3F8: Inference

Classification

José Miguel Hernández–Lobato and Richard E. Turner
Department of Engineering
University of Cambridge

Lent Term

What is classification?

It is the same as regression, but with **discrete outputs**: $y_n \in \{1, \ldots, C\}$, where C is the number of **classes**. Often $C = 2$.

Same goals as in regression. The patterns to identify consists of

- A partition of the input space into C **decision regions**, one for each class.
- Each new input is assigned the class of its corresponding decision region.
- We would also like a measure of **confidence** (probability) in the decisions.

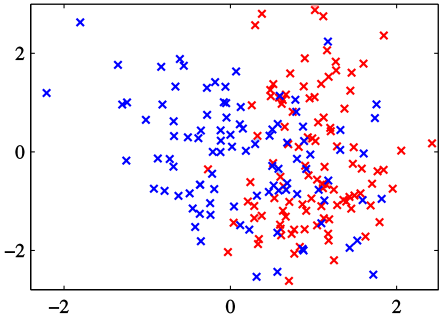


Figure: C. Bishop. *Pattern Recognition and Machine Learning*, 2006.

What is classification?

It is the same as regression, but with **discrete outputs**: $y_n \in \{1, \dots, C\}$, where C is the number of **classes**. Often $C = 2$.

Same goals as in regression. The patterns to identify consists of

- A partition of the input space into C **decision regions**, one for each class.
- Each new input is assigned the class of its corresponding decision region.
- We would also like a measure of **confidence** (probability) in the decisions.

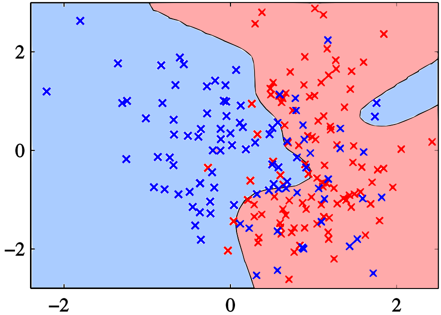


Figure: C. Bishop. *Pattern Recognition and Machine Learning*, 2006.

What is classification?

It is the same as regression, but with **discrete outputs**: $y_n \in \{1, \ldots, C\}$, where C is the number of **classes**. Often $C = 2$.

Same goals as in regression. The patterns to identify consists of

- A partition of the input space into C **decision regions**, one for each class.
- Each new input is assigned the class of its corresponding decision region.
- We would also like a measure of **confidence** (probability) in the decisions.

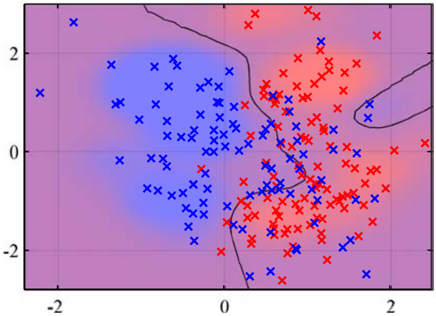
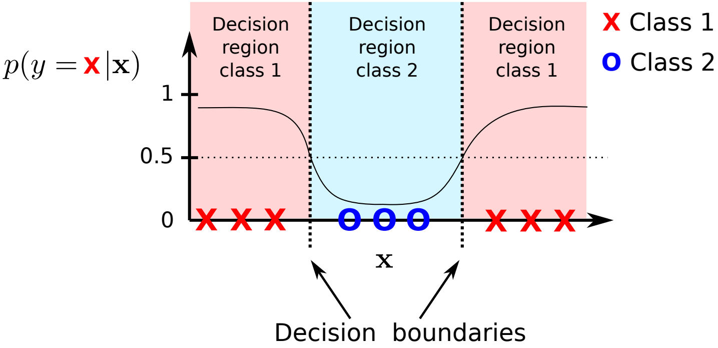


Figure: C. Bishop. *Pattern Recognition and Machine Learning*, 2006.

1D example

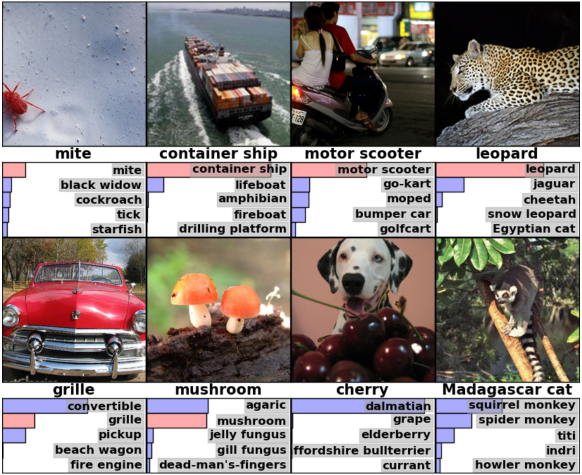
The decision regions are separated by **decision boundaries**.

These are points at which two classes have **equal predictive probability**.



Real-world example

ImageNet: about 22,000 classes and 15 million high-resolution images.



A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

Why not use methods for regression?

Let the output $\mathbf{y}_n$ be a C-dimensional vector with a **one-hot-encoding** of the class for $\widetilde{\mathbf{x}}_n$ ($y_{n,c} = 1$ if the class is c and $y_{n,c} = 0$, otherwise).

We can then solve C linear regression problems, one for each class:

$$\mathbf{W} = \left(\widetilde{\mathbf{X}}^{\mathsf{T}}\widetilde{\mathbf{X}}\right)^{-1}\widetilde{\mathbf{X}}^{\mathsf{T}}\mathbf{Y}\,,$$

with $\mathbf{Y} = (\mathbf{y}_1; \ldots; \mathbf{y}_N)^{\mathsf{T}}$ and then predict the class with **highest entry** of $\widetilde{\mathbf{x}}_\star^{\mathsf{T}}\mathbf{W}$.

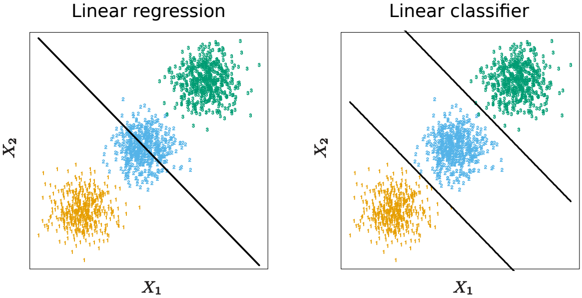
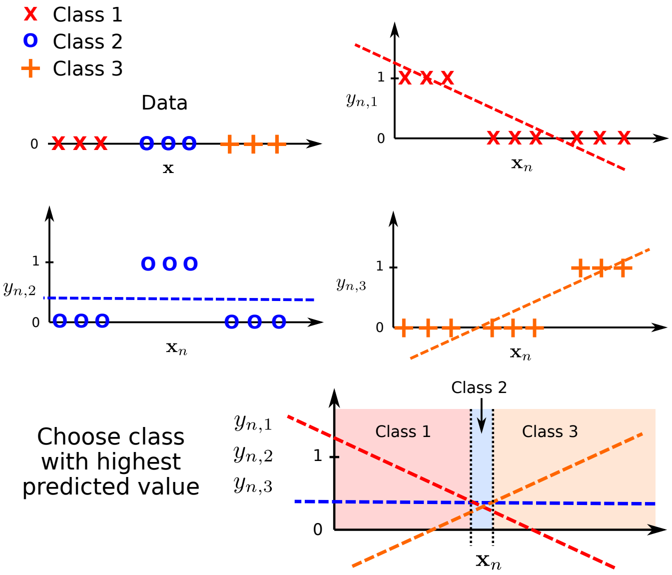


Figure: G. James, D. Witten, T. Hastie and R. Tibshirani. *An Introduction to statistical learning*, 2013.

1D example



Class 2 is underrepresented in the resulting predictions!

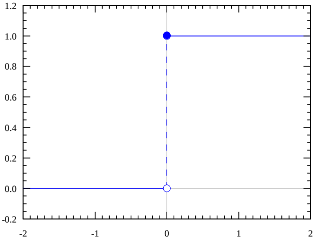
Deterministic linear classification

Works by mapping the output of the linear model into **discrete class labels**.

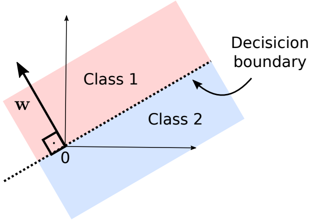
Assume $y_n \in \{0, 1\}$ (binary classification). Then, we can define $y_n = H(\mathbf{w}^\mathsf{T}\widetilde{\mathbf{x}})$,

where $H(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$ is the Heaviside step function.

Heaviside function



What is the **geometric interpretation** of **w**?



w is orthogonal to the decision boundary!

Problem: deterministic predictions.

- Misclassification errors are not allowed.
- Inference is hard: what is the MLE?

Probabilistic linear classification

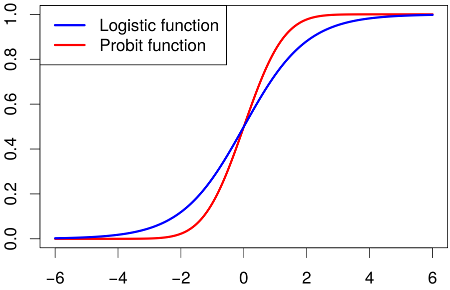
Works by mapping the output of the linear model into **class probabilities**.

$$p(y_n = 1|\widetilde{\mathbf{x}}, \mathbf{w}) = \sigma(\mathbf{w}^\mathsf{T}\widetilde{\mathbf{x}})\,,$$

where $\sigma(\cdot)$ is a monotonically increasing function that maps $\mathbb{R}$ into $[0, 1]$.

For example:

- The logistic function: $\sigma(x) = 1/(1 + \exp(-x))$.
- The probit function or Gaussian CDF: $\sigma(x) = \int_{-\infty}^{x} \mathcal{N}(z|0, 1)\, dz$.



Logistic regression (classification)

Assume $\sigma(x)$ is the **logistic function** and that $y_n \in \{-1, 1\}$. Then

$$p(y_n|\mathbf{x}_n, \mathbf{w}) = \frac{1+y_n}{2}\sigma(\mathbf{w}^\mathsf{T}\widetilde{\mathbf{x}}_n) + \frac{1-y_n}{2}(1-\sigma(\mathbf{w}^\mathsf{T}\widetilde{\mathbf{x}}_n)) = \sigma(y_n\mathbf{w}^\mathsf{T}\widetilde{\mathbf{x}}_n)\,,$$

since $1-\sigma(x) = \sigma(-x)$. For $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$, the **log-likelihood** is

$$\mathcal{L}(\mathbf{w}) = \log p(y_1, \ldots, y_n|\mathbf{x}_1, \ldots, \mathbf{x}_N, \mathbf{w})$$

$$\mathcal{L}(\mathbf{w}) = \sum_{n=1}^N \log p(y_n|\mathbf{x}_n, \mathbf{w}) = \sum_{n=1}^N \log \sigma(y_n\mathbf{w}^\mathsf{T}\widetilde{\mathbf{x}}_n)\,.$$

We can then use $d\sigma(x)/dx = \sigma(x)(1-\sigma(x))$ to obtain the gradient:

$$\frac{d\mathcal{L}(\mathbf{w})}{d\mathbf{w}} = \sum_{n=1}^N y_n \underbrace{(1-\sigma(y_n\mathbf{w}^\mathsf{T}\widetilde{\mathbf{x}}_n))}_{\text{Error Probability}}\widetilde{\mathbf{x}}_n\,.$$

No closed-form solution for MLE, but gradient has **geometric interpretation**.

Gradient ascent

The batch gradient ascent rule to maximize $\mathcal{L}(\mathbf{w})$ is

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} + \alpha \frac{d\mathcal{L}(\mathbf{w})}{d\mathbf{w}} = \mathbf{w}^{\text{old}} + \alpha \sum_{n=1}^{N} y_n (1 - \sigma(y_n \mathbf{w}^{\mathsf{T}} \widetilde{\mathbf{x}}_n)) \widetilde{\mathbf{x}}_n \,.$$

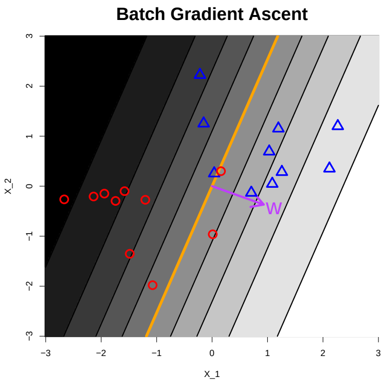
where $\alpha > 0$ is the **learning rate**.

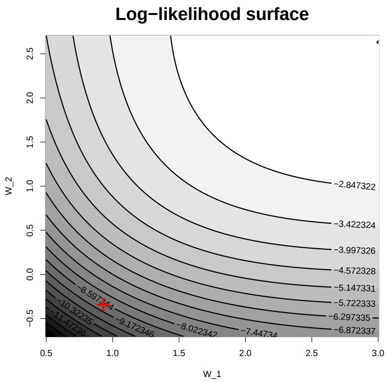
Gradient ascent

The batch gradient ascent rule to maximize $\mathcal{L}(\mathbf{w})$ is

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} + \alpha \frac{d\mathcal{L}(\mathbf{w})}{d\mathbf{w}} = \mathbf{w}^{\text{old}} + \alpha \sum_{n=1}^{N} y_n(1 - \sigma(y_n \mathbf{w}^{\mathrm{T}} \widetilde{\mathbf{x}}_n))\widetilde{\mathbf{x}}_n \,.$$

where $\alpha > 0$ is the **learning rate**.



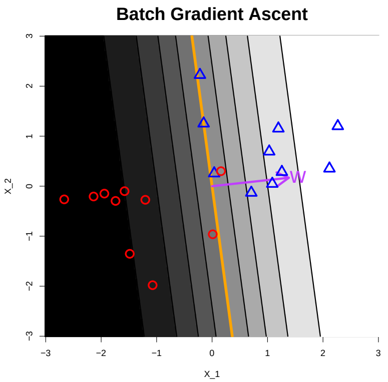


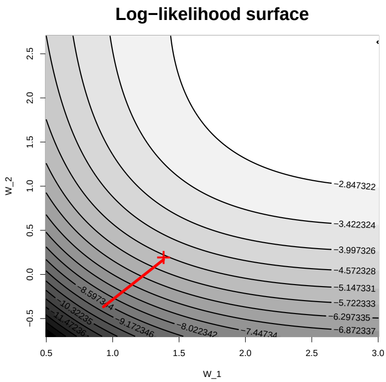
Gradient ascent

The batch gradient ascent rule to maximize $\mathcal{L}(\mathbf{w})$ is

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} + \alpha \frac{d\mathcal{L}(\mathbf{w})}{d\mathbf{w}} = \mathbf{w}^{\text{old}} + \alpha \sum_{n=1}^{N} y_n(1 - \sigma(y_n \mathbf{w}^{\mathrm{T}} \widetilde{\mathbf{x}}_n))\widetilde{\mathbf{x}}_n \,.$$

where $\alpha > 0$ is the **learning rate**.



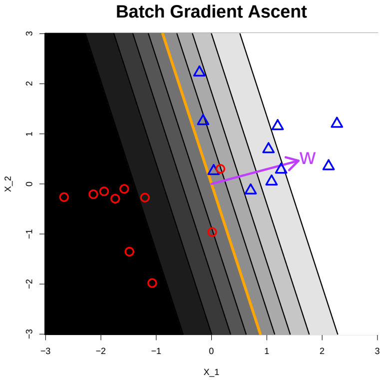


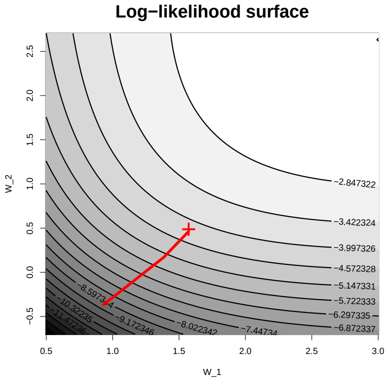
Gradient ascent

The batch gradient ascent rule to maximize $\mathcal{L}(\mathbf{w})$ is

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} + \alpha \frac{d\mathcal{L}(\mathbf{w})}{d\mathbf{w}} = \mathbf{w}^{\text{old}} + \alpha \sum_{n=1}^{N} y_n(1 - \sigma(y_n \mathbf{w}^{\mathrm{T}} \widetilde{\mathbf{x}}_n))\widetilde{\mathbf{x}}_n \,.$$

where $\alpha > 0$ is the **learning rate**.





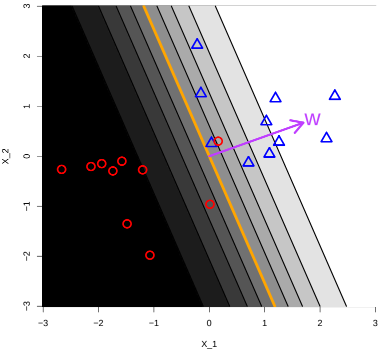
Gradient ascent

The batch gradient ascent rule to maximize $\mathcal{L}(\mathbf{w})$ is

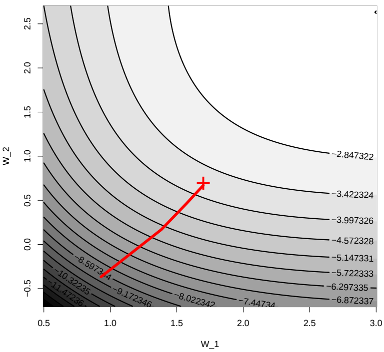
$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} + \alpha \frac{d\mathcal{L}(\mathbf{w})}{d\mathbf{w}} = \mathbf{w}^{\text{old}} + \alpha \sum_{n=1}^{N} y_n (1 - \sigma(y_n \mathbf{w}^{\mathrm{T}} \widetilde{\mathbf{x}}_n)) \widetilde{\mathbf{x}}_n \,.$$

where $\alpha > 0$ is the **learning rate**.

Batch Gradient Ascent



Log−likelihood surface

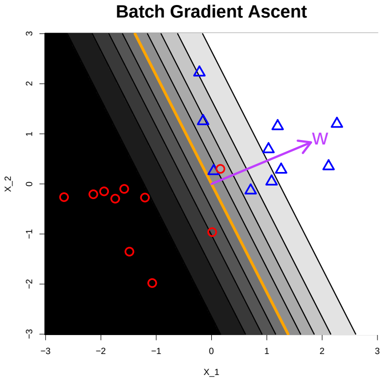


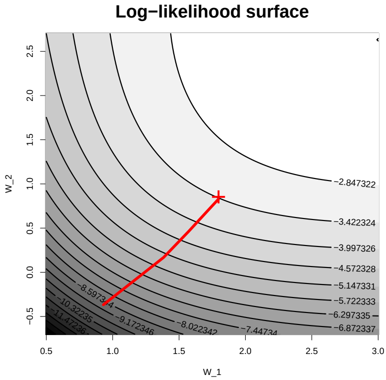
Gradient ascent

The batch gradient ascent rule to maximize $\mathcal{L}(\mathbf{w})$ is

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} + \alpha \frac{d\mathcal{L}(\mathbf{w})}{d\mathbf{w}} = \mathbf{w}^{\text{old}} + \alpha \sum_{n=1}^{N} y_n (1 - \sigma(y_n \mathbf{w}^{\mathrm{T}} \widetilde{\mathbf{x}}_n)) \widetilde{\mathbf{x}}_n \,.$$

where $\alpha > 0$ is the **learning rate**.



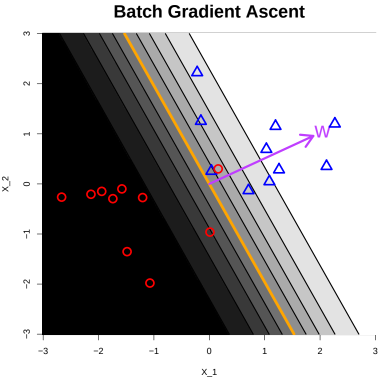


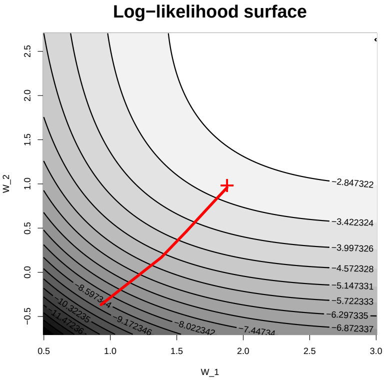
Gradient ascent

The batch gradient ascent rule to maximize $\mathcal{L}(\mathbf{w})$ is

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} + \alpha \frac{d\mathcal{L}(\mathbf{w})}{d\mathbf{w}} = \mathbf{w}^{\text{old}} + \alpha \sum_{n=1}^{N} y_n(1 - \sigma(y_n \mathbf{w}^{\mathsf{T}} \widetilde{\mathbf{x}}_n))\widetilde{\mathbf{x}}_n \, .$$

where $\alpha > 0$ is the the **learning rate**.



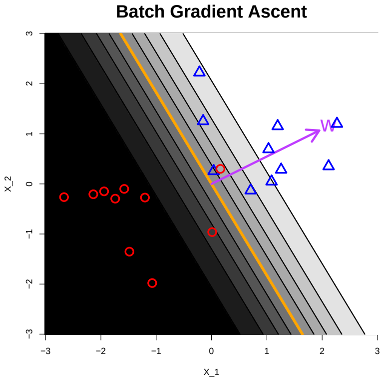


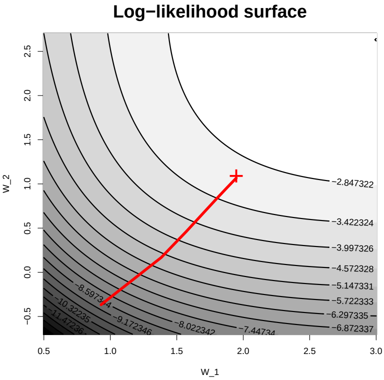
Gradient ascent

The batch gradient ascent rule to maximize $\mathcal{L}(\mathbf{w})$ is

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} + \alpha \frac{d\mathcal{L}(\mathbf{w})}{d\mathbf{w}} = \mathbf{w}^{\text{old}} + \alpha \sum_{n=1}^{N} y_n (1 - \sigma(y_n \mathbf{w}^{\mathsf{T}} \widetilde{\mathbf{x}}_n)) \widetilde{\mathbf{x}}_n \,.$$

where $\alpha > 0$ is the **learning rate**.



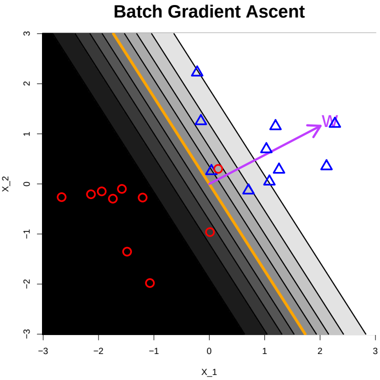


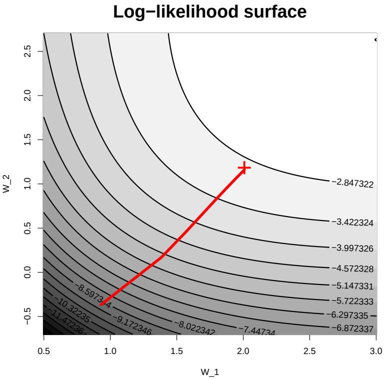
Gradient ascent

The batch gradient ascent rule to maximize $\mathcal{L}(\mathbf{w})$ is

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} + \alpha \frac{d\mathcal{L}(\mathbf{w})}{d\mathbf{w}} = \mathbf{w}^{\text{old}} + \alpha \sum_{n=1}^{N} y_n (1 - \sigma(y_n \mathbf{w}^{\mathsf{T}} \widetilde{\mathbf{x}}_n)) \widetilde{\mathbf{x}}_n \,.$$

where $\alpha > 0$ is the **learning rate**.



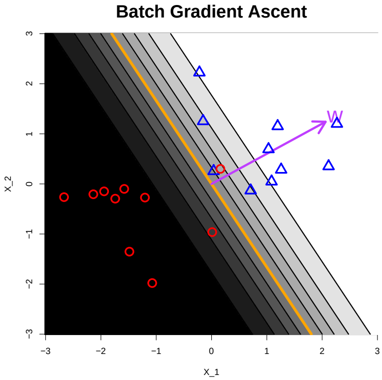


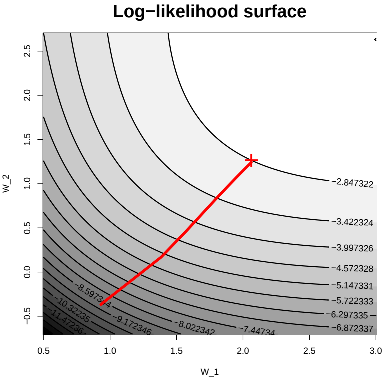
Gradient ascent

The batch gradient ascent rule to maximize $\mathcal{L}(\mathbf{w})$ is

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} + \alpha \frac{d\mathcal{L}(\mathbf{w})}{d\mathbf{w}} = \mathbf{w}^{\text{old}} + \alpha \sum_{n=1}^{N} y_n (1 - \sigma(y_n \mathbf{w}^{\mathrm{T}} \widetilde{\mathbf{x}}_n)) \widetilde{\mathbf{x}}_n \,.$$

where $\alpha > 0$ is the **learning rate**.



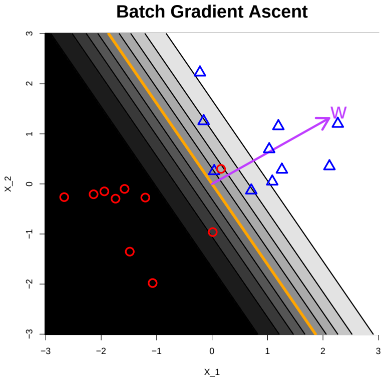


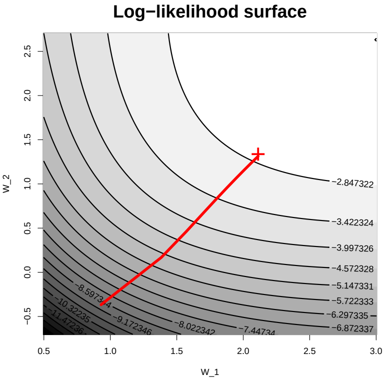
Gradient ascent

The batch gradient ascent rule to maximize $\mathcal{L}(\mathbf{w})$ is

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} + \alpha \frac{d\mathcal{L}(\mathbf{w})}{d\mathbf{w}} = \mathbf{w}^{\text{old}} + \alpha \sum_{n=1}^{N} y_n (1 - \sigma(y_n \mathbf{w}^{\mathsf{T}} \widetilde{\mathbf{x}}_n)) \widetilde{\mathbf{x}}_n \, .$$

where $\alpha > 0$ is the **learning rate**.



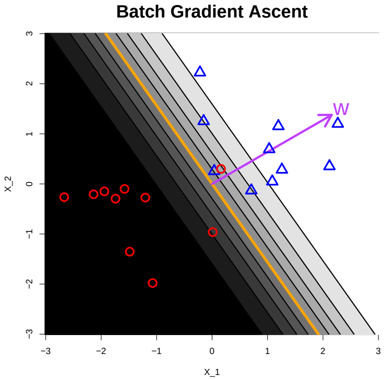


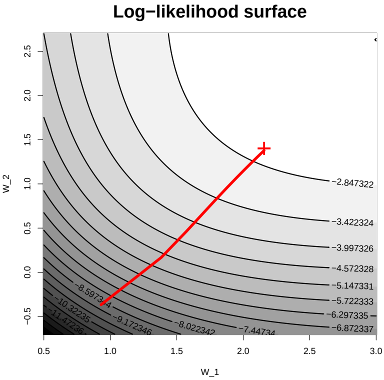
Gradient ascent

The batch gradient ascent rule to maximize $\mathcal{L}(\mathbf{w})$ is

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} + \alpha \frac{d\mathcal{L}(\mathbf{w})}{d\mathbf{w}} = \mathbf{w}^{\text{old}} + \alpha \sum_{n=1}^{N} y_n (1 - \sigma(y_n \mathbf{w}^{\mathsf{T}} \widetilde{\mathbf{x}}_n)) \widetilde{\mathbf{x}}_n \, .$$

where $\alpha > 0$ is the **learning rate**.



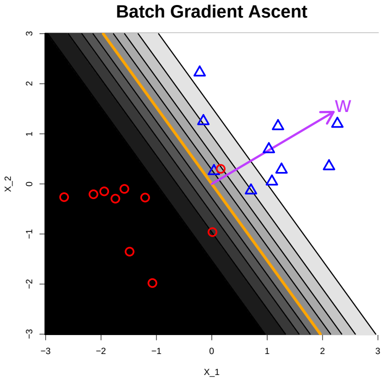


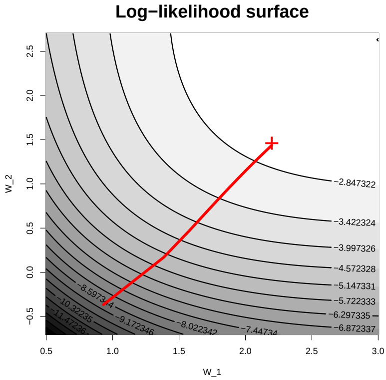
Gradient ascent

The batch gradient ascent rule to maximize $\mathcal{L}(\mathbf{w})$ is

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} + \alpha \frac{d\mathcal{L}(\mathbf{w})}{d\mathbf{w}} = \mathbf{w}^{\text{old}} + \alpha \sum_{n=1}^{N} y_n (1 - \sigma(y_n \mathbf{w}^{\mathsf{T}} \widetilde{\mathbf{x}}_n)) \widetilde{\mathbf{x}}_n \,.$$

where $\alpha > 0$ is the **learning rate**.



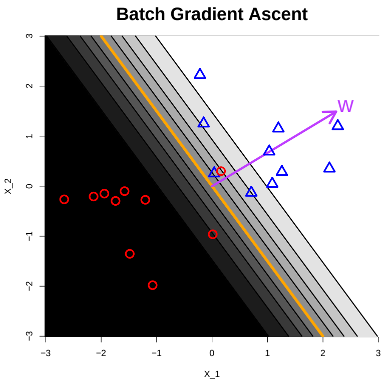


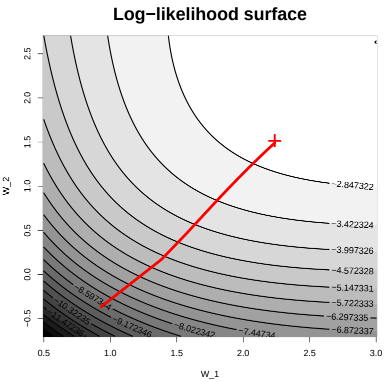
Gradient ascent

The batch gradient ascent rule to maximize $\mathcal{L}(\mathbf{w})$ is

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} + \alpha \frac{d\mathcal{L}(\mathbf{w})}{d\mathbf{w}} = \mathbf{w}^{\text{old}} + \alpha \sum_{n=1}^{N} y_n (1 - \sigma(y_n \mathbf{w}^{\mathrm{T}} \widetilde{\mathbf{x}}_n)) \widetilde{\mathbf{x}}_n \, .$$

where $\alpha > 0$ is the **learning rate**.



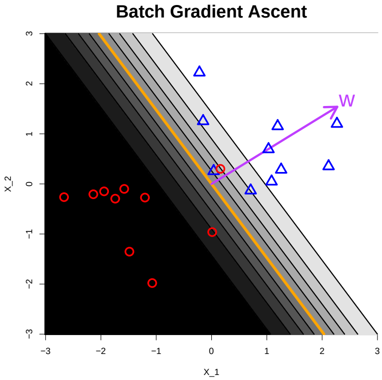


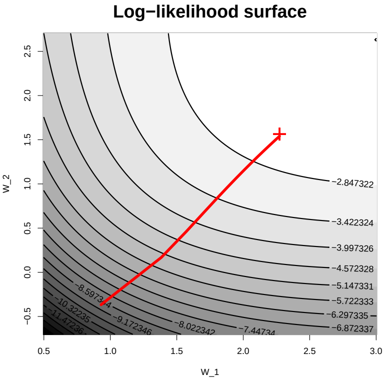
Gradient ascent

The batch gradient ascent rule to maximize $\mathcal{L}(\mathbf{w})$ is

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} + \alpha \frac{d\mathcal{L}(\mathbf{w})}{d\mathbf{w}} = \mathbf{w}^{\text{old}} + \alpha \sum_{n=1}^{N} y_n(1 - \sigma(y_n \mathbf{w}^{\mathrm{T}} \widetilde{\mathbf{x}}_n))\widetilde{\mathbf{x}}_n \,.$$

where $\alpha > 0$ is the **learning rate**.



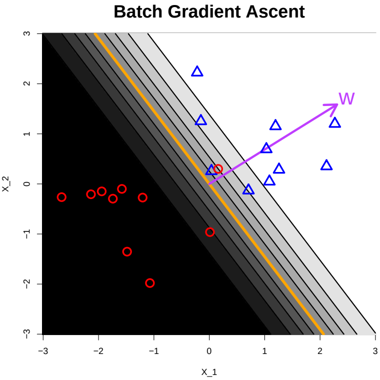


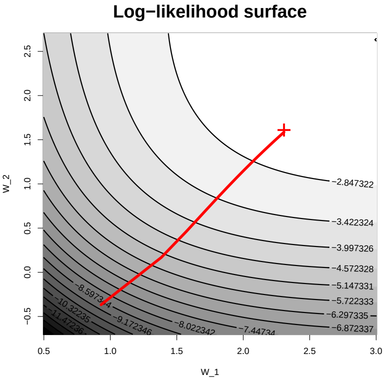
Gradient ascent

The batch gradient ascent rule to maximize $\mathcal{L}(\mathbf{w})$ is

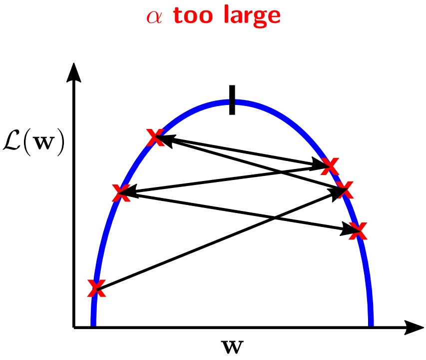
$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} + \alpha \frac{d\mathcal{L}(\mathbf{w})}{d\mathbf{w}} = \mathbf{w}^{\text{old}} + \alpha \sum_{n=1}^{N} y_n (1 - \sigma(y_n \mathbf{w}^{\mathsf{T}} \widetilde{\mathbf{x}}_n)) \widetilde{\mathbf{x}}_n \, .$$

where $\alpha > 0$ is the **learning rate**.

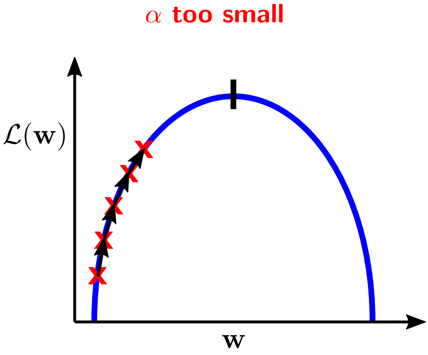




Choosing the learning rate



The optimization bounces around the maximum and it could diverge!



Convergence to the maximum is very slow!

No rule exists to choose α optimally. Only trial and error!

Linear classification with more than 2 classes

We can map multiple outputs to discrete class labels using the **max function**:

$$y_n = \underset{k \in \{1,\ldots,C\}}{\arg\max}\, \mathbf{w}_k^{\mathsf{T}} \widetilde{\mathbf{x}}\,,$$

but this has similar problems as in the deterministic binary classification case.

Instead, use the **soft-max function** to map the outputs into class probabilities:

$$p(y_n = k | \mathbf{w}_1, \ldots, \mathbf{w}_K, \widetilde{\mathbf{x}}_n) = \frac{\exp(\mathbf{w}_k^{\mathsf{T}} \widetilde{\mathbf{x}}_n)}{\sum_{k'=1}^{K} \exp(\mathbf{w}_{k'}^{\mathsf{T}} \widetilde{\mathbf{x}}_n)}\,.$$

Equivalent to logistic regression when $C = 2$.

Non-linear logistic regression

Replace $\mathbf{x}$ with non-linear functions of the inputs $\boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \ldots, \phi_M(\mathbf{x}))^\mathsf{T}$.

Inference does not change, just replace each $\mathbf{x}_n$ with the new $\boldsymbol{\phi}(\mathbf{x}_n)$.

For example, $(x_1, x_2) \to (x_1, x_2, x_1 x_2, x_1^2, x_2^2)$.