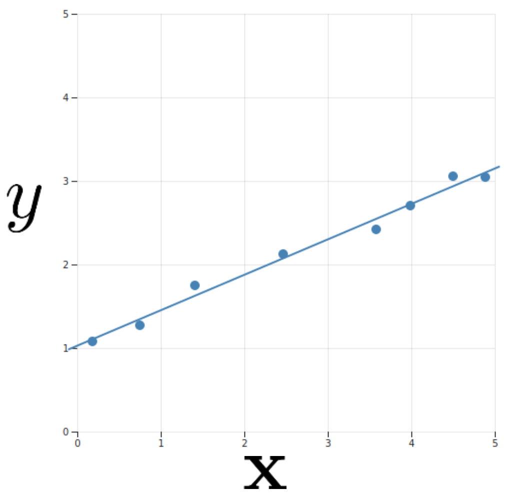
Photogrammetry & Robotics Lab

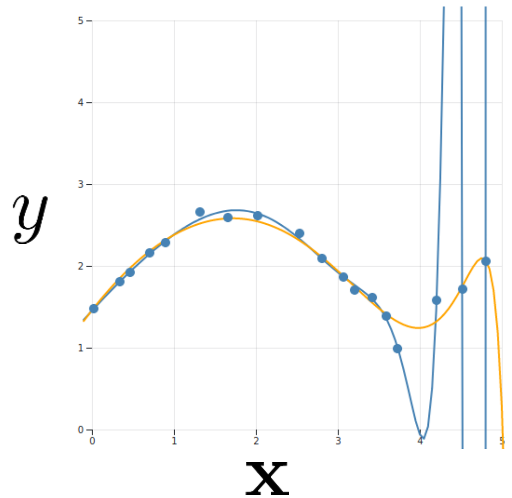
Machine Learning for Robotics and Computer Vision

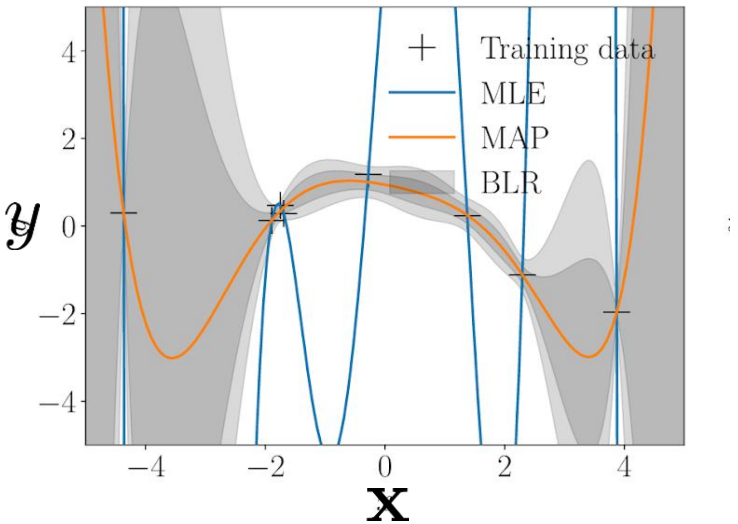
Classification

Jens Behley

Recap: Last Lecture

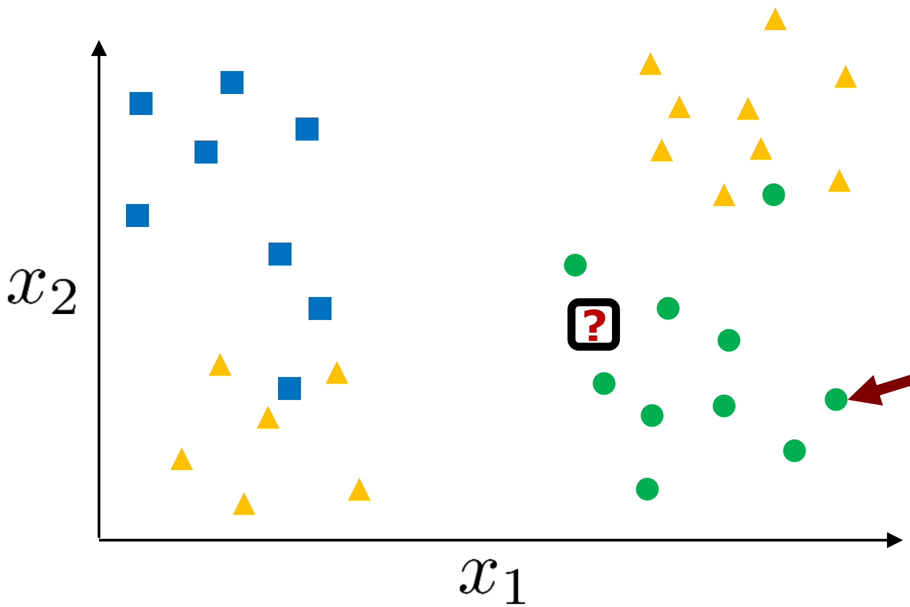






- Linear Regression and variants
- Derivation of ML, MAP, and Bayesian Estimate
- Discriminative vs. Generative Modeling

Classification



$\mathbf{x}$	y	
(0.2, 0.9)	0	■
(0.3, 0.2)	1	▲
(0.1, 0.3)	1	▲
(0.9, 0.2)	2	●
(0.9, 0.2)	2	●
⋮	⋮	
(0.7, 0.4)	?	□

- **Classification:** Given an input $\mathbf{x} \in \mathbb{R}^D$, we want to determine class/label $y \in \{0, \ldots, K-1\}$
- Find $P(y|\mathbf{x})$ that determine class for unseen data points $\mathbf{x} \in \mathbb{R}^D$

[CVMLI Chap. 9]

Generative Model for Classification

$$P(y|x) = \frac{P(\mathbf{x}|y)P(y)}{P(x)}$$

- Model $P(y|\mathbf{x})$ directly: **Discriminative Model**
- Linear Regression is a discriminative model
- For a **generative model**, model instead:
  - $P(\mathbf{x}|y)$ and $P(y)$
  - Or $P(\mathbf{x}, y)$ $\quad (P(\mathbf{x}, y) = P(x|y)P(y))$

Naïve Bayes

- Naïve Bayes assumes independence of features

$$P(\mathbf{x}|y) = \prod_{d=1}^{D} P(x_d|y)$$

- Model assumptions:
  - Normal Distribution for features

$$P(x_d|y=k) = \mathcal{N}(x_d|\mu_{k,d}, \sigma^2_{k,d})$$

  - Categorical Distribution for labels

$$P(y=k) = \lambda_k \qquad \text{with } \sum_{k=1}^{K} \lambda_k = 1$$

Learning Naïve Bayes

- Derive only **maximum likelihood** estimate, but also MAP and Bayesian estimation possible.

- **Receipt** for deriving the learning rule:
  1. Determine negative likelihood term
  2. Find gradient for parameters
  3. Determine parameters, where gradient is zero

NLL for Naïve Bayes

- Maximum Likelihood parameters:

$$
\begin{aligned}
\theta* &= \arg\max_{\theta} P(\theta | y_{1:N}, \mathbf{x}_{1:N}) \\
&= \arg\max_{\theta} P(y_{1:N} | \mathbf{x}_{1:N}, \theta) \\
&= \arg\max_{\theta} \prod_{i=1}^{N} P(y_i | \mathbf{x}_i, \theta)
\end{aligned}
$$

- With Naïve Bayes assumption, we get:

$$
\theta* = \arg\max_{\theta} \prod_{i=1}^{N} \prod_{d=1}^{D} P(x_{i,d} | y_i, \theta) P(y_i | \theta)
$$

NLL for Naïve Bayes (cont.)

- Negative log-transform:

$$\theta * = \arg\min_{\theta} -\log \prod_{i=1}^{N} \prod_{d=1}^{D} P(x_{i,d}|y_i, \theta) P(y_i|\theta)$$

$$= \arg\min_{\theta} \underbrace{- \sum_{i=1}^{N} \sum_{d=1}^{D} \log P(x_{i,d}|y_i, \theta) + \log P(y_i|\theta)}_{\mathcal{L}(\theta)},$$

Derivation for P(x|y)

- Derive now parameters μ_d, σ_d^2 for specific feature d and $y = k$

$$-\sum_{i=1}^{N} \log P(x_{i,d}|y_i, \theta)$$

$$= -\sum_{i=1}^{N} \log \mathcal{N}(x_{i,d}|\mu_{y_i,d}, \sigma_{y_i,d}^2)$$

- As we are interest in examples with $y = k$

$$= -\sum_{i=1}^{N_k} \log \left((2\pi\sigma_{k,d}^2)^{-\frac{1}{2}} \exp \left(-\frac{1}{2} \frac{(x_{i,d} - \mu_{k,d})^2}{\sigma_{k,d}^2} \right) \right)$$

$$= \sum_{i=1}^{N_k} \frac{1}{2} \log(2\pi) + \frac{1}{2} \log(\sigma_{k,d}^2) + \frac{1}{2} \frac{(x_d - \mu_{k,d})^2}{\sigma_{k,d}^2}$$

Derivation for P(y|x) (cont.)

- Getting gradient and setting this to zero:

$$\frac{\partial \mathcal{L}(\theta)}{\partial \mu_d} = \frac{1}{\partial \mu_d} \sum_{i=1}^{N_k} \frac{1}{2}\log(2\pi) + \frac{1}{2}\log(\sigma_d^2) + \frac{1}{2}\frac{(x_d-\mu_d)^2}{\sigma_d^2}$$

$$= \sum_{i=1}^{N_k} \frac{1}{\sigma_d^2}(x_d-\mu_d)$$

$$= \frac{1}{\sigma_d^2}\left(\sum_{i=1}^{N_k} x_d - \sum_{i=1}^{N_k} \mu_d\right)$$

$$= \frac{1}{\sigma_d^2}\left(\sum_{i=1}^{N_k} x_d - N_k\mu_d\right)$$ **= 0 →** $$\boxed{\mu_d = \frac{1}{N_k}\sum_{i=1}^{N_k} x_d}$$

Derivation for P(y|x) (cont.)

- For σ_d^2 , we can do the same:

$$\frac{\partial \mathcal{L}(\theta)}{\partial \sigma_d^2} = \frac{N_k}{2\sigma_d^2} - \frac{1}{2(\sigma_d^2)^2} \sum_{i=1}^{N_k} (x_d - \mu_d)^2$$

- Setting this to zero gives:

$$\rightarrow \quad \sigma_d^2 = \frac{1}{N_k} \sum_{i=1}^{N_k} (x_d - \mu_d)^2$$

- Thus, maximum likelihood parameters of feature normal distributions:

$$\mu_d = \frac{1}{N_k} \sum_{i=1}^{N_k} x_d \qquad \sigma_d^2 = \frac{1}{N_k} \sum_{i=1}^{N_k} (x_d - \mu_d)^2$$

Derivation of P(y)

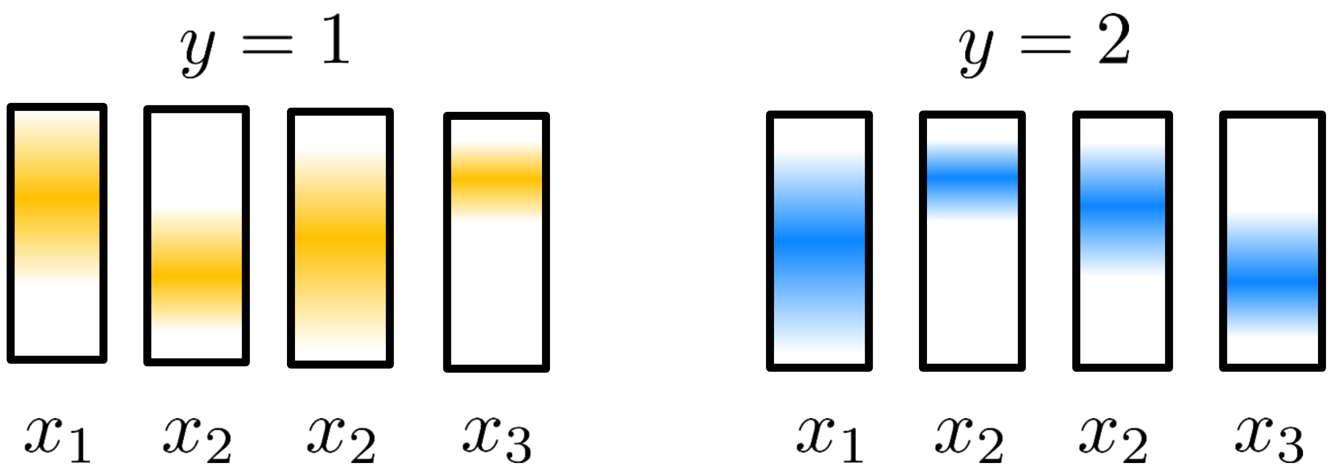
- For the categorical distribution,

$$P(y = k) = \lambda_k \qquad \text{with} \sum_{k=1}^{K} \lambda_k = 1$$

  the maximum likelihood parameters λ_k are given by:

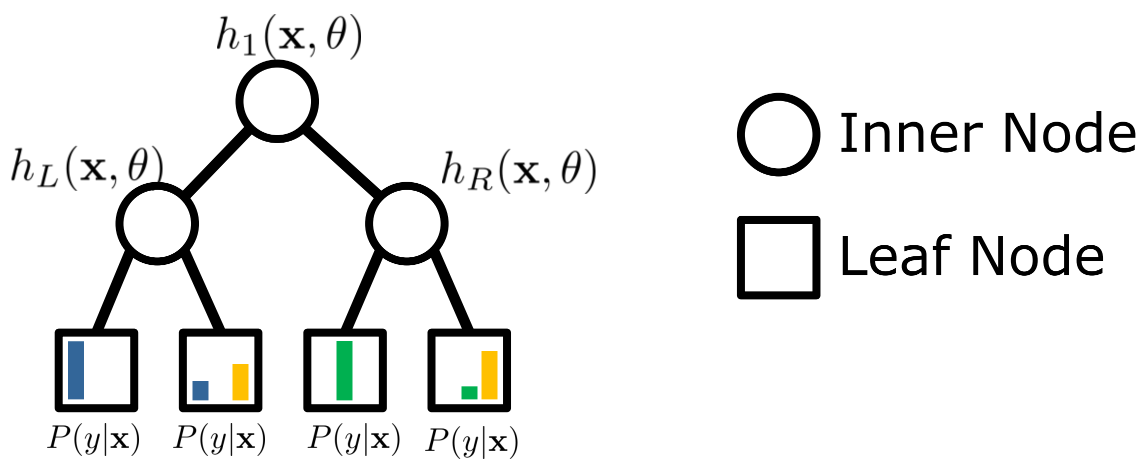
$$\lambda_k = \frac{N_k}{\sum_{j=1}^{K} N_j}$$

Intuition of Feature Likelihoods



- Each feature likelihood defines per feature "range" of feature values
- **Limitations**
  - Only single per-feature range possible
  - Independence assumption

Decision Tree



- **Idea:** Build tree with split functions at inner nodes, where leaves contain $P(y|\mathbf{x})$
- **Split functions** $h(\mathbf{x},\theta) = \{0,1\}$ determine if left or right branch must be traversed.

Split Functions

- Simple threshold split often used:

$$h(\mathbf{x}|d,\tau) = \begin{cases} x_d < \tau, & 0 \\ x_d \geq \tau, & 1 \end{cases}$$

- More complex functions:
  - Linear function of features
  - Non-linear functions

Building Decision Trees

- **CreateNode**($\mathcal{S}$, depth)
  - Reached max depth: `return Leaf(`$P(y|\mathcal{S})$`)`
  - Select good split function $h(\mathbf{x},\theta)$

$$\mathcal{S}_L = \{(\mathbf{x},y) \in \mathcal{S} | h(\mathbf{x},\theta) = 0\}$$
$$\mathcal{S}_R = \{(\mathbf{x},y) \in \mathcal{S} | h(\mathbf{x},\theta) = 1\}$$

  - `node = InnerNode(`$h(\mathbf{x},\theta)$`)`
  - `node.LeftChild =` **CreateNode**($\mathcal{S}_L$, depth + 1)
  - `node.RightChild=` **CreateNode**($\mathcal{S}_R$, depth + 1)
  - **return** `node`

- How to select "good" split function?

Entropy

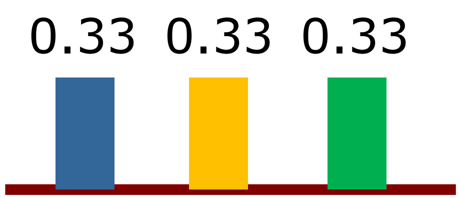
$$\log_2(0) = 0$$

- **Entropy** is defined as

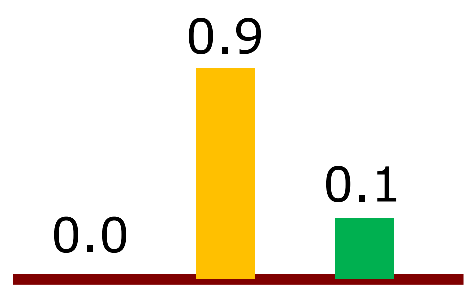
$$H(\mathcal{S}) = -\sum_{k=0}^{K-1} P(y=k)\log_2(P(y=k))$$

with $P(y=k) = |\{(\mathbf{x}, y) \in \mathcal{S} | y = k\}| \cdot |\mathcal{S}|^{-1}$

- Entropy provides measure of information
  - Uniform distribution has highest entropy
  - More peaked distribution has lower entropy



$$H(\mathcal{S}) = 1.585$$



$$H(\mathcal{S}) = 0.469$$

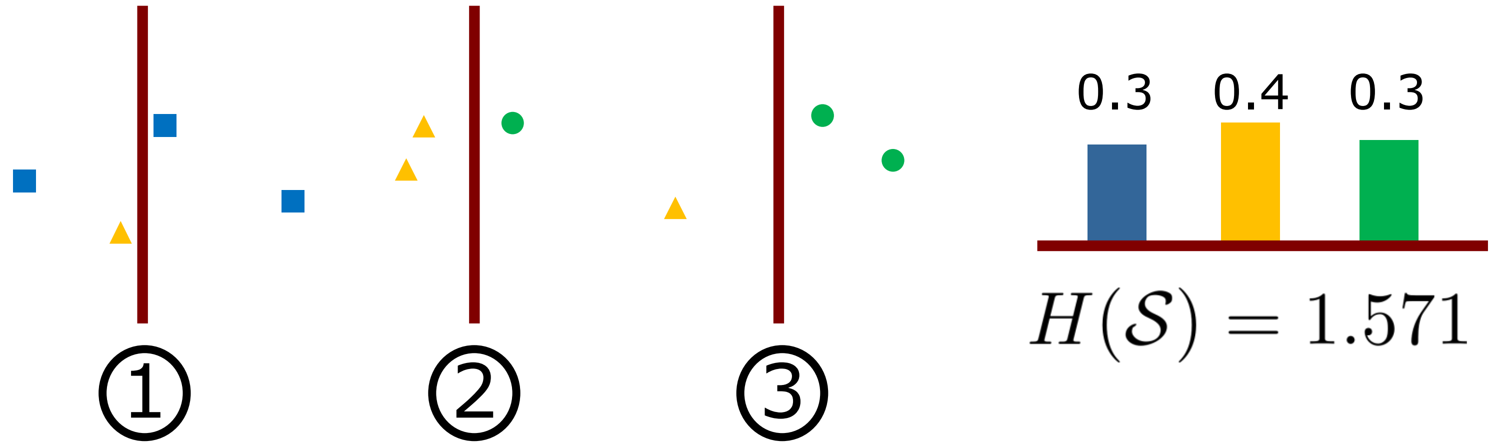
Information Gain

- **Information gain** $I(\mathcal{S}, \mathcal{S}_L, \mathcal{S}_R)$ defined as:

$$I(\mathcal{S}, \mathcal{S}_L, \mathcal{S}_R) = H(\mathcal{S}) - \sum_{i \in \{L,R\}} \frac{|\mathcal{S}_i|}{|\mathcal{S}|} H(\mathcal{S}_i)$$

- Information gain determines change in entropy between the original set $\mathcal{S}$ and the split into subsets $\mathcal{S}_L$ and $\mathcal{S}_R$

Example: Information Gain



$$H(\mathcal{S}) = 1.571$$

$$I(\mathcal{S}, \mathcal{S}_L, \mathcal{S}_R) = H(\mathcal{S}) - \sum_{i \in \{L,R\}} \frac{|\mathcal{S}_i|}{|\mathcal{S}|} H(\mathcal{S}_i)$$

① $I(\mathcal{S}, \mathcal{S}_L, \mathcal{S}_R) = 1.571 - (\frac{2}{10} 1.0 + \frac{8}{10} 1.561) = 0.122$

② $I(\mathcal{S}, \mathcal{S}_L, \mathcal{S}_R) = 1.571 - (\frac{6}{10} 1.0 + \frac{4}{10} 0.811) = 0.646$

③ $I(\mathcal{S}, \mathcal{S}_L, \mathcal{S}_R) = 1.571 - (\frac{8}{10} 1.406 + \frac{2}{10} 0.0) = 0.446$

Selecting Split Functions

- Split functions with parameters d and τ

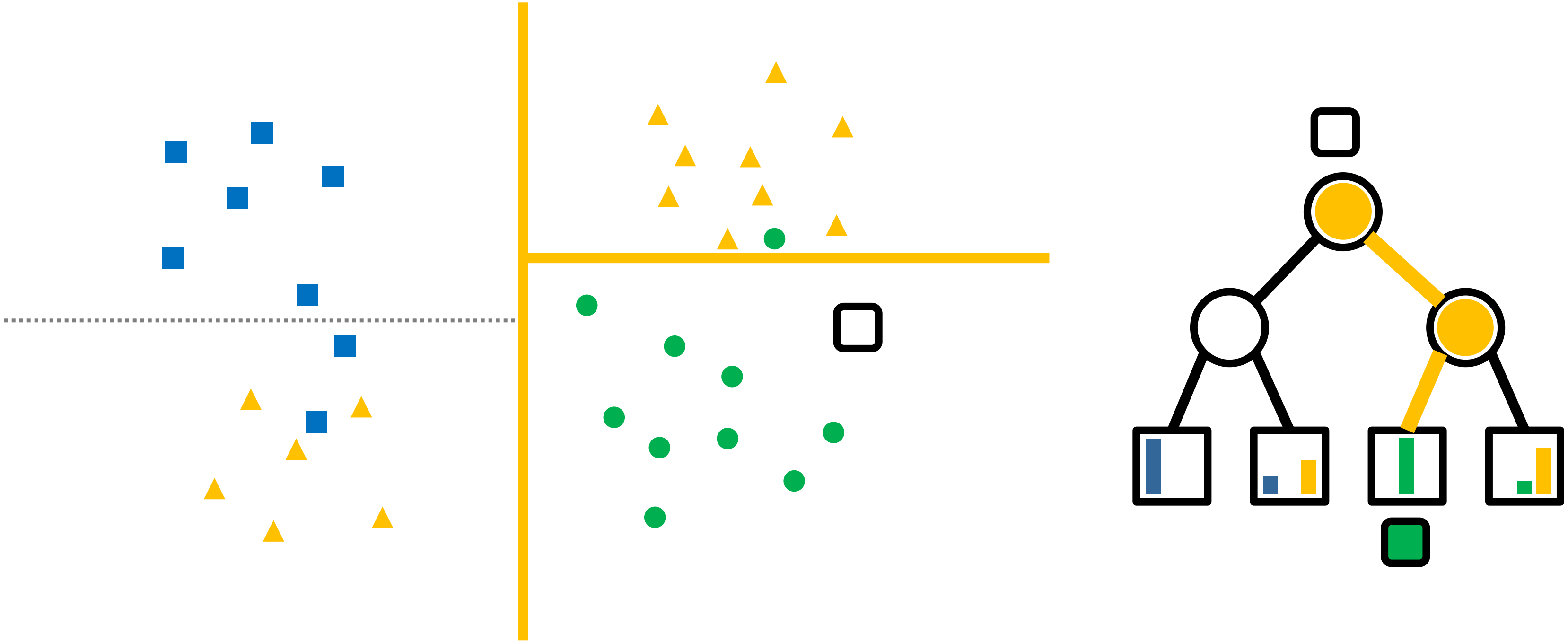
$$h(\mathbf{x}|d,\tau) = \begin{cases} x_d < \tau, & 0 \\ x_d \geq \tau, & 1 \end{cases}$$

- Set of possible split functions can be splits at all features:

$$\mathcal{H}_{\mathcal{S}} = \{h(\mathbf{x}|d,\tau) | d \in \{1,\ldots,D\}, \tau \in \{x_d | (\mathbf{x},y) \in \mathcal{S}\}\}$$

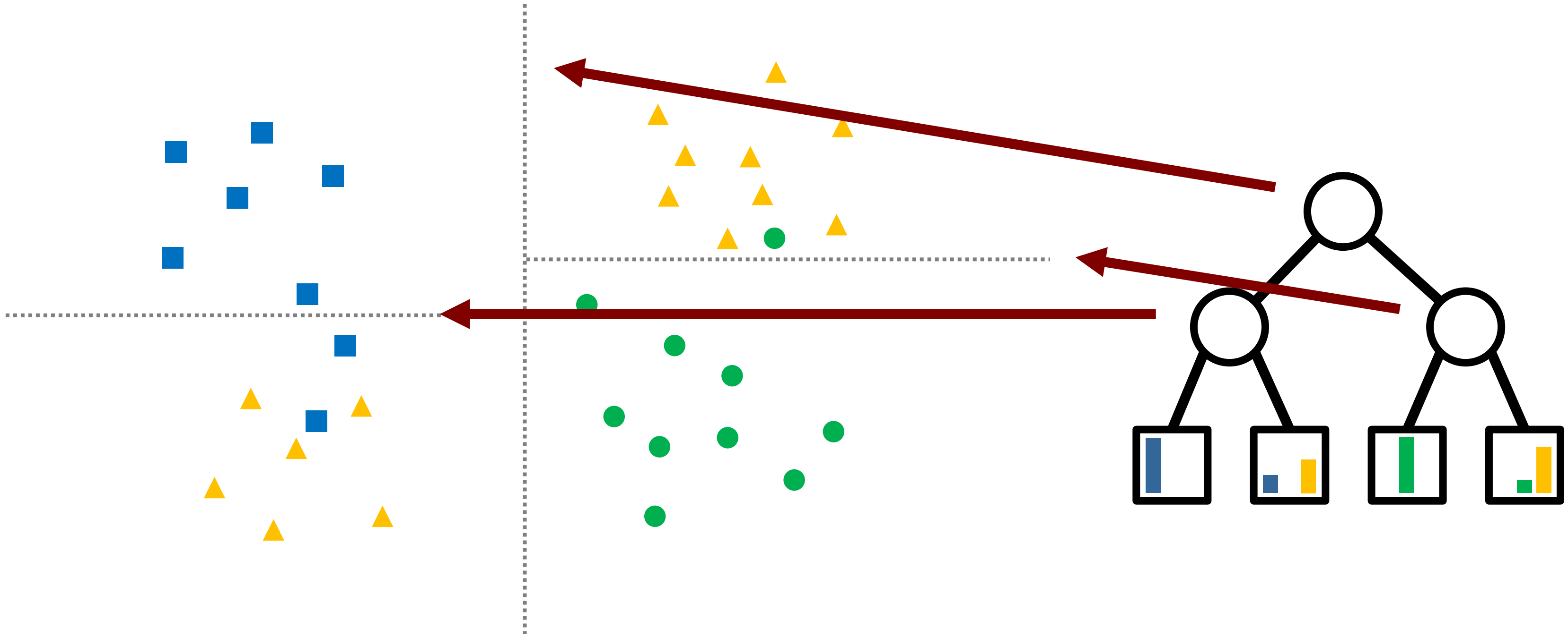
- With larger training sets: fixed number of splits per feature dimension

Example: Inference



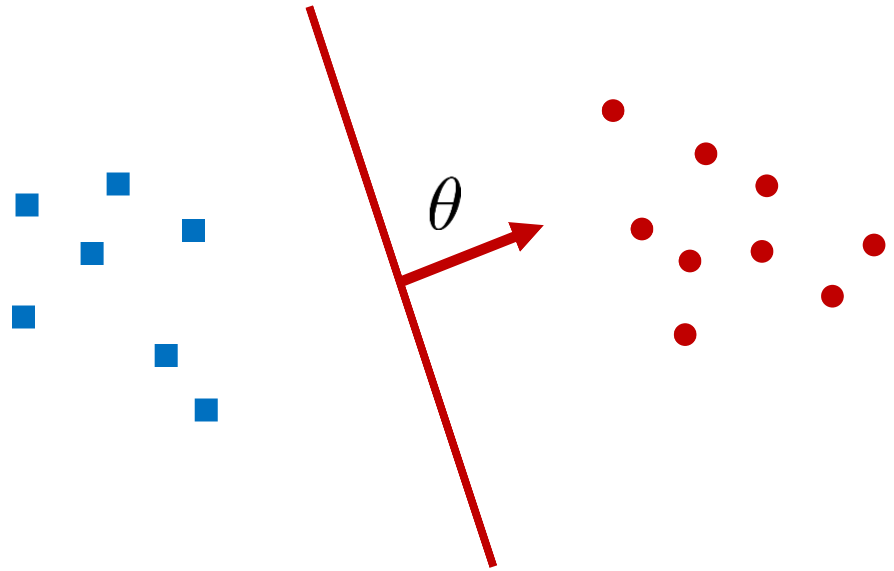
- **Inference:** Start at root and evaluate split functions until leaf is reached

Decision Boundaries



- Each split induces a decision boundary between classes
- Can we directly learn decision boundaries?

Geometric Viewpoint



- Decision boundary is hyperplane between classes:

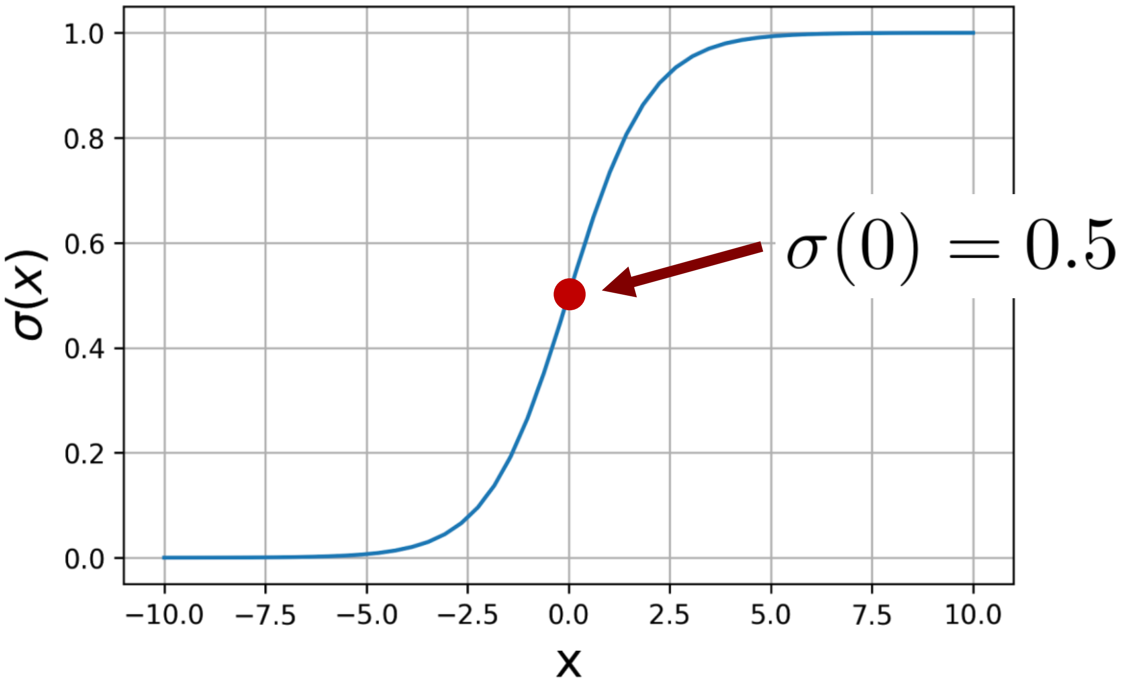
$$d = \theta^T \mathbf{x} + \theta_0$$

- **Idea:** (Scaled) Distance d to plane corresponds to confidence $P(y|\mathbf{x}) \in [0, 1]$

Sigmoid function

- **Sigmoid** function $\sigma : \mathbb{R} \mapsto [0, 1]$ turns x into a value between 0 and 1 defined by:

$$\sigma(x) = \frac{1}{1+\exp(-x)}$$



Logistic Regression

- For **binary** classification $y = \{0, 1\}$, we want:

$$P(y = 0|\mathbf{x}) = 1 - P(y = 1|\mathbf{x})$$

- In **Logistic Regression** we define our model as:

$$P(y = 1|\mathbf{x}) = \sigma(\theta^T \mathbf{x})$$
$$= \frac{1}{1 + \exp(-\theta^T \mathbf{x})}$$
$$P(y = 0|\mathbf{x}) = 1 - P(y = 1|\mathbf{x})$$

- (We used again as in the Linear Regression: $\mathbf{x} := (1, \mathbf{x}^T)^T$)

[CVMLI Chap. 9.1]

Maximum Likelihood Estimation

- NLL for Logistic Regression:

$$-\log \prod_{i=1}^{N} P(y_i|\mathbf{x}_i) = -\sum_{i=1}^{N} \log P(y_i|\mathbf{x}_i)$$

$$= -\sum_{i=1}^{N} \mathbf{1}\{y_i = 0\} \log P(y_i = 0|\mathbf{x}_i) + \mathbf{1}\{y_i = 1\} \log P(y_i = 1|\mathbf{x}_i)$$

Indicator function

$$\boxed{\mathbf{1}(a) = \begin{cases} 1, & \text{a is true} \\ 0, & \text{a is false} \end{cases}}$$

$$\vdots$$

(see lecture notes)

$$\vdots$$

$$\underbrace{= -\sum_{i=1}^{N} (\mathbf{1}\{y_i = 1\} - 1)\theta^T \mathbf{x}_i - \log(1 + \exp(-\theta^T \mathbf{x}))}_{\mathcal{L}(\theta)}$$

[CVMLI Chap. 9.1]

Gradient of NLL

- For the gradient follows:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{1}{\partial \theta}\left(-\sum_{i=1}^{N}(\mathbf{1}\{y_i = 1\} - 1)\theta^T\mathbf{x} - \log\left(1 + \exp(-\theta^T\mathbf{x})\right)\right)$$

$$= -\sum_{i=1}^{N}(\mathbf{1}\{y_i = 1\} - 1)\mathbf{x} - \frac{1}{1+\exp(-\theta^T\mathbf{x})}\exp(-\theta^T\mathbf{x})(-\mathbf{x})$$

$$= -\sum_{i=1}^{N}(\mathbf{1}\{y_i = 1\} - 1)\mathbf{x} + \underbrace{\frac{\exp(-\theta^T\mathbf{x})}{1 + \exp(-\theta^T\mathbf{x})}}_{1-\sigma(\theta^T\mathbf{x})}\mathbf{x}$$

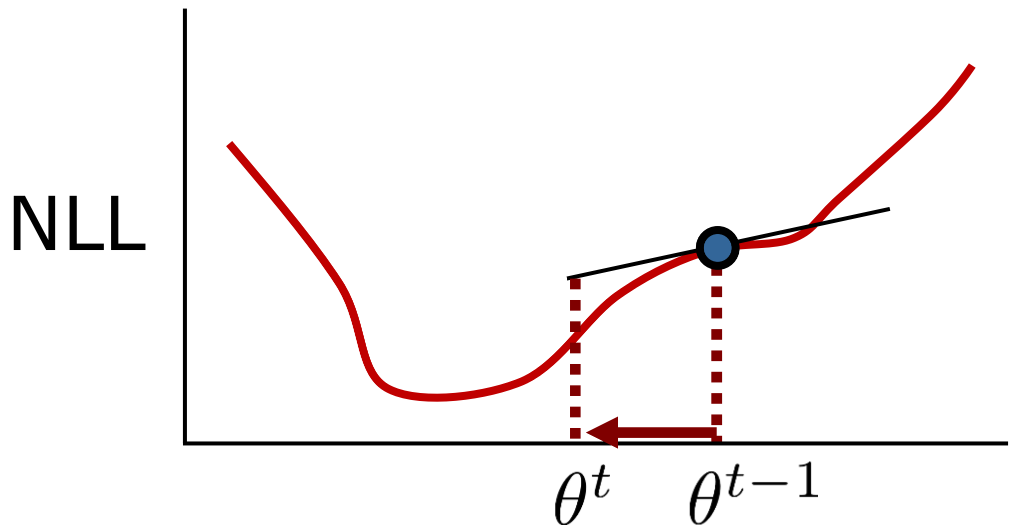
$$= \sum_{i=1}^{N}(\sigma(\theta^T\mathbf{x}) - \mathbf{1}\{y_i = 1\})\mathbf{x}$$

$$= \sum_{i=1}^{N}(P(y_i = 1|\mathbf{x}) - \mathbf{1}\{y_i = 1\})\mathbf{x}$$

- **Problem:** Setting this to zero, no closed form solution!

[CVMLI Chap. 9.1]

Gradient Descent



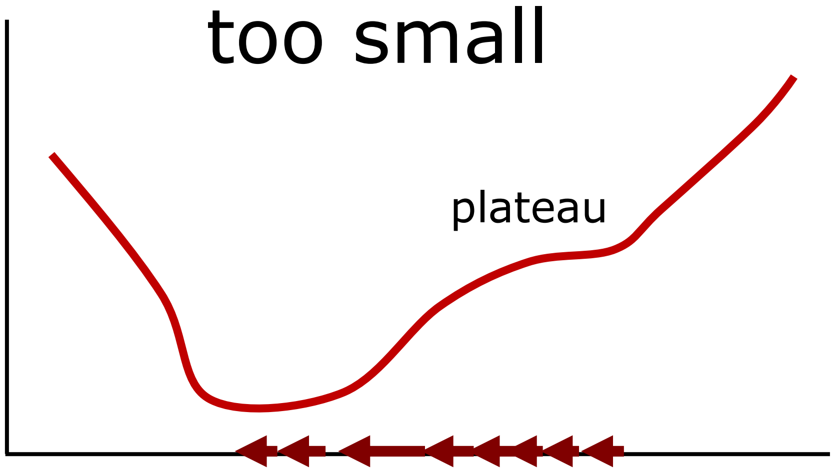
- **Gradient Descent** updates parameters iteratively with negative gradient:

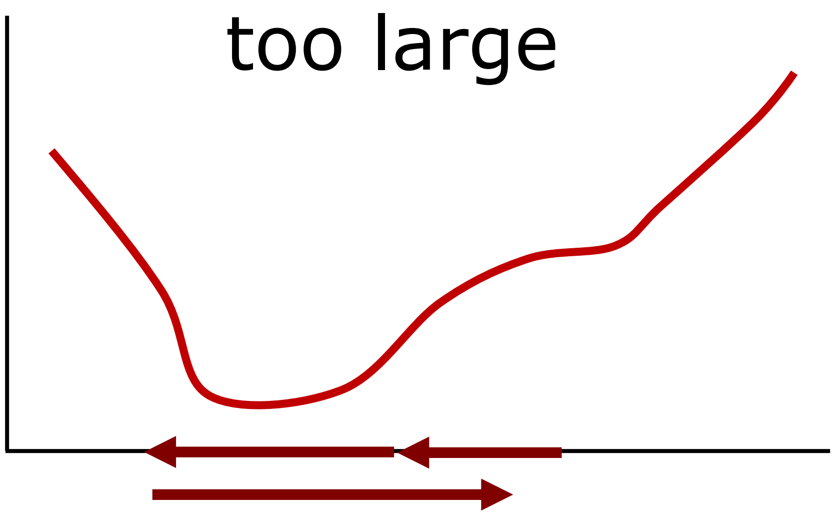
$$\theta^t = \theta^{t-1} - \eta \frac{d\mathcal{L}}{d\theta}$$

- η is called the **learning rate**

Learning Rate

- Learning rate influences progress of minimization
  - Too small: slow progression of optimization
  - Too large: overshooting possible





Possible solution: Hessian

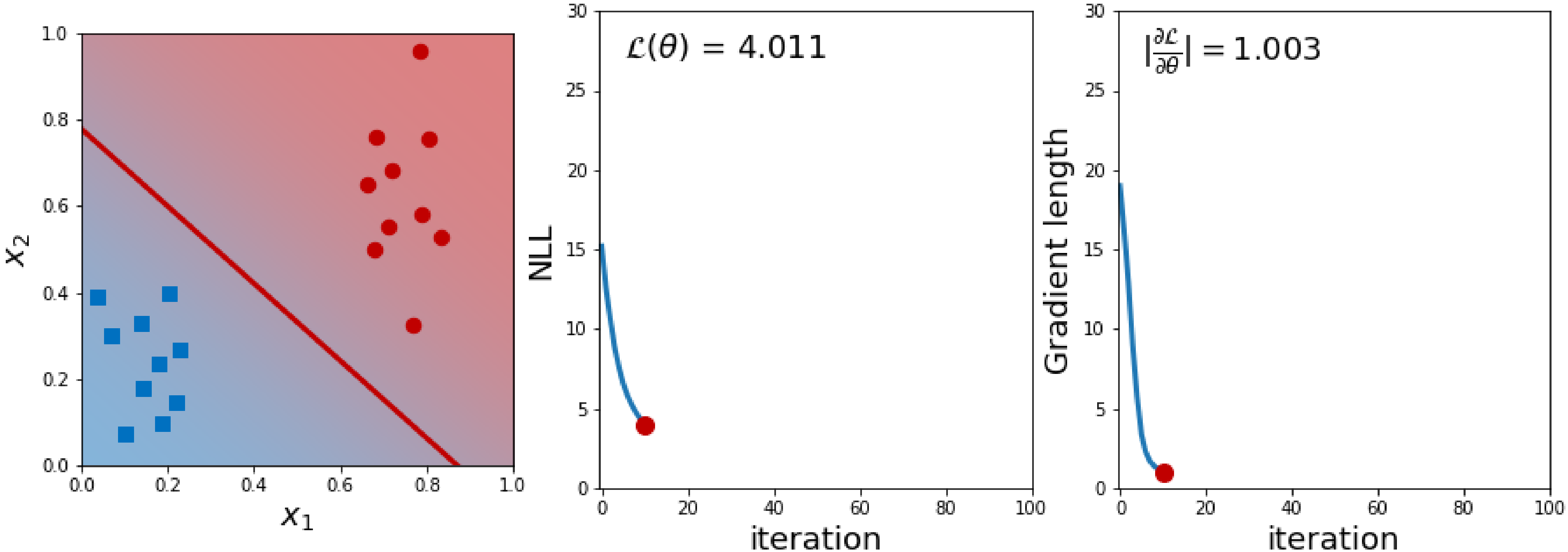
- Finding right learning rate tricky.
- **Solution:** Scaling gradient with second-order derivative (Hessian)
- Newton's method uses Hessian, as follows:

$$\theta^t = \theta^{t-1} - \eta \left(\frac{\partial^2 \mathcal{L}}{\partial^2 \theta} \right)^{-1} \frac{\partial \mathcal{L}}{\partial \theta}$$

- Later, we will see first-order optimizers that do a better job without the Hessian

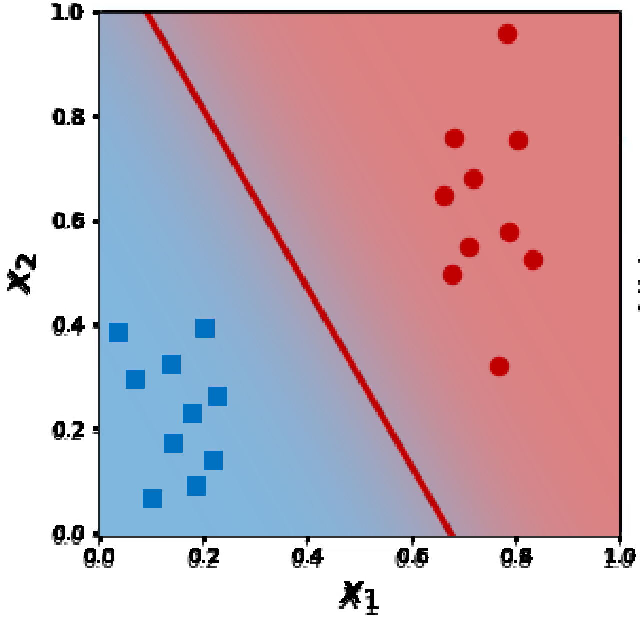
[CVMLI Chap. B.2.2]

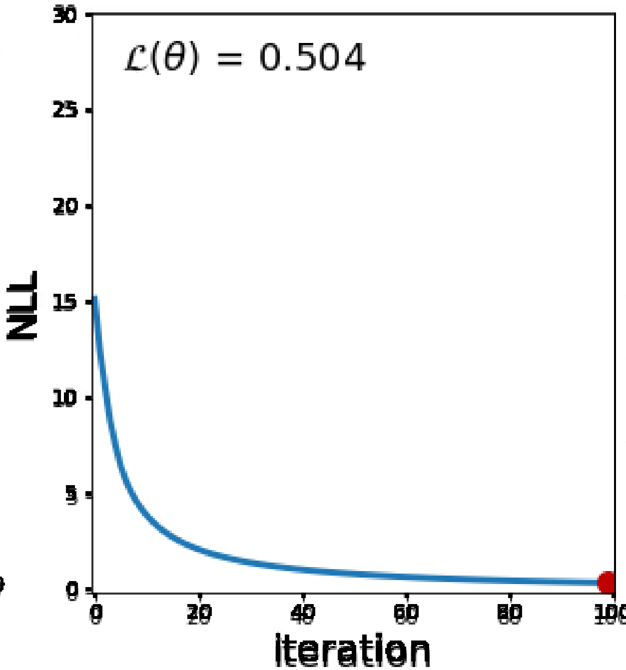
Example: Logistic Regression

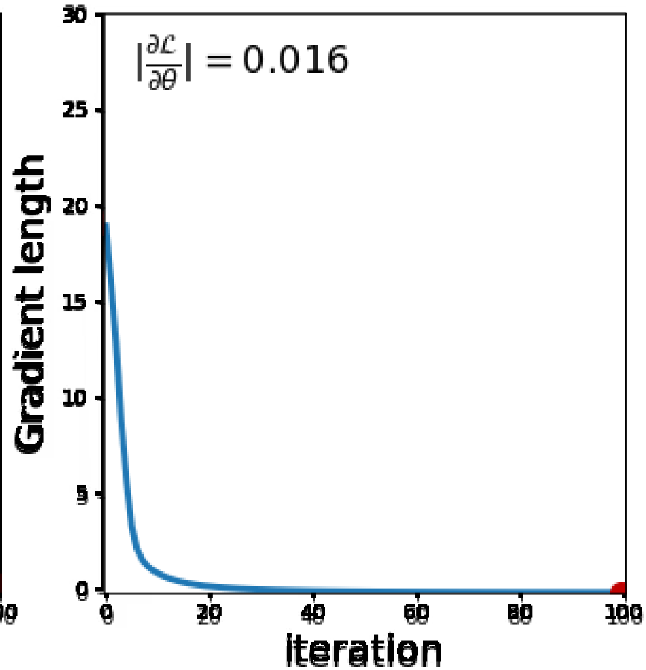


- Here we show the evolution of the decision boundary and the NLL on the middle and length of the gradient on the right

Example: Logistic Regression

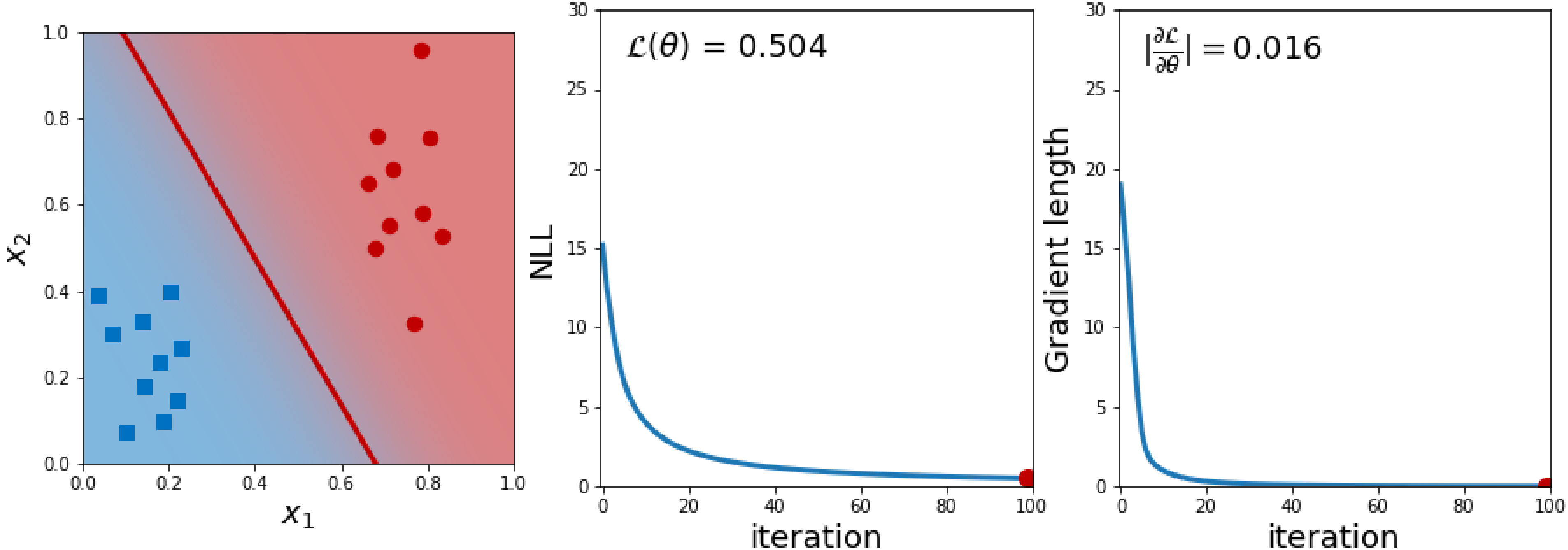






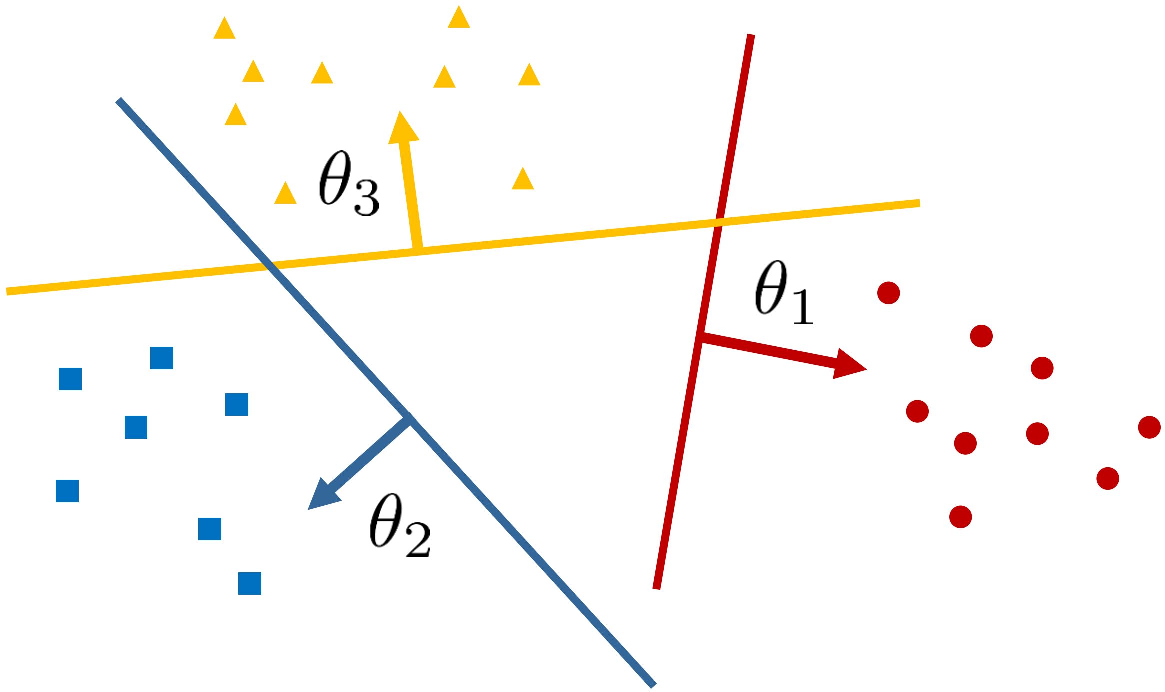
- Here we show the evolution of the decision boundary and the NLL on the middle and length of the gradient on the right

Convergence Criteria



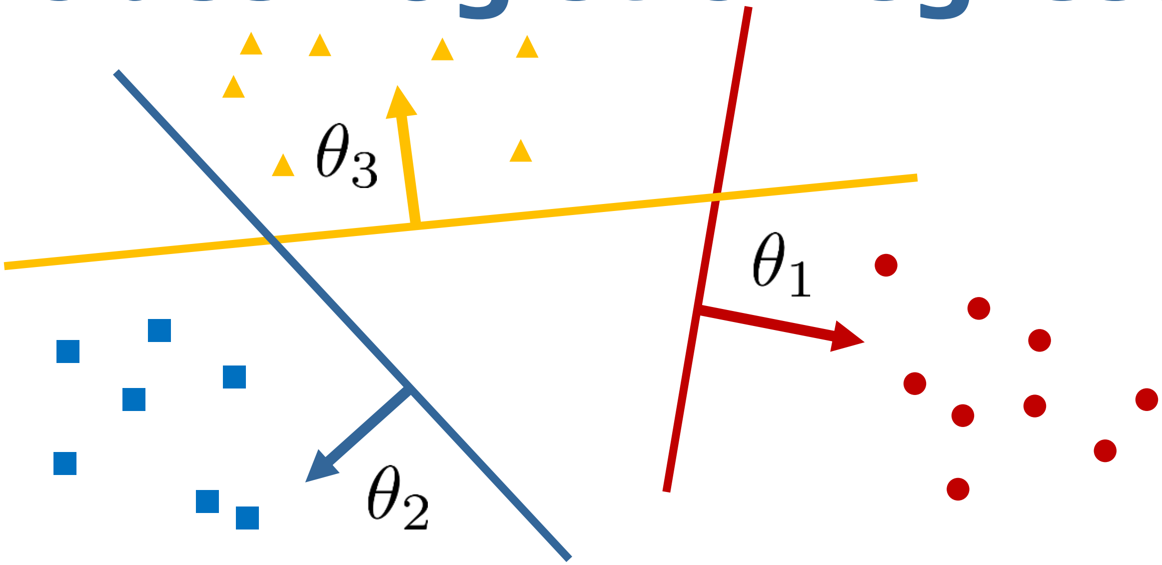
- Decide when to stop gradient update
  - Fixed number of iterations
  - Length of gradient near zero
  - Objective function (e.g., NLL) plateaus

Multi-class Logistic Regression



- We can extend Logistic Regression to handle multiple classes
- Instead of a single term $\mathbf{x}^T\theta$, we introduce a term for every class $\mathbf{x}^T\theta_k$ for every class

Multi-class Logistic Regression



- **Idea:** class with largest $\mathbf{x}^T\theta_k$ should have highest confidence $P(y=k|\mathbf{x})$
- Want to find parameters such that distance is maximize for correct class

- How to turn "distances" into probability distribution?

Softmax

- Softmax function $\mathrm{softmax} : \mathbb{R}^D \mapsto [0,1]^D$ is given by

$$\mathrm{softmax}(\mathbf{x}) = \mathbf{s}$$

with

$$s_i = \frac{\exp(x_i)}{\sum_{d=1}^{D} \exp(x_d)}$$

- Properties of softmax function:
  - $s_i \in [0,1]$
  - $\sum_{i=1}^{D} s_i = 1$

Intuition of Softmax

- Some examples for intuition about the output of softmax function:
  - softmax(10,10,10) = (1/3, 1/3, 1/3)
  - softmax(10,11,10) = (0.21, 0.58, 0.21)
  - softmax(10,13,10) = (0.045, 0.91, 0.045)
  - softmax(9, 11,10) = (0.09, 0.67, 0.24)

Softmax Regression

- Equipped with this, we model $P(y|\mathbf{x})$ as

$$P(y|\mathbf{x}) = \mathrm{softmax}(\theta_0^T \mathbf{x}, \ldots, \theta_{K-1}^T \mathbf{x})$$

$$P(y = k|\mathbf{x}) = \frac{\exp(\theta_k^T \mathbf{x})}{\sum_{j=0}^{K-1} \exp(\theta_j^T \mathbf{x})}$$

- Also here, no closed form solution:
  - Determine gradient of NLL to get MLE
  - Use gradient descent to find best parameters

Gradient for Softmax Regression

- NLL for Softmax Regression:

$$\mathcal{L}(\theta) = -\sum_{i=1}^{N} \log(P(y_i|\mathbf{x}_i))$$

$$= -\sum_{i=1}^{N} \log \frac{\exp(\theta_{y_i}^T \mathbf{x}_i)}{\sum_{j=0}^{K-1} \exp(\theta_j^T \mathbf{x}_i)}$$

$$= \sum_{i=1}^{N} \log \left(\sum_{j=0}^{K-1} \exp(\theta_j^T \mathbf{x}_i) \right) - \theta_{y_i}^T \mathbf{x}_i$$

Gradient: Softmax Regression

- NLL was given by:

$$= \sum_{i=1}^{N} \log\left(\sum_{j=0}^{K-1} \exp(\theta_j^T \mathbf{x}_i)\right) - \theta_{y_i}^T \mathbf{x}_i$$

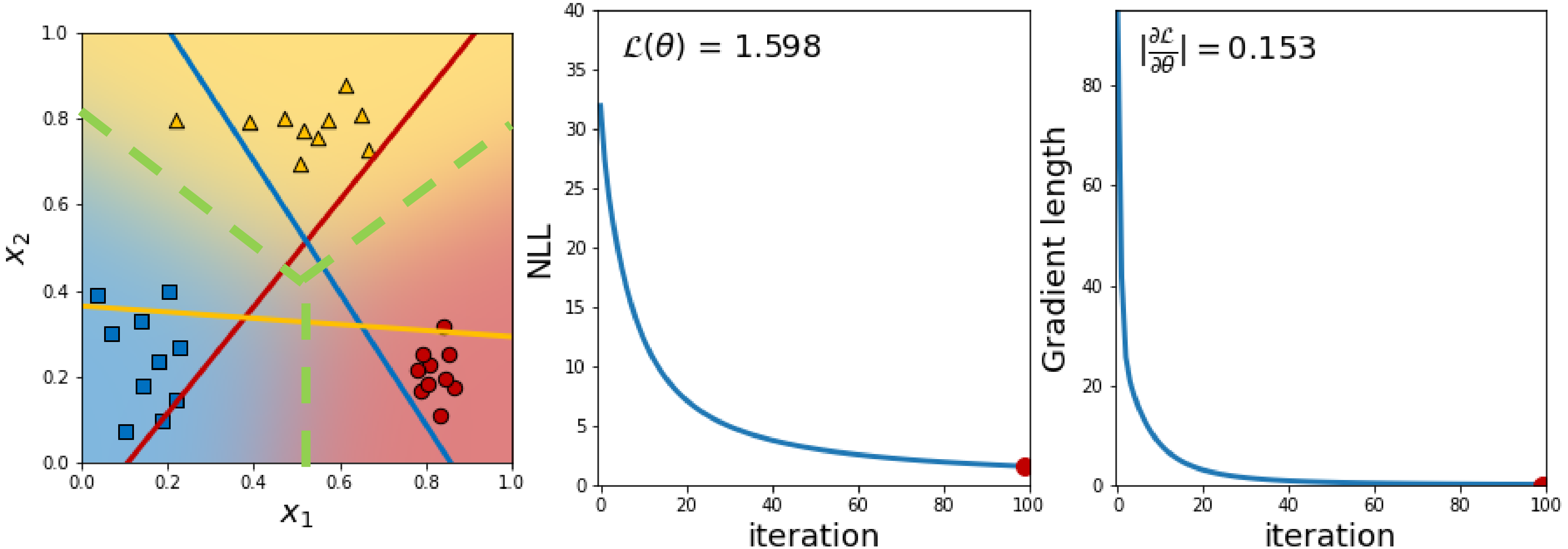
- Gradient for $\boxed{\text{class parameters } \theta_j}$:

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta_j} = \sum_{i=1}^{N} \frac{1}{\sum_{j=0}^{K-1} \exp(\theta_j^T \mathbf{x}_i)} \exp(\theta_j^T \mathbf{x}_i)\mathbf{x}_i - \mathbf{1}\{y_i = j\}\mathbf{x}_i$$

$$= \sum_{i=1}^{N} \left(\frac{\exp(\theta_j^T \mathbf{x})}{\sum_{j=0}^{K-1} \exp(\theta_j^T \mathbf{x}_i)} - \mathbf{1}\{y_i = j\}\right) \mathbf{x}_i$$

$$= \sum_{i=1}^{N} \left(P(y = j|\mathbf{x}) - \mathbf{1}\{y_i = j\}\right) \mathbf{x}_i$$

Example: Softmax Regression



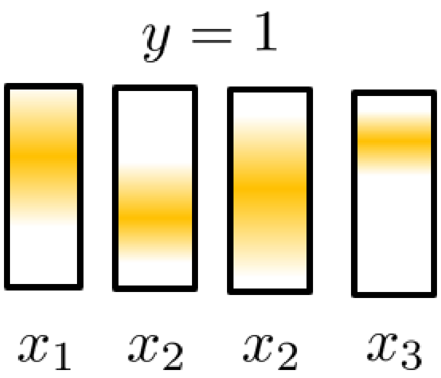
- Again predictions, NLL, and gradient length
- θ_j does not correspond anymore to decision boundary!

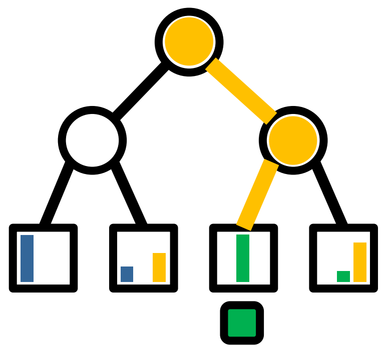
More on Logistic Regression

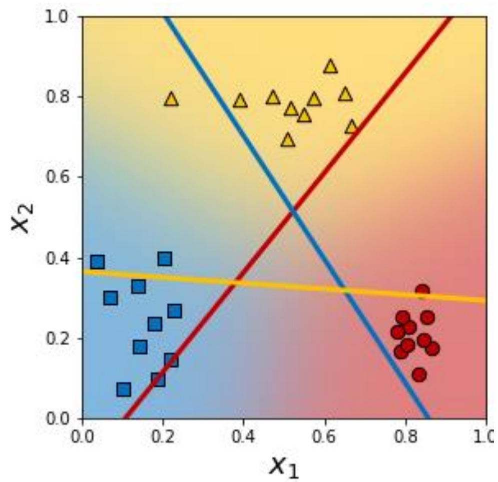
- As with linear regression:
  - **Regularization** via L2 norm on parameters
  - **Non-linear logistic regression** by using non-linear transform
  - **Bayesian logistic regression**, but only approximate solution possible

- See Prince’s book for more information!

Summary







- Classification models
  - Naïve Bayes (Generative Model)
  - Decision Tree (Discriminative Model)
  - Logistic/Softmax Regression (Discriminative Model)

- Optimization with Gradient Descent