GHENT UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

E016350 - Artificial Intelligence

Lecture 3

Machine learning

Optimization Algorithms and Logistic Regression

Aleksandra Pizurica

Ghent University
Spring 2024

Outline

These slides are based on: Andrew Ng and Tengyu Ma: Lecture Notes (CS229) Machine learning https://cs229.stanford.edu/main_notes.pdf and Moses Charikar and Sanmi Koyejo: (CS221) Artificial Intelligence: Principles and Techniques (Stanford)

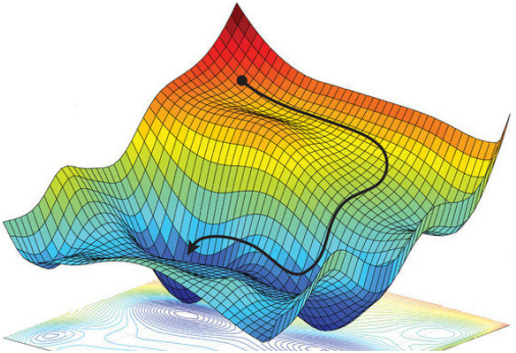
Outline

Optimization in machine learning

Our learning task: determine the parameters (weights) $\mathbf{w}$ of a hypothesis $h_{\mathbf{w}}(\mathbf{x})$ that approximates true, unknown function $y = f(\mathbf{x})$.

To find the optimal $\mathbf{w}$ we minimize a loss function.

– Why using gradient descent?

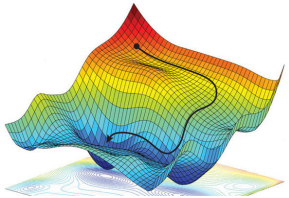


Picture credit: N. Azizan R. and B. Hassibi. Stochastic Gradient/Mirror Descent: Minimax Optimality and Implicit Regularization. ICLR 2019.

What is the Steepest Direction?

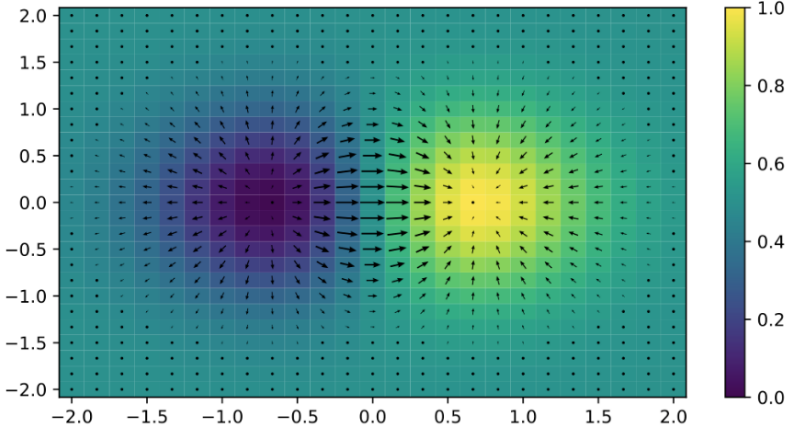
Goal: take a step $\Delta : \Delta_1^2 + \Delta_2^2 < \epsilon$ such that $Loss(\mathbf{w} + \Delta) \leq Loss(\mathbf{w})$

$$\min_{\Delta:\Delta_1^2+\Delta_2^2<\epsilon} Loss(\mathbf{w}+\Delta); \qquad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}; \qquad \Delta = \begin{bmatrix} \Delta_1 \\ \Delta_2 \end{bmatrix}$$



- First-order Taylor expansion
  $Loss(\mathbf{w}+\Delta) \approx Loss(\mathbf{w}) + \frac{\partial Loss}{\partial w_1}\Big|_{\mathbf{w}} \Delta_1 + \frac{\partial Loss}{\partial w_2}\Big|_{\mathbf{w}} \Delta_2 = Loss(\mathbf{w}) + \Delta^\top \nabla_{\mathbf{w}} Loss(\mathbf{w})$
- So, for maximum leverage out of moving along Δ:
  align Δ with $-\nabla_{\mathbf{w}} Loss(\mathbf{w})$
- I.e., steepest direction (down) = (negative) gradient direction in a given point

A visualization of the gradient field



The gradient of a scalar-valued differentiable function of several variables is the vector field whose value at a point p gives the direction and the rate of fastest increase at that point.
https://en.wikipedia.org/wiki/Gradient

Gradient in d dimensions

Same reasoning in arbitrary number of directions

In d dimensions, $\mathbf{w} \in \mathbb{R}^d$ and the gradient of the loss function in particular $\mathbf{w}$ point:

$$\nabla_{\mathbf{w}} Loss(\mathbf{w}) = \begin{bmatrix} \frac{\partial Loss}{\partial w_1}(\mathbf{w}) \\ \frac{\partial Loss}{\partial w_2}(\mathbf{w}) \\ . \\ . \\ . \\ \frac{\partial Loss}{\partial w_d}(\mathbf{w}) \end{bmatrix}$$

Weight optimization

We optimize the $\mathbf{w} \in \mathbb{R}^d$ by applying the gradient descent to the training loss:

$$w_j \leftarrow w_j - \alpha \frac{\partial}{\partial w_j} TrainLoss(w_1, ..., w_d)$$

Sometimes we write this more compactly as

$$w_j \leftarrow w_j - \alpha \frac{\partial}{\partial w_j} TrainLoss(\mathbf{w})$$

or in a vector form

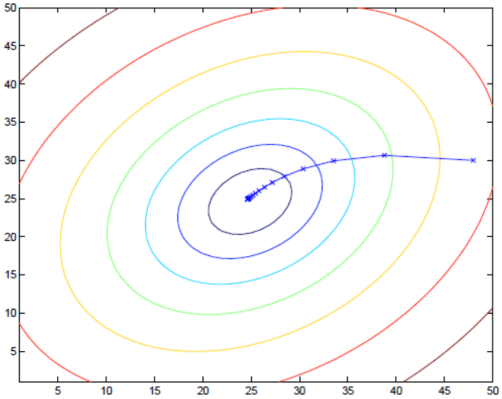
$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} TrainLoss(\mathbf{w})$$

- Perform update in downhill direction for each coordinate
- The steeper the slope (i.e. the larger the magnitude of the derivative) the bigger the step for that coordinate.

Gradient descent algorithm

Idea:

- Start somewhere
- Repeat: Take a step in the gradient direction



Credit: Stanford CS229 Course Notes. Trajectory for gradient descent is like climbing down into a valley

Optimization procedure: Gradient Descent (GD)

$$TrainLoss(\mathbf{w}) = \frac{1}{|\mathcal{D}_{train}|} \sum_{(x,y)\in\mathcal{D}_{train}} L(x, y, \mathbf{w})$$

- init $\mathbf{w} = [0, \ldots, 0]$
- for iter $1, 2, ...$

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} TrainLoss(\mathbf{w})$$

- α: learning rate — tweaking parameter that needs to be chosen carefully
- How? Try multiple choices
  - Crude rule of thumb: update changes $\mathbf{w}$ about 0.1–1%

Influence of the learning rate

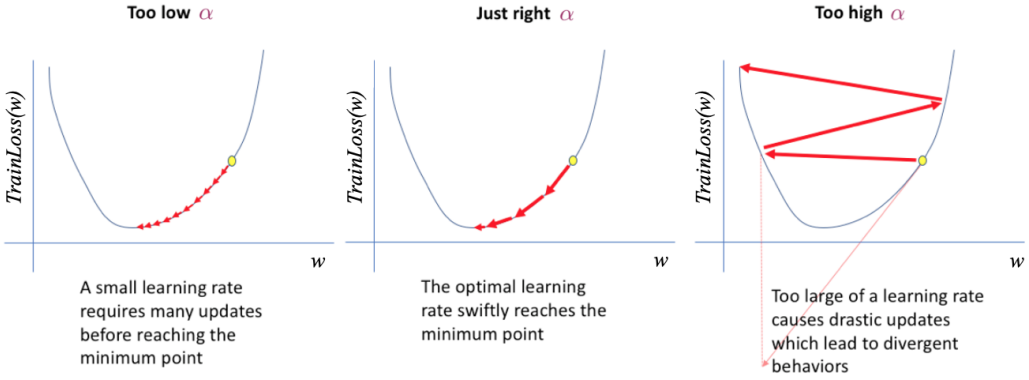


Illustration Credit: Edouard Duchesnay, NeuroSpin CEA Université Paris-Saclay, France.

Stochastic Gradient Descent (SGD)

$$TrainLoss(\mathbf{w}) = \frac{1}{|\mathcal{D}_{train}|} \sum_{(x,y)\in\mathcal{D}_{train}} L(x, y, \mathbf{w})$$

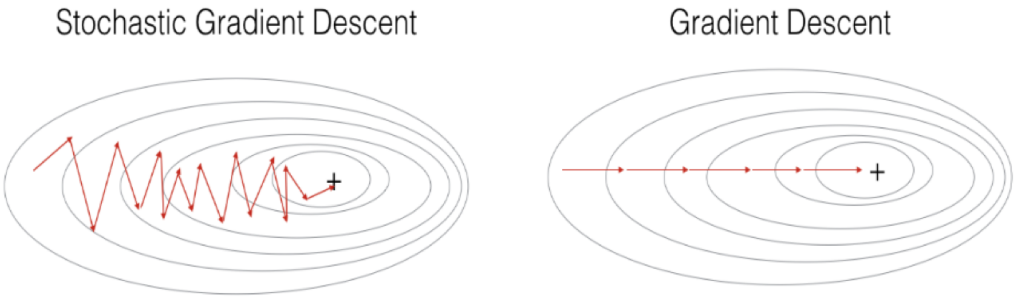
- init $\mathbf{w} = [0, \ldots, 0]$
- for iter $1, 2, \ldots$
  - For $(x, y) \in \mathcal{D}_{train}$:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} L(x, y, \mathbf{w})$$

Motivation:

- Gradient descent algorithm is slow: going over all the training examples in each iteration is expensive when lots of data!
- Rather than looping through all the training examples to compute a single gradient, update the weights based on each example $\rightarrow$ SGD

SGD vs. GD



"+" denotes a minimum of the cost. SGD leads to many oscillations to reach convergence. But each step is a lot faster to compute for SGD than for GD, as it uses only one training example (vs. the whole batch for GD).

Illustration Credit: Ankit-AI - Sharing AI: Optimization Algorithms for Machine Learning.

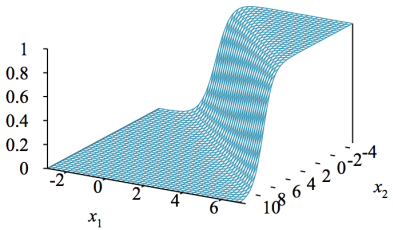
Outline

Logistic regression - Reminder

We consider binary classification: to each input data point $\mathbf{x} \in \mathbb{R}^d$ we assign a class label $y \in \{0, 1\}$.

Let $g(z)$ denote the logistic (sigmoid) function:

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



For some weight vector $\mathbf{w} \in \mathbb{R}^d$, the hypothesis

$$h_{\mathbf{w}}(\mathbf{x}) = g(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}}$$

can be interpreted as the probability that $\mathbf{x}$ belongs to class 1, i.e., the probability that $y = 1$.

Logistic regression - Probabilistic interpretation formally

We said: $h_{\mathbf{w}}(\mathbf{x}) = g(\mathbf{w} \cdot \mathbf{x})$ can be interpreted as the probability that $y = 1$. Formally:

$$\begin{aligned} P(y = 1|\mathbf{x}, \mathbf{w}) &= h_{\mathbf{w}}(\mathbf{x}) \\ P(y = 0|\mathbf{x}, \mathbf{w}) &= 1 - h_{\mathbf{w}}(\mathbf{x}) \end{aligned}$$

Logistic regression - Probabilistic interpretation formally

We said: $h_{\mathbf{w}}(\mathbf{x}) = g(\mathbf{w} \cdot \mathbf{x})$ can be interpreted as the probability that $y = 1$. Formally:

$$\begin{aligned} P(y = 1|\mathbf{x}, \mathbf{w}) &= h_{\mathbf{w}}(\mathbf{x}) \\ P(y = 0|\mathbf{x}, \mathbf{w}) &= 1 - h_{\mathbf{w}}(\mathbf{x}) \end{aligned}$$

Or, more compactly:

$$P(y|\mathbf{x}, \mathbf{w}) = (h_{\mathbf{w}}(\mathbf{x}))^{y}(1 - h_{\mathbf{w}}(\mathbf{x}))^{(1-y)}$$

Logistic regression - Probabilistic interpretation formally

We said: $h_{\mathbf{w}}(\mathbf{x}) = g(\mathbf{w} \cdot \mathbf{x})$ can be interpreted as the probability that $y = 1$. Formally:

$$\begin{aligned} P(y = 1|\mathbf{x}, \mathbf{w}) &= h_{\mathbf{w}}(\mathbf{x}) \\ P(y = 0|\mathbf{x}, \mathbf{w}) &= 1 - h_{\mathbf{w}}(\mathbf{x}) \end{aligned}$$

Or, more compactly:

$$P(y|\mathbf{x}, \mathbf{w}) = (h_{\mathbf{w}}(\mathbf{x}))^y (1 - h_{\mathbf{w}}(\mathbf{x}))^{(1-y)}$$

If the training examples were generated independently, the likelihood of the weights is:

$$\mathcal{L}(\mathbf{w}) = \prod_{i=1}^{N} P(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{w}) = \prod_{i=1}^{N} \left(h_{\mathbf{w}}(\mathbf{x}^{(i)}) \right)^{y^{(i)}} \left(1 - h_{\mathbf{w}}(\mathbf{x}^{(i)}) \right)^{1-y^{(i)}}$$

Logistic regression - Probabilistic interpretation formally

We said: $h_{\mathbf{w}}(\mathbf{x}) = g(\mathbf{w} \cdot \mathbf{x})$ can be interpreted as the probability that $y = 1$. Formally:

$$\begin{aligned} P(y = 1|\mathbf{x}, \mathbf{w}) &= h_{\mathbf{w}}(\mathbf{x}) \\ P(y = 0|\mathbf{x}, \mathbf{w}) &= 1 - h_{\mathbf{w}}(\mathbf{x}) \end{aligned}$$

Or, more compactly:

$$P(y|\mathbf{x}, \mathbf{w}) = (h_{\mathbf{w}}(\mathbf{x}))^y (1 - h_{\mathbf{w}}(\mathbf{x}))^{(1-y)}$$

If the training examples were generated independently, the likelihood of the weights is:

$$\mathcal{L}(\mathbf{w}) = \prod_{i=1}^{N} P(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{w}) = \prod_{i=1}^{N} \left(h_{\mathbf{w}}(\mathbf{x}^{(i)}) \right)^{y^{(i)}} \left(1 - h_{\mathbf{w}}(\mathbf{x}^{(i)}) \right)^{1-y^{(i)}}$$

it is easier to maximize the log likelihood:

$$\ell(\mathbf{w}) = \log \mathcal{L}(\mathbf{w}) = \sum_{i=1}^{N} y^{(i)} \log h_{\mathbf{w}}(\mathbf{x}^{(i)}) \; + \; (1 - y^{(i)}) \log(1 - h_{\mathbf{w}}(\mathbf{x}^{(i)})$$

Logistic regression under the maximum likelihood optimization

Now we can determine the update rule for the logistic regression by maximizing the log-likelihood of the weights. This is the most common form of the logistic regression.

Logistic regression under the maximum likelihood optimization

Now we can determine the update rule for the logistic regression by maximizing the log-likelihood of the weights. This is the most common form of the logistic regression.

Note that now $TrainLoss(\mathbf{w}) = -\ell(\mathbf{w})$, so we are applying the gradient descent algorithm to $-\ell(\mathbf{w})$, or equivalently, we are applying the **gradient ascent** to $\ell(\mathbf{w})$:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \nabla_{\mathbf{w}} \ell(\mathbf{w})$$

Logistic regression under the maximum likelihood optimization

Now we can determine the update rule for the logistic regression by maximizing the log-likelihood of the weights. This is the most common form of the logistic regression.

Note that now $TrainLoss(\mathbf{w}) = -\ell(\mathbf{w})$, so we are applying the gradient descent algorithm to $-\ell(\mathbf{w})$, or equivalently, we are applying the **gradient ascent** to $\ell(\mathbf{w})$:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \nabla_{\mathbf{w}} \ell(\mathbf{w})$$

We start with one training example $(\mathbf{x}, y)$:

$$\frac{\partial}{\partial w_j}\ell(\mathbf{w}) = \Big(y\frac{1}{g(\mathbf{w}\cdot\mathbf{x})} - (1-y)\frac{1}{1-g(\mathbf{w}\cdot\mathbf{x})}\Big)\frac{\partial}{\partial w_j}g(\mathbf{w}\cdot\mathbf{x})$$

Logistic regression under the maximum likelihood optimization

Now we can determine the update rule for the logistic regression by maximizing the log-likelihood of the weights. This is the most common form of the logistic regression.

Note that now $TrainLoss(\mathbf{w}) = -\ell(\mathbf{w})$, so we are applying the gradient descent algorithm to $-\ell(\mathbf{w})$, or equivalently, we are applying the **gradient ascent** to $\ell(\mathbf{w})$:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \nabla_{\mathbf{w}} \ell(\mathbf{w})$$

We start with one training example $(\mathbf{x}, y)$:

$$\begin{aligned}
\frac{\partial}{\partial w_j}\ell(\mathbf{w}) &= \Big(y\frac{1}{g(\mathbf{w}\cdot\mathbf{x})} - (1-y)\frac{1}{1-g(\mathbf{w}\cdot\mathbf{x})}\Big)\frac{\partial}{\partial w_j} g(\mathbf{w}\cdot\mathbf{x}) \\
&= \Big(y\frac{1}{g(\mathbf{w}\cdot\mathbf{x})} - (1-y)\frac{1}{1-g(\mathbf{w}\cdot\mathbf{x})}\Big) g(\mathbf{w}\cdot\mathbf{x})(1-g(\mathbf{w}\cdot\mathbf{x}))\frac{\partial}{\partial w_j}(\mathbf{w}\cdot\mathbf{x})
\end{aligned}$$

Logistic regression under the maximum likelihood optimization

Now we can determine the update rule for the logistic regression by maximizing the log-likelihood of the weights. This is the most common form of the logistic regression.

Note that now $TrainLoss(\mathbf{w}) = -\ell(\mathbf{w})$, so we are applying the gradient descent algorithm to $-\ell(\mathbf{w})$, or equivalently, we are applying the **gradient ascent** to $\ell(\mathbf{w})$:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \nabla_{\mathbf{w}} \ell(\mathbf{w})$$

We start with one training example $(\mathbf{x}, y)$:

$$\begin{aligned}
\frac{\partial}{\partial w_j}\ell(\mathbf{w}) &= \Big(y\frac{1}{g(\mathbf{w}\cdot\mathbf{x})} - (1-y)\frac{1}{1-g(\mathbf{w}\cdot\mathbf{x})}\Big)\frac{\partial}{\partial w_j} g(\mathbf{w}\cdot\mathbf{x}) \\
&= \Big(y\frac{1}{g(\mathbf{w}\cdot\mathbf{x})} - (1-y)\frac{1}{1-g(\mathbf{w}\cdot\mathbf{x})}\Big) g(\mathbf{w}\cdot\mathbf{x})(1-g(\mathbf{w}\cdot\mathbf{x}))\frac{\partial}{\partial w_j}(\mathbf{w}\cdot\mathbf{x}) \\
&= (y(1-g(\mathbf{w}\cdot\mathbf{x})) \;-\; (1-y)g(\mathbf{w}\cdot\mathbf{x}))x_j
\end{aligned}$$

Logistic regression under the maximum likelihood optimization

Now we can determine the update rule for the logistic regression by maximizing the log-likelihood of the weights. This is the most common form of the logistic regression.

Note that now $TrainLoss(\mathbf{w}) = -\ell(\mathbf{w})$, so we are applying the gradient descent algorithm to $-\ell(\mathbf{w})$, or equivalently, we are applying the **gradient ascent** to $\ell(\mathbf{w})$:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \nabla_{\mathbf{w}} \ell(\mathbf{w})$$

We start with one training example $(\mathbf{x}, y)$:

$$\begin{aligned}
\frac{\partial}{\partial w_j}\ell(\mathbf{w}) &= \Big(y\frac{1}{g(\mathbf{w}\cdot\mathbf{x})} - (1-y)\frac{1}{1-g(\mathbf{w}\cdot\mathbf{x})}\Big)\frac{\partial}{\partial w_j} g(\mathbf{w}\cdot\mathbf{x}) \\
&= \Big(y\frac{1}{g(\mathbf{w}\cdot\mathbf{x})} - (1-y)\frac{1}{1-g(\mathbf{w}\cdot\mathbf{x})}\Big) g(\mathbf{w}\cdot\mathbf{x})(1-g(\mathbf{w}\cdot\mathbf{x}))\frac{\partial}{\partial w_j}(\mathbf{w}\cdot\mathbf{x}) \\
&= (y(1-g(\mathbf{w}\cdot\mathbf{x})) \;-\; (1-y)g(\mathbf{w}\cdot\mathbf{x}))x_j \\
&= (y - h_{\mathbf{w}}(\mathbf{x}))x_j
\end{aligned}$$

Logistic regression under the maximum likelihood optimization

The maximum likelihood estimation gives us the following update rule

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \nabla_{\mathbf{w}} \ell(\mathbf{w})$$

where for one training example we derived

$$\frac{\partial}{\partial w_j} \ell(\mathbf{w}) \quad = \quad (y - h_{\mathbf{w}}(\mathbf{x})) x_j$$

Hence, the MLE update rule for the logistic regression, with one example, is

$$w_j \leftarrow w_j + \alpha (y - h_{\mathbf{w}}(\mathbf{x})) x_j$$

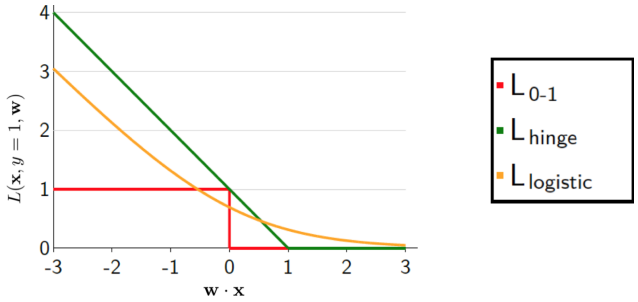
and with all training examples

$$w_j \leftarrow w_j + \alpha \sum_{(\mathbf{x},y) \in \mathcal{D}_{train}} (y - h_{\mathbf{w}}(\mathbf{x})) x_j$$

Note: Looks exactly the same as for the least-squares linear regression ($h_{\mathbf{w}}$ is different)

Classification losses

Some common losses for binary classification (shown for $y = 1$).



- **Logistic** loss:
  $L_{\text{logistic}}(\mathbf{x}, y, \mathbf{w}) = -\ell(\mathbf{w}) = -y \log h_{\mathbf{w}}(\mathbf{x}) - (1 - y) \log(1 - h_{\mathbf{w}}(\mathbf{x}))$
  - For $y = 1$, this becomes: $L_{\text{logistic}}(\mathbf{x}, y = 1, \mathbf{w}) = \log(1 + e^{-\mathbf{w} \cdot \mathbf{x}})$
- **Hinge** loss:
  - $L_{\text{hinge}}(\mathbf{x}, y = 1, \mathbf{w}) = \max(1 - \mathbf{w} \cdot \mathbf{x}, 0)$ (and $L_{\text{hinge}}(\mathbf{x}, 0, \mathbf{w}) = \max(1 + \mathbf{w} \cdot \mathbf{x}, 0)$)
  - This loss is often used with support vector machines (SVM)

Next time

- Multiclass logistic regression
- Learning with non-linear features
- Decision trees