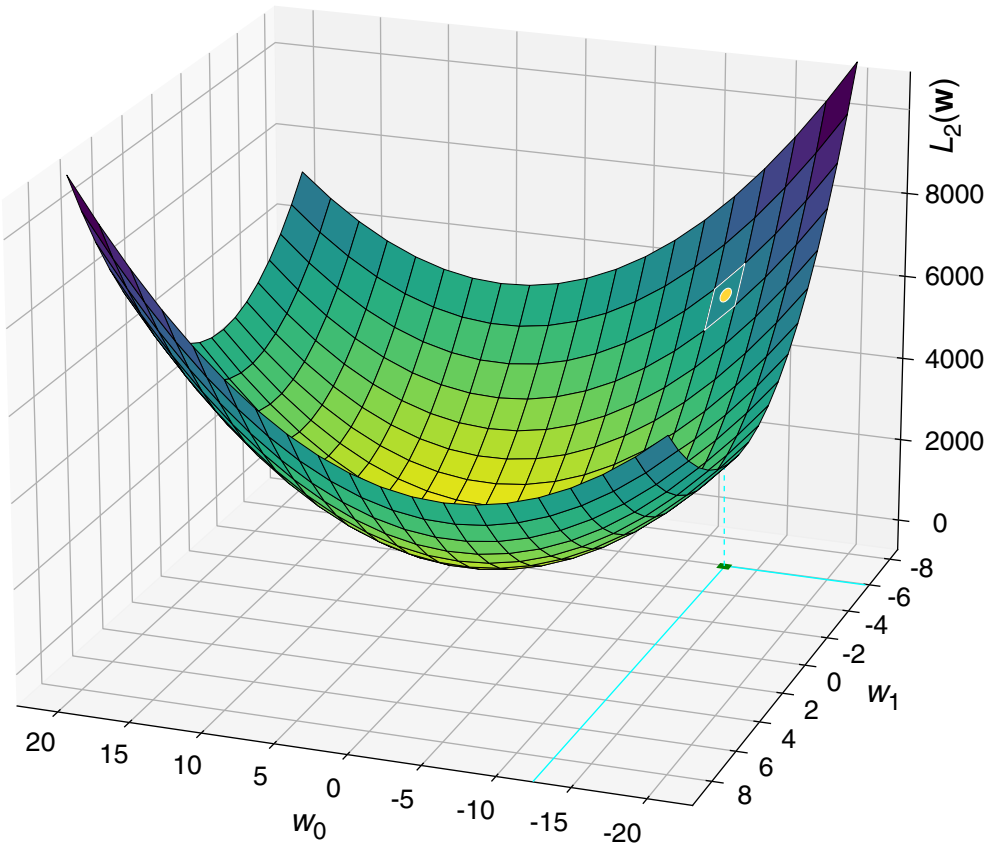
$$L_2(\mathbf{w}) \;=\; \frac{1}{2}\cdot \mathsf{RSS}(\mathbf{w}) \;=\; \frac{1}{2}\cdot \sum_{(\mathbf{x},c)\in D} (c - y(\mathbf{x}))^2$$

[pointwise squared loss]

# Gradient Descent in Detail

## (1) Linear Regression + Squared Loss



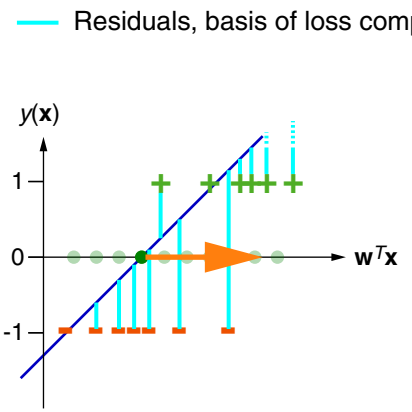

$$y(\mathbf{x}) \overset{(\star)}{=} \mathbf{w}^T\mathbf{x}$$

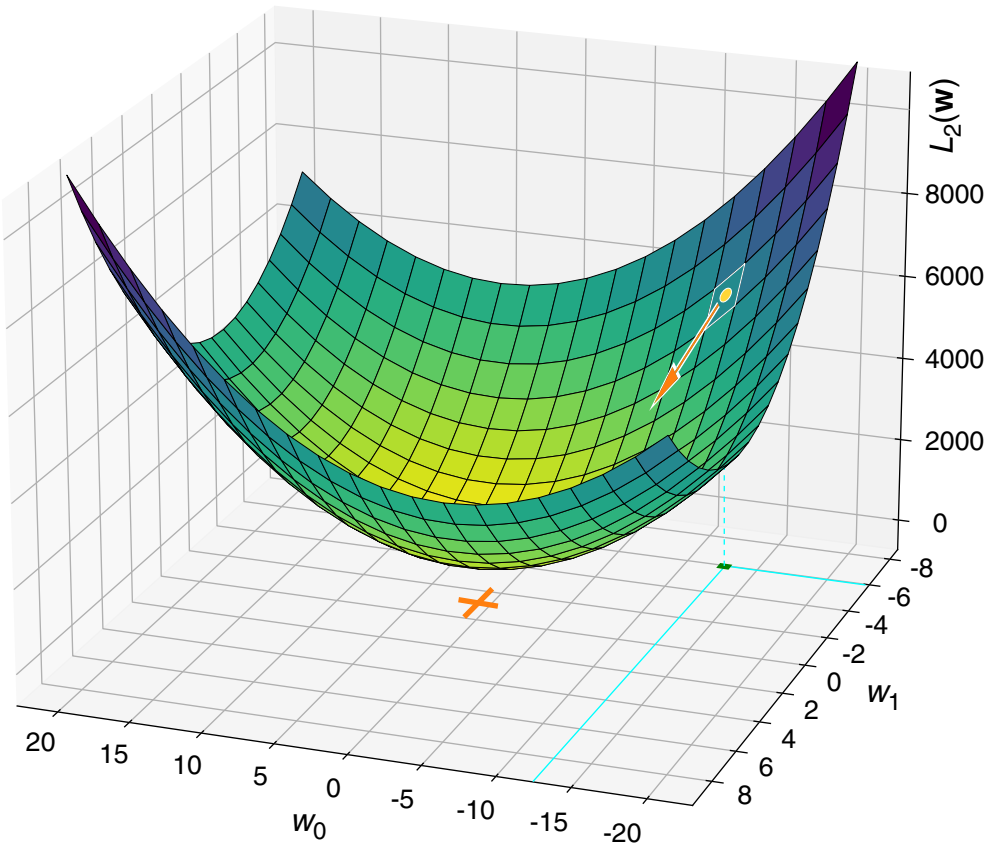
$$L_2(\mathbf{w}) \;=\; \frac{1}{2}\cdot \mathsf{RSS}(\mathbf{w}) \;=\; \frac{1}{2}\cdot \sum_{(\mathbf{x},c)\in D} (c - y(\mathbf{x}))^2$$

[pointwise squared loss]

# Gradient Descent in Detail

## (1) Linear Regression + Squared Loss




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

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

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

# 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:

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

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

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

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

$$\overset{(1)}{=} \sum_{(\mathbf{x},c) \in D} (c - y(\mathbf{x})) \cdot \frac{\partial}{\partial w_j} (c - y(\mathbf{x}))$$

$$= \sum_{(\mathbf{x},c) \in D} (c - \mathbf{w}^T \mathbf{x}) \cdot \frac{\partial}{\partial w_j} (c - \mathbf{w}^T \mathbf{x}) \quad \text{// } j\text{-th summand depends on } w_j.$$

$$= \sum_{(\mathbf{x},c) \in D} (c - \mathbf{w}^T \mathbf{x}) \cdot (-x_j)$$

$$= -\sum_{(\mathbf{x},c) \in D} (c - \mathbf{w}^T \mathbf{x}) \cdot x_j$$

- 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, $\text{BGD}_\sigma$, BGD, IGD, PT]

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

BGD$(D, \eta)$

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. $y(\mathbf{x}) \stackrel{(\star)}{=} \mathbf{w}^T\mathbf{x}$
6. $\delta = c - y(\mathbf{x})$
7. $\Delta\mathbf{w} \stackrel{(\star)}{=} \Delta\mathbf{w} + \eta \cdot \delta \cdot \mathbf{x}$ // $-\delta \cdot \mathbf{x}$ 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*$(D, y(), t)$)
11. *return*$(\mathbf{w})$

# Gradient Descent in Detail

## The BGD Algorithm [algorithms: LMS, $\text{BGD}_\sigma$, BGD, IGD, PT]

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

BGD$(D, \eta)$

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. [ ] Model function evaluation.
6. [ ] Calculation of residual.
7. [ ] Calculation of derivative of the loss, accumulate for $D$.
8. ENDDO
9. [ ] Parameter vector update $\widehat{=}$ one gradient step down.
10. **UNTIL**(*convergence*$(D, y(), t)$)
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 $\eta$ in this direction.

- Recap. The function *convergence*() 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 $\varepsilon$. 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.

# 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 a 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 shown in the following. Moreover, the delta rule forms the basis of the backpropagation algorithm.

- 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 $\eta$ 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:
  - BFGS algorithm (Broyden-Fletcher-Goldfarb-Shanno)
  - L-BFGS algorithm (limited memory BFGS)
  - Conjugate gradient method
  - Newton-Raphson algorithm

# Gradient Descent in Detail

## The IGD Algorithm [algorithms: LMS, $BGD_\sigma$, BGD, IGD, PT]

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

IGD$(D, \eta)$

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

# Gradient Descent in Detail

## The IGD Algorithm [algorithms: LMS, $\text{BGD}_\sigma$, BGD, IGD, PT]

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

IGD$(D, \eta)$

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

Remarks (IGD) :

- The sequence of incremental weight adaptations approximates the gradient descent of the batch approach. If $\eta$ is chosen sufficiently small this approximation can be done at arbitrary precision.
- The computation of the global loss, $L_2(\mathbf{w})$ (RSS), 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.
- A related method to incremental gradient descent is *stochastic* gradient descent, SGD, which estimates the gradient from a randomly selected subset of the data.
- When, as is done here, the squared loss is chosen as loss function, the incremental gradient descent algorithm, IGD, is very similar to the LMS algorithm. The only difference between IGD with squared loss and LMS is that the former exploits the entire example set $D$ while the latter works on a random subset of $D$. I.e., LMS corresponds to an SGD with squared loss.

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.

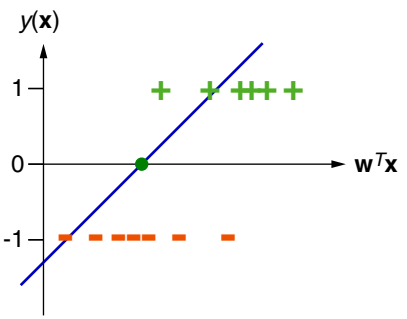
  2. For the evaluation of the effectiveness of a hypothesis, where the proportion of correctly and misclassified examples is analyzed.

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

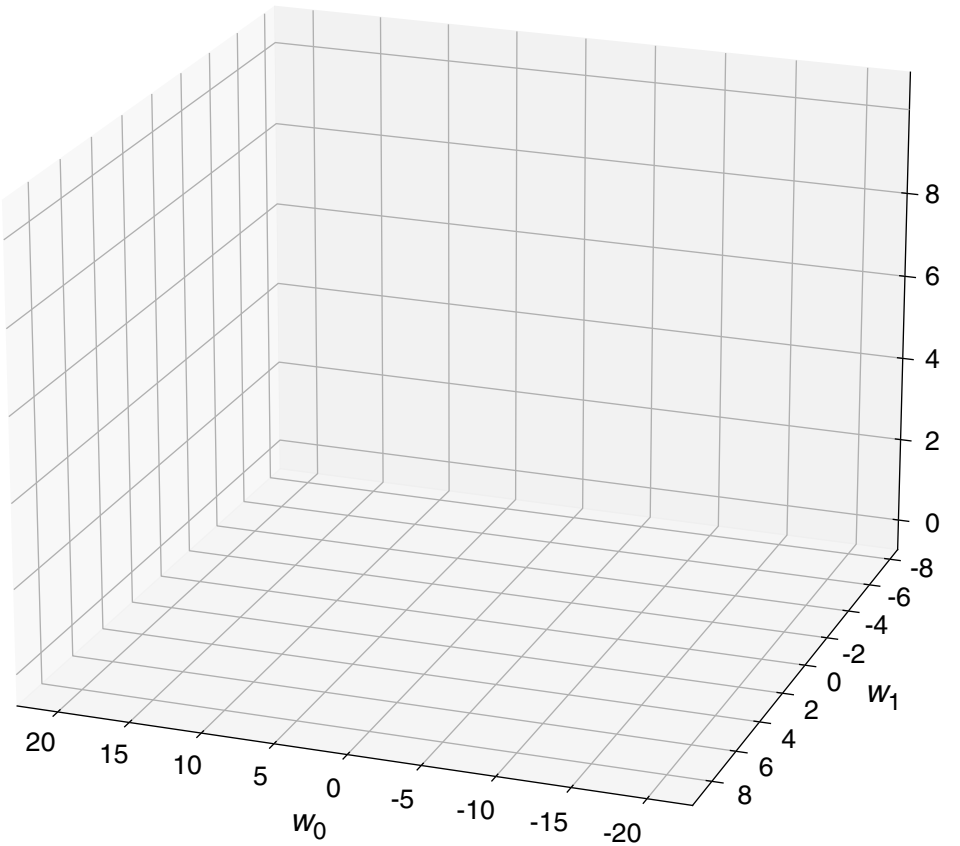
  Typically, fitting (optimization) (1.) and effectiveness evaluation (2.) are done with different loss functions. E.g., logistic regression uses $L_\sigma$ and $L_{0/1}$ for fitting and evaluation respectively. However, linear regression (not classification) uses RSS (the $L_2$ loss) for both fitting 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 [squared loss]

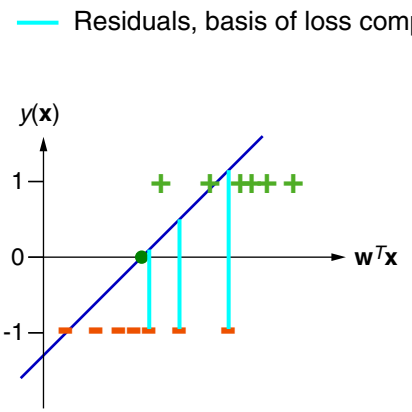


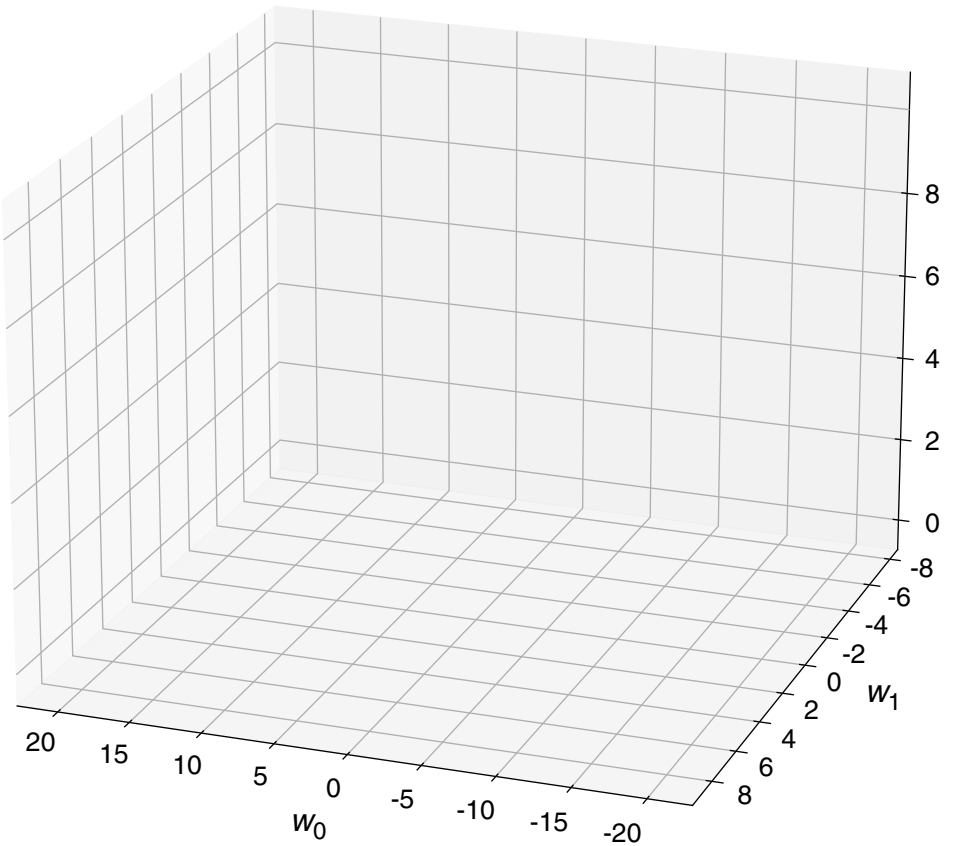

$$y(\mathbf{x}) \overset{(\star)}{=} \mathbf{w}^T\mathbf{x}$$



# Gradient Descent in Detail

## (2) Linear Regression + 0/1 Loss [squared loss]



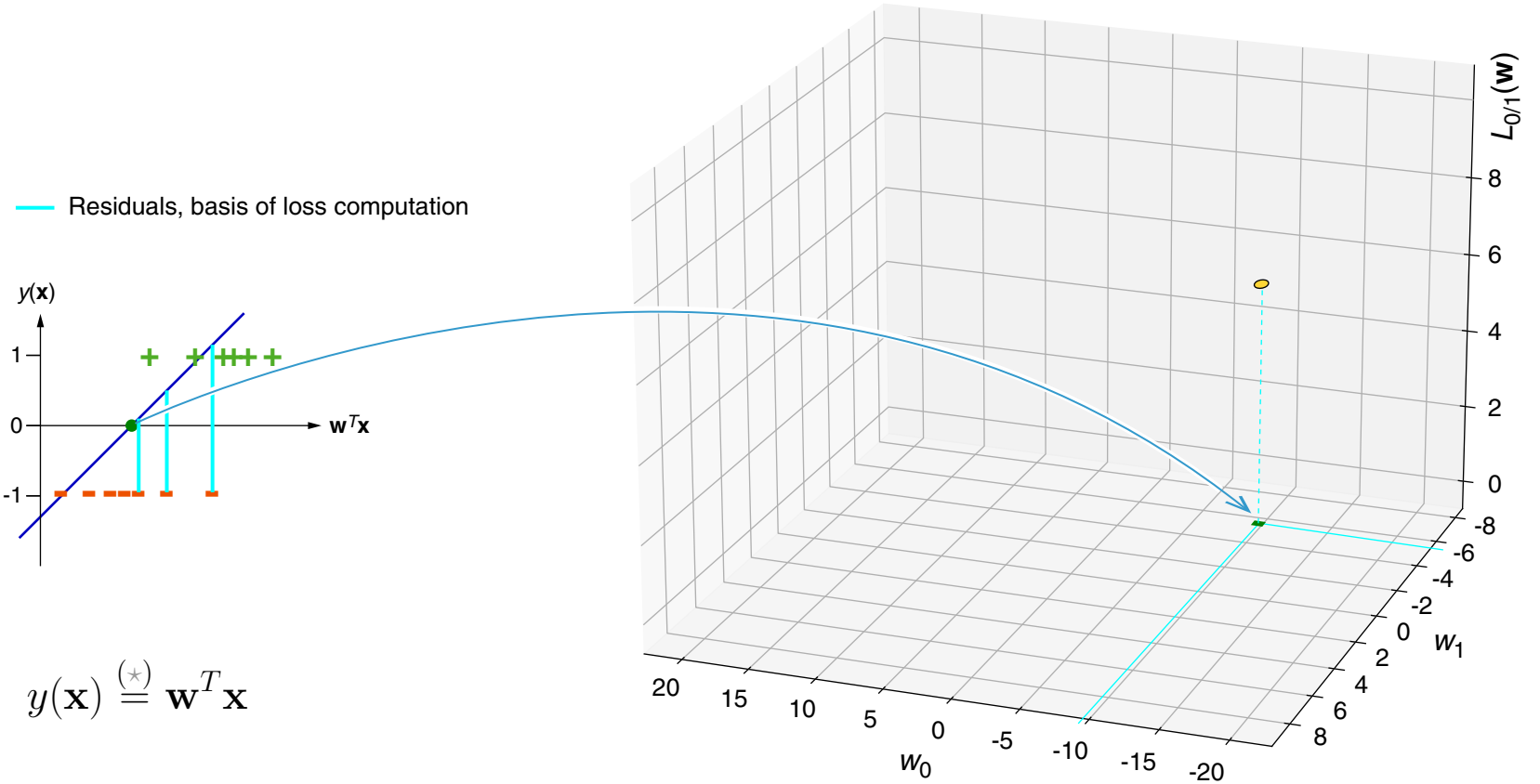




$$y(\mathbf{x}) \overset{(\star)}{=} \mathbf{w}^T\mathbf{x}$$

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

# Gradient Descent in Detail

## (2) Linear Regression + 0/1 Loss [squared loss]

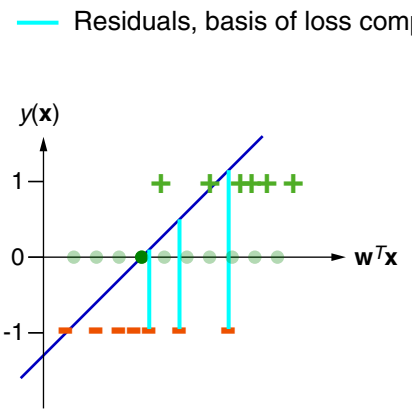



$$y(\mathbf{x}) \overset{(\star)}{=} \mathbf{w}^T\mathbf{x}$$

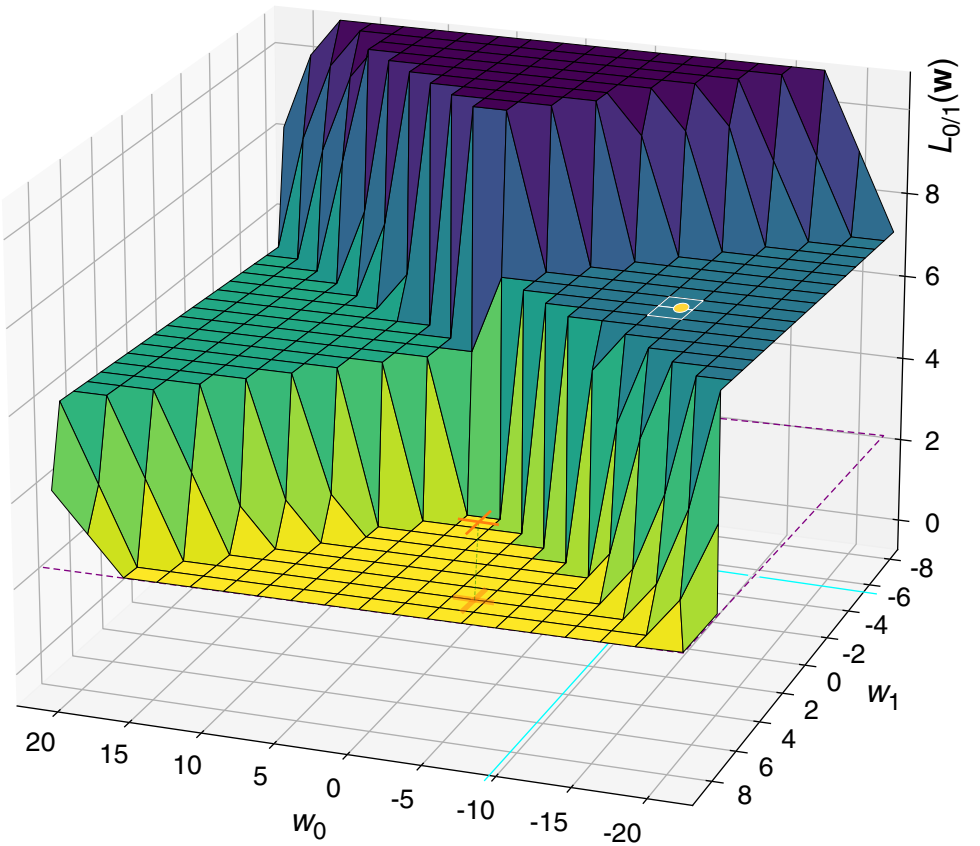
$$L_{0/1}(\mathbf{w}) = \sum_{(\mathbf{x},c)\in D} I_{\neq}(c, \mathsf{sign}(y(\mathbf{x}))) = \sum_{(\mathbf{x},c)\in D} \frac{1}{2} \cdot \left|c - \mathsf{sign}(\mathbf{w}^T\mathbf{x})\right| \quad \text{[pointwise 0/1 loss]}$$

# Gradient Descent in Detail

## (2) Linear Regression + 0/1 Loss [squared loss]




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




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

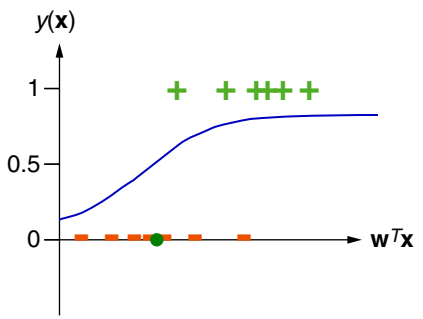
$L_{0/1}(\mathbf{w})$ cannot be expressed as a differentiable function.

Remarks:

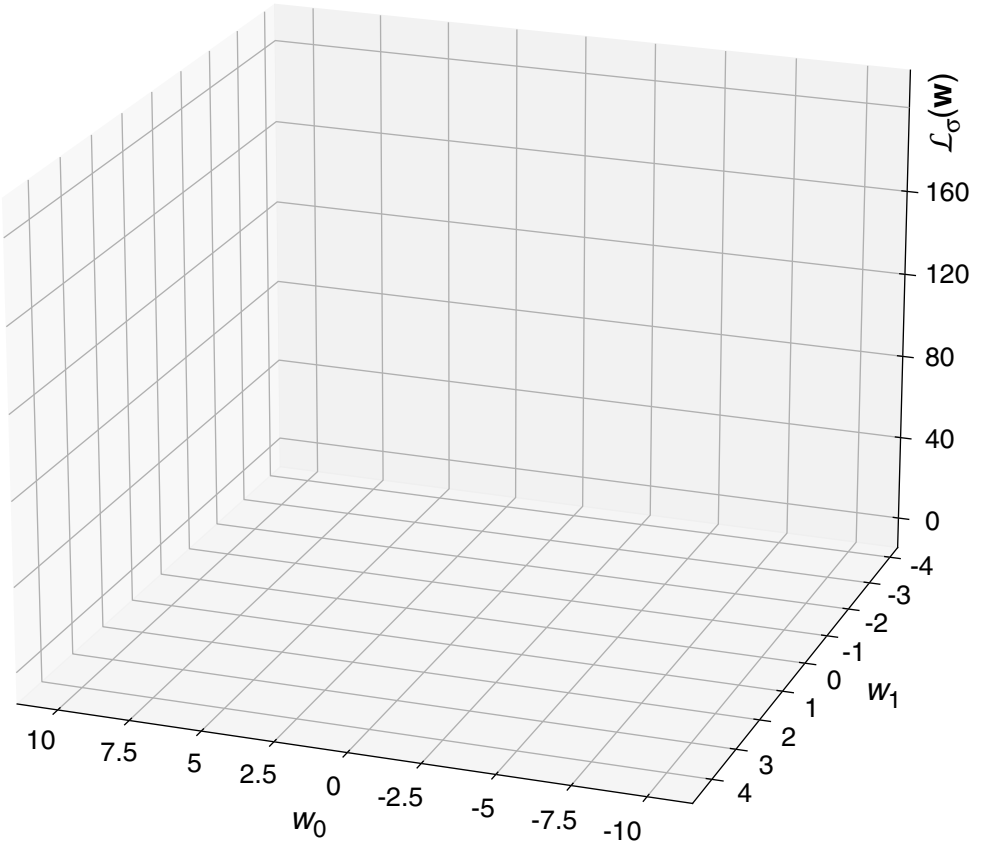
- Since $L_{0/1}(\mathbf{w})$ is not a differentiable function, the gradient descent method cannot be applied to determine its minimum.
- Recap. $I_{\neq}$ is an indicator function that returns 1 if its arguments are *unequal* (and 0 if its arguments are equal).
- Recap. We label $y(0)$ with the "positive"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

# Gradient Descent in Detail

## (3) Logistic Regression + Logistic Loss + Regularization [squared loss]

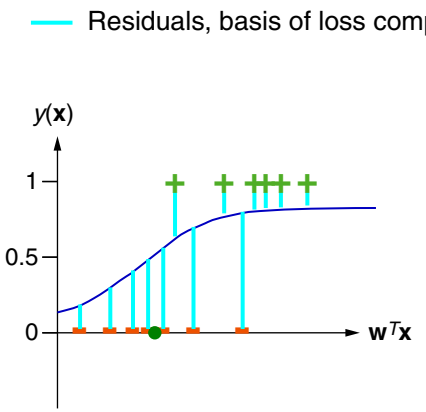


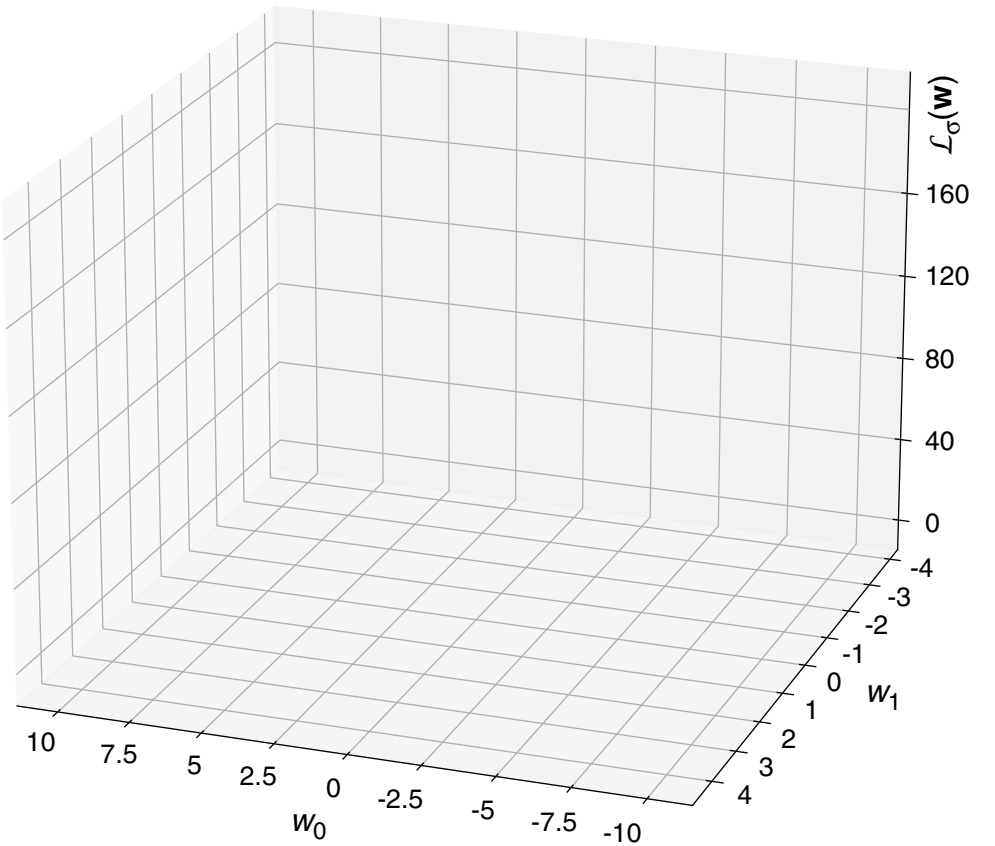

$$y(\mathbf{x}) \overset{(\star)}{=} \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$$



# Gradient Descent in Detail

## (3) Logistic Regression + Logistic Loss + Regularization [squared loss]




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

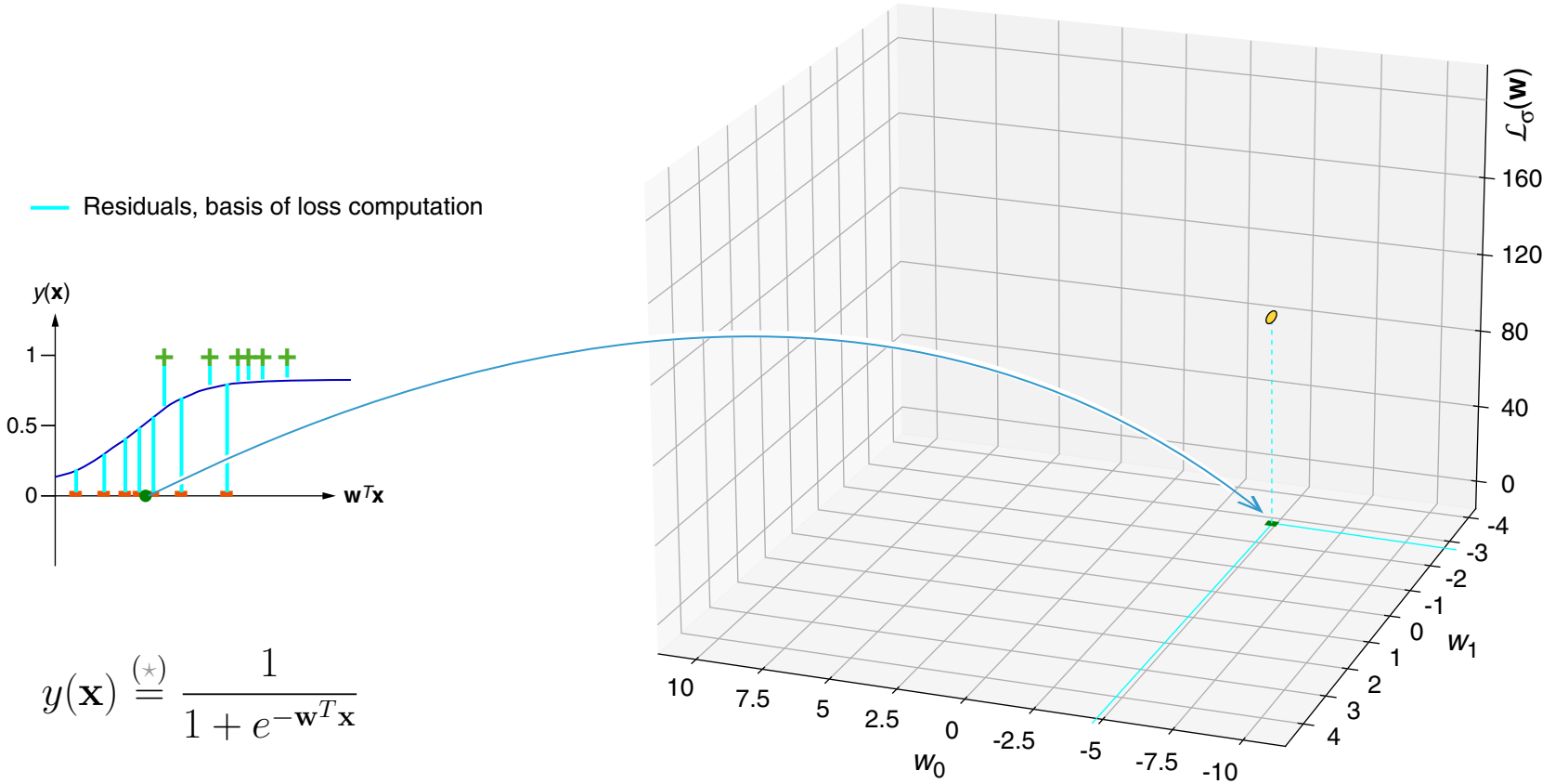
$$\mathcal{L}_\sigma(\mathbf{w}) \;=\; L_\sigma + \lambda{\cdot}R_{\|\vec{\mathbf{w}}\|_2^2} \;=\; \sum_{(\mathbf{x},c)\in D} l_\sigma(c, y(\mathbf{x})) + \lambda{\cdot}\vec{\mathbf{w}}^T\vec{\mathbf{w}}$$

[definitions: $L_\sigma$, $l_\sigma$, $R_{\|\vec{\mathbf{w}}\|_2^2}$]

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

# Gradient Descent in Detail

## (3) Logistic Regression + Logistic Loss + Regularization [squared loss]




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

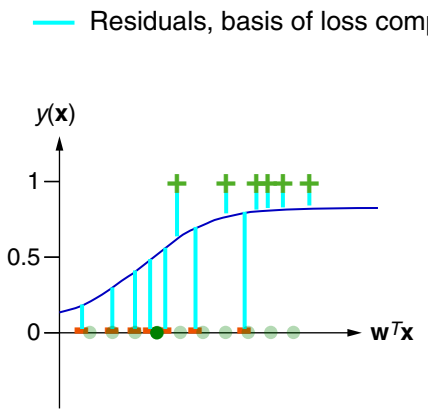
$$\mathcal{L}_\sigma(\mathbf{w}) \;=\; L_\sigma + \lambda{\cdot}R_{\|\vec{\mathbf{w}}\|_2^2} \;=\; \sum_{(\mathbf{x},c)\in D} l_\sigma(c, y(\mathbf{x})) + \lambda{\cdot}\vec{\mathbf{w}}^T\vec{\mathbf{w}}$$

[definitions: $L_\sigma$, $l_\sigma$, $R_{\|\vec{\mathbf{w}}\|_2^2}$]

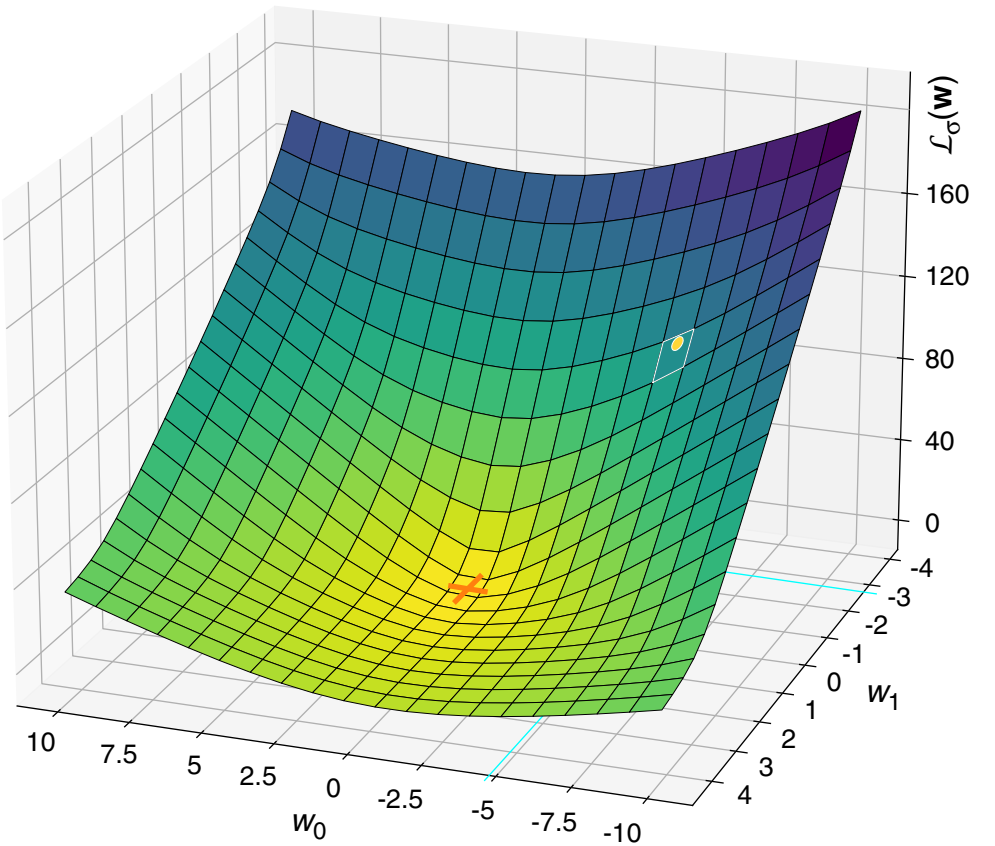
$$= \sum_{(\mathbf{x},c)\in D} -c\cdot\log(y(\mathbf{x})) - (1-c)\cdot\log(1-y(\mathbf{x})) + \lambda{\cdot}\vec{\mathbf{w}}^T\vec{\mathbf{w}}$$

# Gradient Descent in Detail

## (3) Logistic Regression + Logistic Loss + Regularization [squared loss]




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



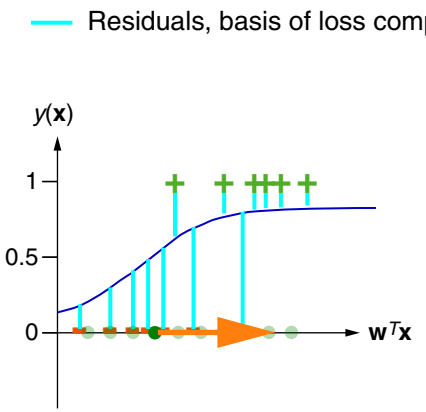

$$\mathcal{L}_\sigma(\mathbf{w}) \;=\; L_\sigma + \lambda{\cdot}R_{\|\vec{\mathbf{w}}\|_2^2} \;=\; \sum_{(\mathbf{x},c)\in D} l_\sigma(c, y(\mathbf{x})) + \lambda{\cdot}\vec{\mathbf{w}}^T\vec{\mathbf{w}}$$

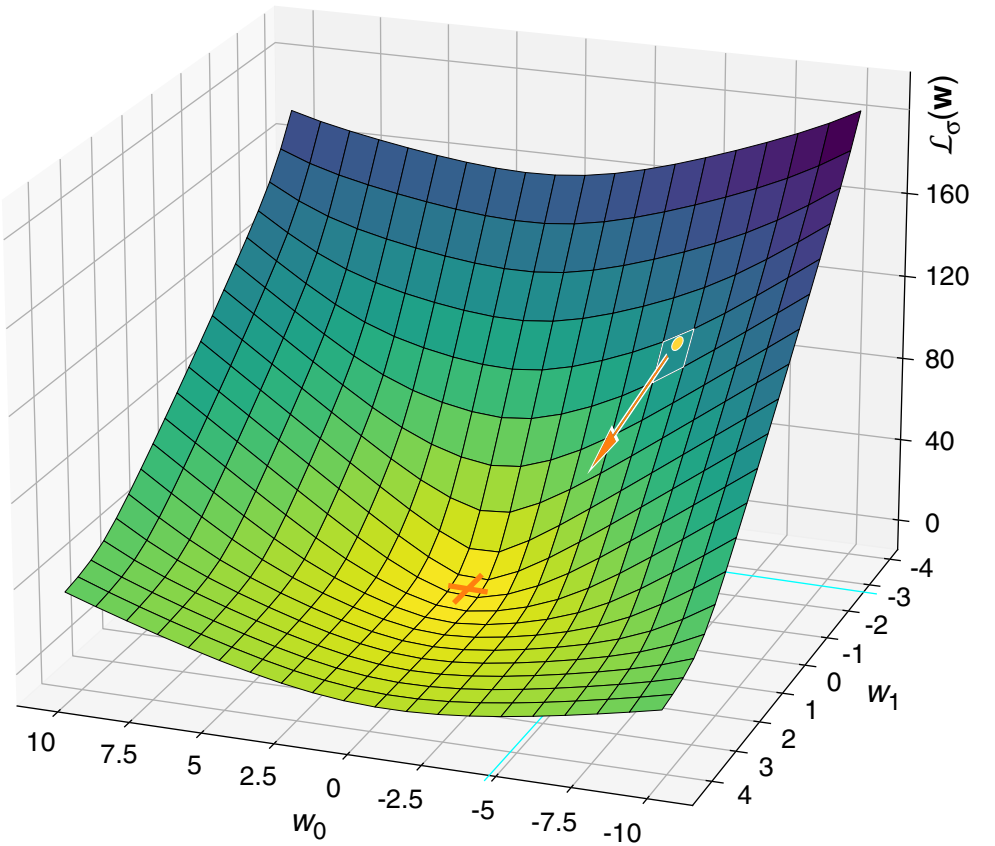
[definitions: $L_\sigma$, $l_\sigma$, $R_{\|\vec{\mathbf{w}}\|_2^2}$]

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

# Gradient Descent in Detail

## (3) Logistic Regression + Logistic Loss + Regularization [squared loss]




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

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

[definitions: $L_\sigma$, $l_\sigma$, $R_{\|\vec{\mathbf{w}}\|_2^2}$]

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

# 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:

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

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 $w_j,\ j = 0, \ldots, p$:

$$
\begin{aligned}
\frac{\partial}{\partial w_j}\mathcal{L}_\sigma(\mathbf{w}) &= \frac{\partial}{\partial w_j} L_\sigma(\mathbf{w}) \ + \ \frac{\partial}{\partial w_j} \lambda \cdot R_{\|\vec{\mathbf{w}}\|_2^2}(\mathbf{w}) \\
&= \sum_{(\mathbf{x},c)\in D} \frac{\partial}{\partial w_j} l_\sigma(c, y(\mathbf{x})) \ + \ \lambda \cdot \frac{\partial}{\partial w_j} \vec{\mathbf{w}}^T \vec{\mathbf{w}} \\
&= \sum_{(\mathbf{x},c)\in D} \frac{\partial}{\partial w_j} \left[ -c \cdot \log(\sigma(\mathbf{w}^T\mathbf{x})) - (1-c) \cdot \log(1 - \sigma(\mathbf{w}^T\mathbf{x})) \right] \ + \ \lambda \cdot \frac{\partial}{\partial w_j} \sum_{i=1}^{p} w_i^2 \\
&= \sum_{(\mathbf{x},c)\in D} \left[ -c \cdot \frac{\partial}{\partial w_j} \log(\sigma(\mathbf{w}^T\mathbf{x})) - (1-c) \cdot \frac{\partial}{\partial w_j} \log(1 - \sigma(\mathbf{w}^T\mathbf{x})) \right] \ \overset{(1)}{+} \ \lambda \cdot \frac{\partial}{\partial w_j} w_j^2 \\
&\overset{(2)}{=} \sum_{(\mathbf{x},c)\in D} \left[ -c \cdot \frac{1}{\sigma(\mathbf{w}^T\mathbf{x})} \cdot \frac{\partial}{\partial w_j} \sigma(\mathbf{w}^T\mathbf{x}) \right. \\
&\qquad \left. - (1-c) \cdot \frac{1}{1 - \sigma(\mathbf{w}^T\mathbf{x})} \cdot \left( - \frac{\partial}{\partial w_j} \sigma(\mathbf{w}^T\mathbf{x}) \right) \right] \ \overset{(1)}{+} \ 2\lambda \cdot w_j \\
&= \ \hookrightarrow \text{p. 159}
\end{aligned}
$$

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