10a. Markov Chain Monte Carlo

Markov Chain Monte Carlo

- In high-dimensional spaces, rejection sampling and importance sampling are very inefficient
- An alternative is Markov Chain Monte Carlo (MCMC)
- It keeps a record of the current state and the proposal depends on that state
- Most common algorithms are the Metropolis-Hastings algorithm and Gibbs Sampling

Markov Chains Revisited

A Markov Chain is a distribution over discrete-state random variables $\mathbf{x}_1, \ldots, \mathbf{x}_M$ so that

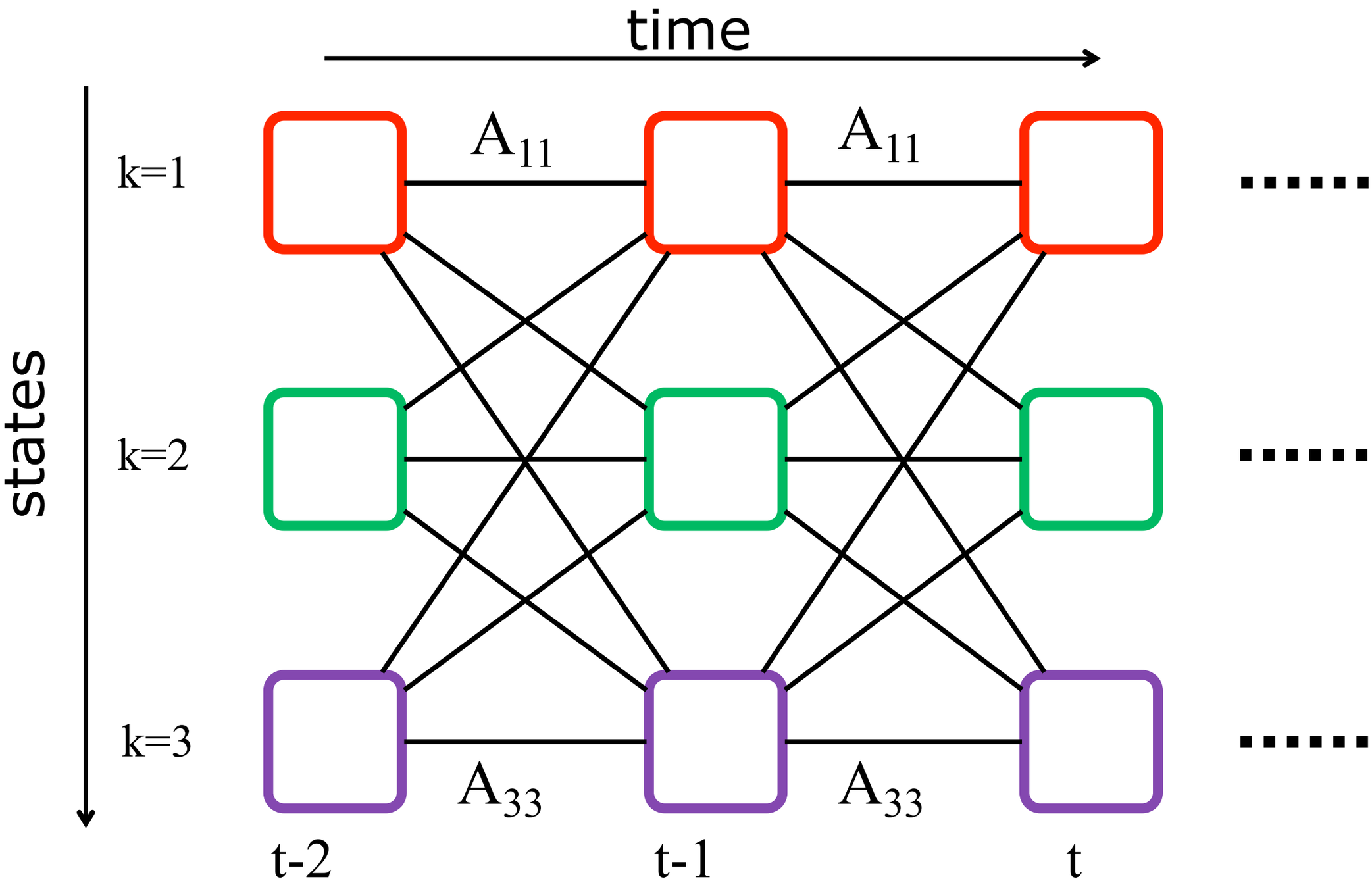
$$p(\mathbf{x}_1, \ldots, \mathbf{x}_T) = p(\mathbf{x}_1)p(\mathbf{x}_2 \mid \mathbf{x}_1) \cdots = p(\mathbf{x}_1) \prod_{t=2}^{T} p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$$

The graphical model of a Markov chain is this:

We will denote $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ as a row vector $\boldsymbol{\pi}_t$

A Markov chain can also be visualized as a **state transition diagram.**

The State Transition Diagram



The Stationary Distribution

The probability to reach state k is $\pi_{k,t} = \sum_{i=1}^{K} \pi_{i,t-1} A_{ik}$

Or, in matrix notation: $\boldsymbol{\pi}_t = \boldsymbol{\pi}_{t-1} A$

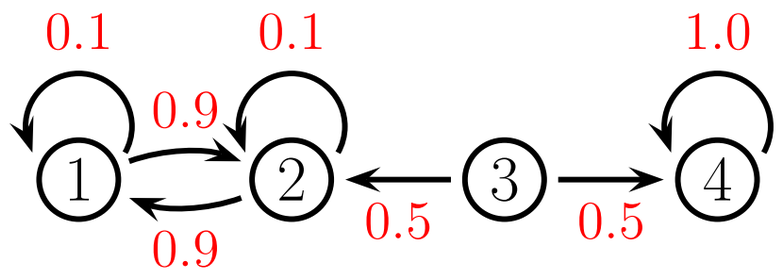
We say that $\boldsymbol{\pi}_t$ is **stationary** if $\boldsymbol{\pi}_t = \boldsymbol{\pi}_{t-1}$

To find the stationary distribution we need to solve the eigenvector problem $A^T \mathbf{v} = \mathbf{v}$.

The stationary distribution is then $\boldsymbol{\pi} = \mathbf{v}^T$ where $\mathbf{v}$ is the eigenvector for which the eigenvalue is 1.

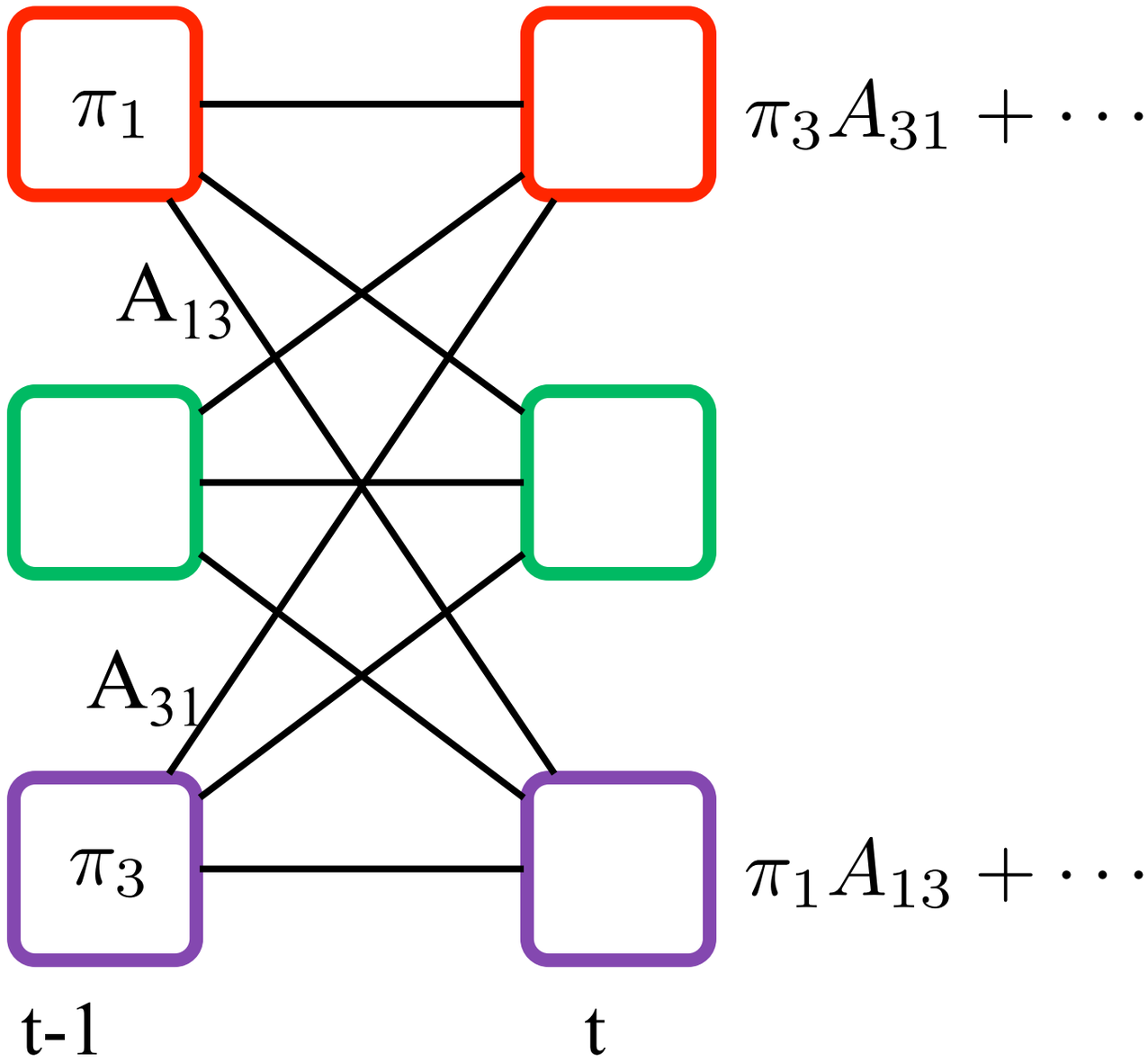
This eigenvector needs to be normalized so that it is a valid distribution.

Existence of a Stationary Distribution



- A Markov chain can have many stationary distributions
- Necessary for having a unique stationary distribution: we can reach every state from any other state in finite steps (the chain is **regular**)
- A transition distribution π_t satisfies the property of **detailed balance** if $\pi_i A_{ij} = \pi_j A_{ji}$
- The chain is then said to be **reversible**

Reversible Chain: Example



Existence of a Stationary Distribution

Theorem: If a Markov chain with transition matrix A is regular and satisfies detailed balance wrt. the distribution π, then π is a stationary distribution of the chain.

Proof:

$$\sum_{i=1}^{K} \pi_i A_{ij} = \sum_{i=1}^{K} \pi_j A_{ji} = \pi_j \sum_{i=1}^{K} A_{ji} = \pi_j \qquad \forall j$$

it follows $\boldsymbol{\pi} = \boldsymbol{\pi} A$.

This is only a sufficient condition, however it is not necessary.

Sampling with a Markov Chain

The main idea of MCMC is to sample state transitions based on a **proposal distribution** q.

The most widely used algorithm is the Metropolis-Hastings (MH) algorithm.

In MH, the decision whether to stay in a given state is based on a given probability.

If the proposal distribution is $q(\mathbf{x}' \mid \mathbf{x})$, then we stay in state $\mathbf{x}'$ with probability

$$\min\left(1, \frac{\tilde{p}(x')q(x \mid x')}{\tilde{p}(x)q(x' \mid x)}\right)$$

Unnormalized target distribution

The Metropolis-Hastings Algorithm

- Initialize x^0
- for $s = 0, 1, 2, \ldots$
  - define $x = x^s$
  - sample $x' \sim q(x' \mid x)$
    - compute acceptance probability

$$\alpha = \frac{\tilde{p}(x')q(x \mid x')}{\tilde{p}(x)q(x' \mid x)}$$

    - compute $r = \min(1, \alpha)$
    - sample $u \sim U(0, 1)$
    - set new sample to

$$x^{s+1} = \begin{cases} x' & \text{if} \quad u < r \\ x^s & \text{if} \quad u \geq r \end{cases}$$

Why Does This Work?

We have to prove that the transition probability of the MH algorithm satisfies detailed balance wrt the target distribution.

Theorem: If $p_{MH}(\mathbf{x}' \mid \mathbf{x})$ is the transition probability of the MH algorithm, then

$$p(\mathbf{x})p_{MH}(\mathbf{x}' \mid \mathbf{x}) = p(\mathbf{x}')p_{MH}(\mathbf{x} \mid \mathbf{x}')$$

Proof:

Why Does This Work?

We have to prove that the transition probability of the MH algorithm satisfies detailed balance wrt the target distribution.

Theorem: If $p_{MH}(\mathbf{x}' \mid \mathbf{x})$ is the transition probability of the MH algorithm, then

$$p(\mathbf{x})p_{MH}(\mathbf{x}' \mid \mathbf{x}) = p(\mathbf{x}')p_{MH}(\mathbf{x} \mid \mathbf{x}')$$

Note: All formulations are valid for discrete and for continuous variables!

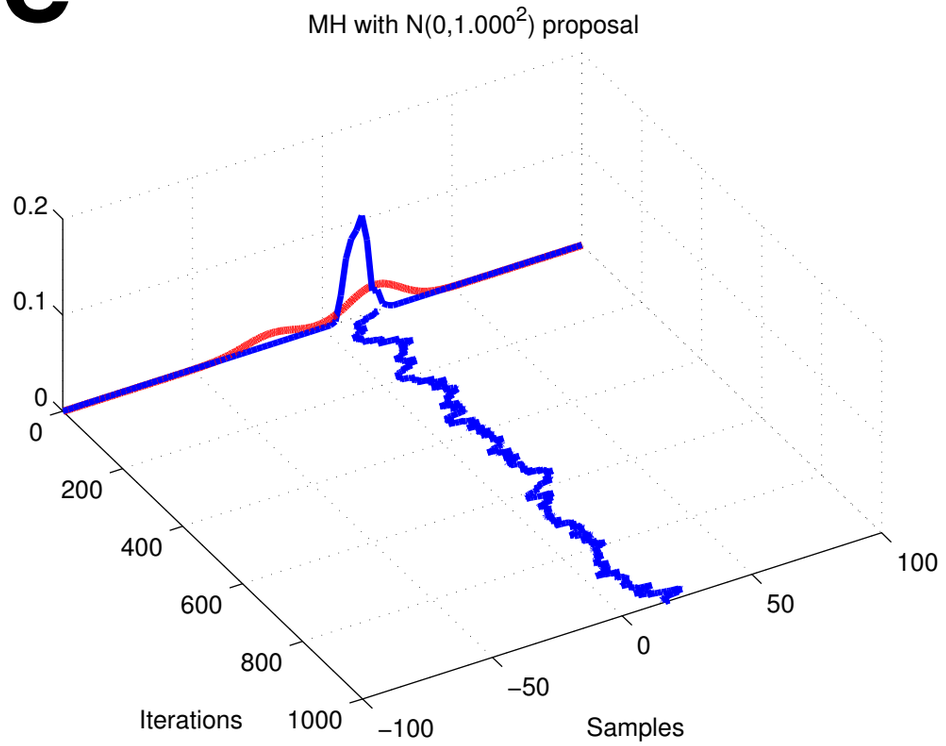
Choosing the Proposal

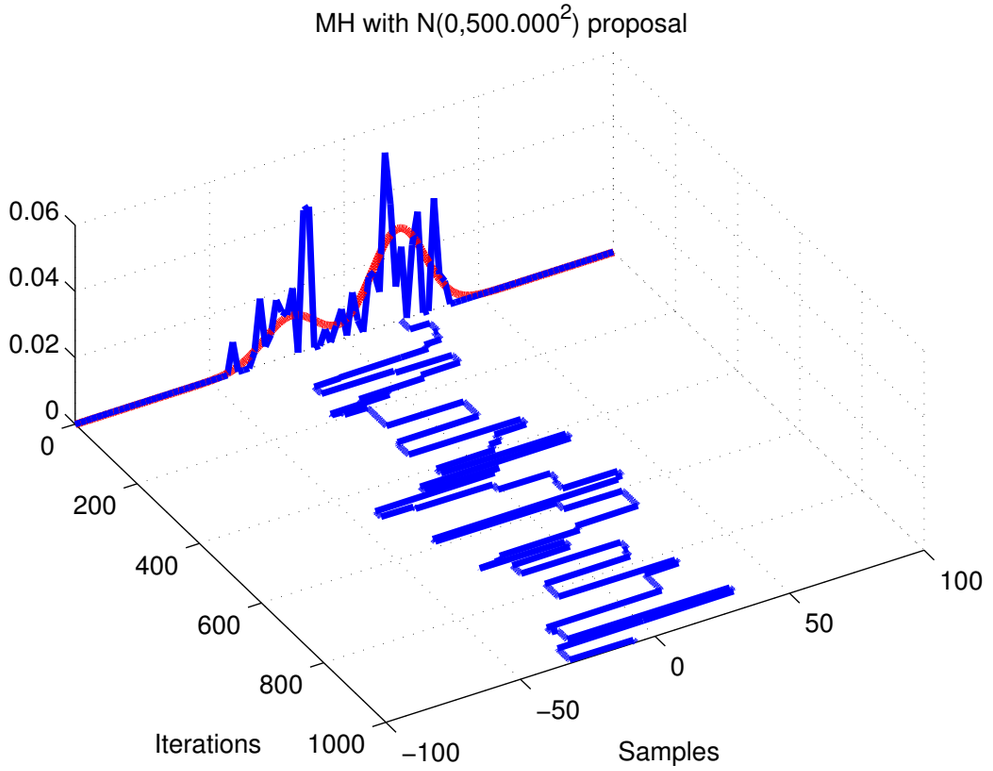
- A proposal distribution is valid if it gives a non-zero probability of moving to the states that have a non-zero probability in the target.
- A good proposal is the Gaussian, because it has a non-zero probability for all states.
- **However:** the variance of the Gaussian is important!
  - with low variance, the sampler does not explore sufficiently, e.g. it is fixed to a particular mode
  - with too high variance, the proposal is rejected too often, the samples are a bad approximation

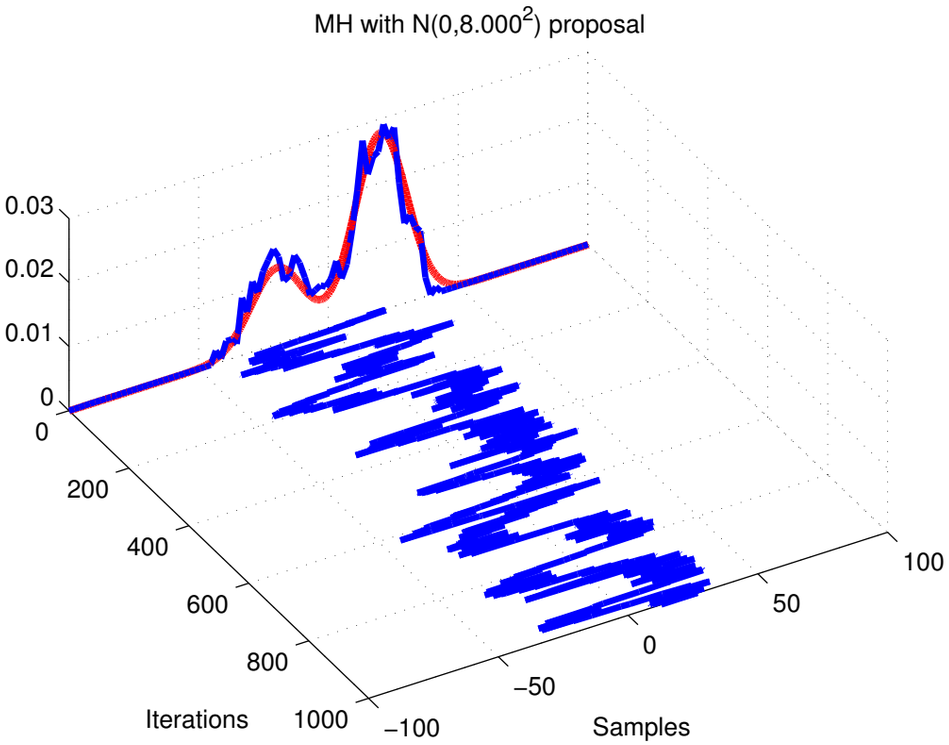
Example

Target is a mixture of 2 1D Gaussians.

Proposal is a Gaussian with different variances.







Gibbs Sampling

- Initialize $\{z_i : i = 1, \ldots, M\}$
- For $\tau = 1, \ldots, T$
  - Sample $z_1^{(\tau+1)} \sim p(z_1 \mid z_2^{(\tau)}, \ldots, z_M^{(\tau)})$
  - Sample $z_2^{(\tau+1)} \sim p(z_2 \mid z_1^{(\tau+1)}, \ldots, z_M^{(\tau)})$
  - ...
  - Sample $z_M^{(\tau+1)} \sim p(z_M \mid z_1^{(\tau+1)}, \ldots, z_{M-1}^{(\tau+1)})$

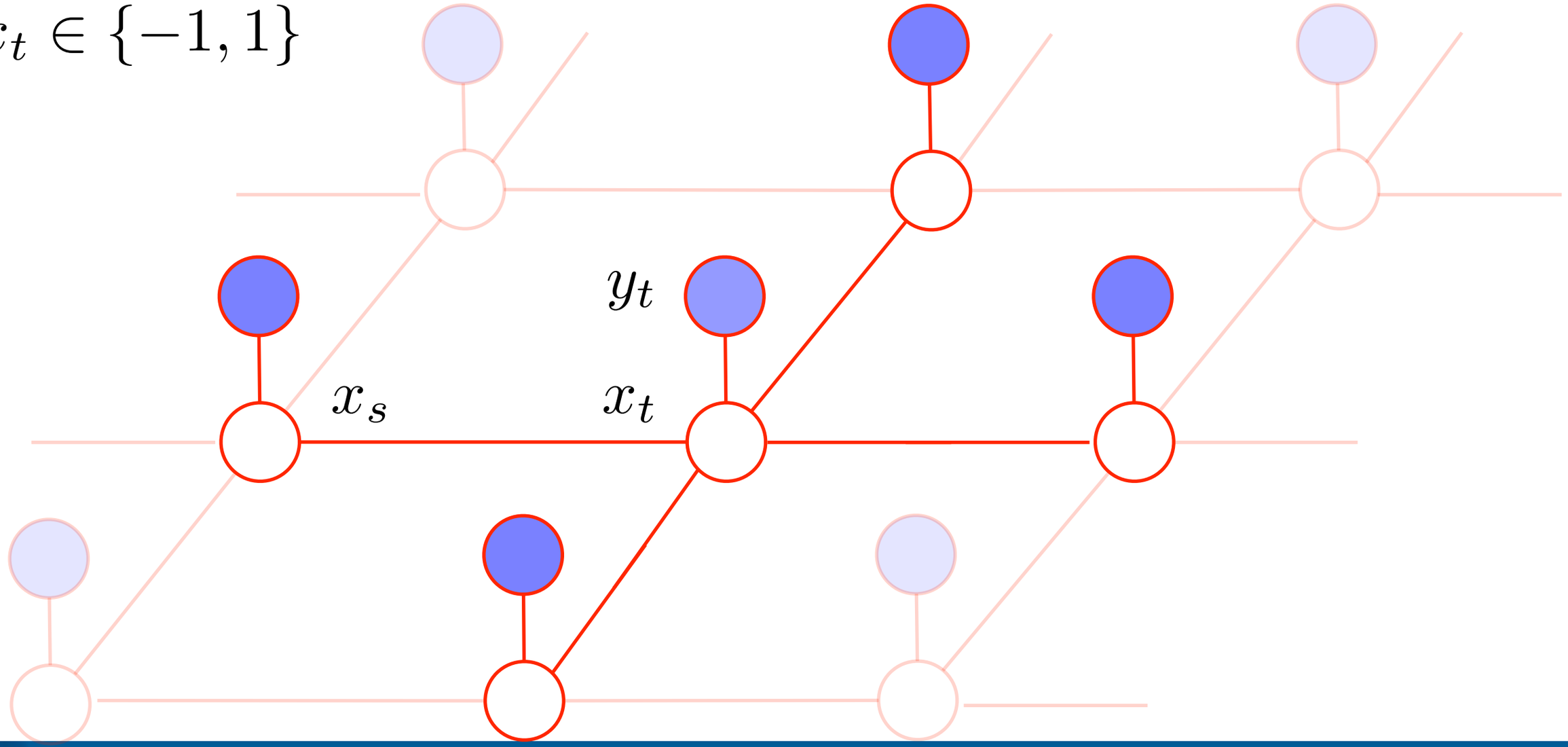
Idea: sample from the full conditional

This can be obtained, e.g. from the Markov blanket in graphical models.

Gibbs Sampling: Example

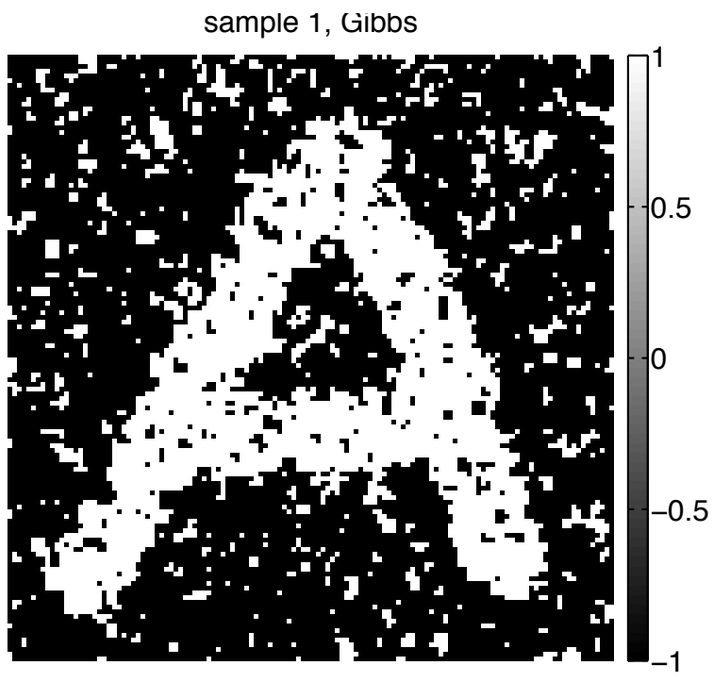
- Use an MRF on a binary image with edge potentials $\psi(x_s, x_t) = \exp(J x_s x_t)$ ("Ising model") and node potentials $\psi(x_t) = \mathcal{N}(y_t \mid x_t, \sigma^2)$

$x_t \in \{-1, 1\}$

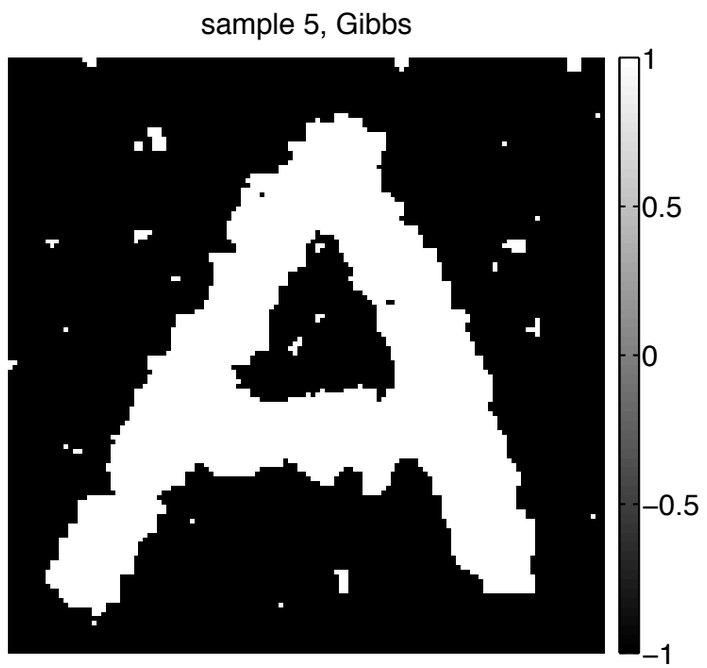


Gibbs Sampling: Example

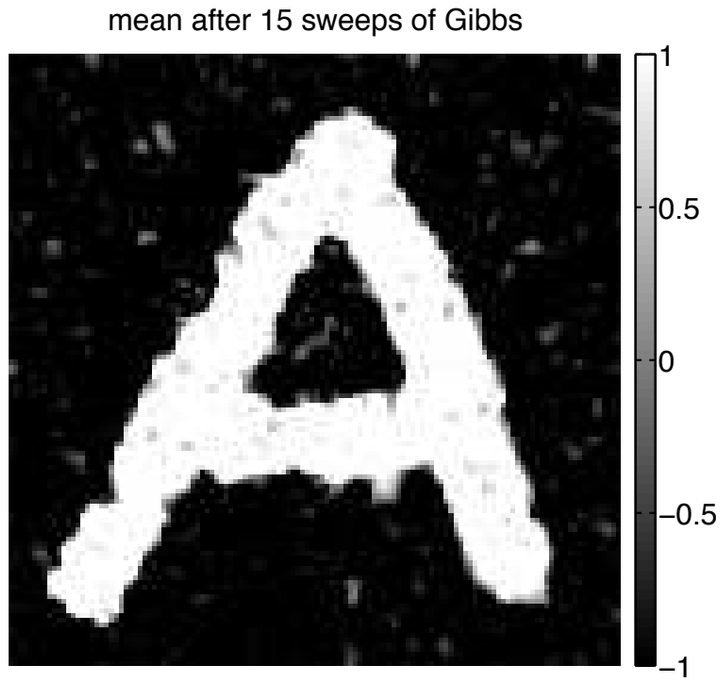
- Use an MRF on a binary image with edge potentials $\psi(x_s, x_t) = \exp(J x_s x_t)$ ("Ising model") and node potentials $\psi(x_t) = \mathcal{N}(y_t \mid x_t, \sigma^2)$
- Sample each pixel in turn



After 1 sample



After 5 samples



Average after 15 samples

Gibbs Sampling for GMMs

- Again, we start with the full joint distribution:

$$p(X, Z, \boldsymbol{\mu}, \Sigma, \boldsymbol{\pi}) = p(X \mid Z, \boldsymbol{\mu}, \Sigma) p(Z \mid \boldsymbol{\pi}) p(\boldsymbol{\pi}) \prod_{k=1}^{K} p(\boldsymbol{\mu}_k) p(\Sigma_k)$$

  (semi-conjugate prior)

- It can be shown that the full conditionals are:

$$p(z_i = k \mid \mathbf{x}_i, \boldsymbol{\mu}, \Sigma, \boldsymbol{\pi}) \propto \pi_k \mathcal{N}(\mathbf{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k)$$

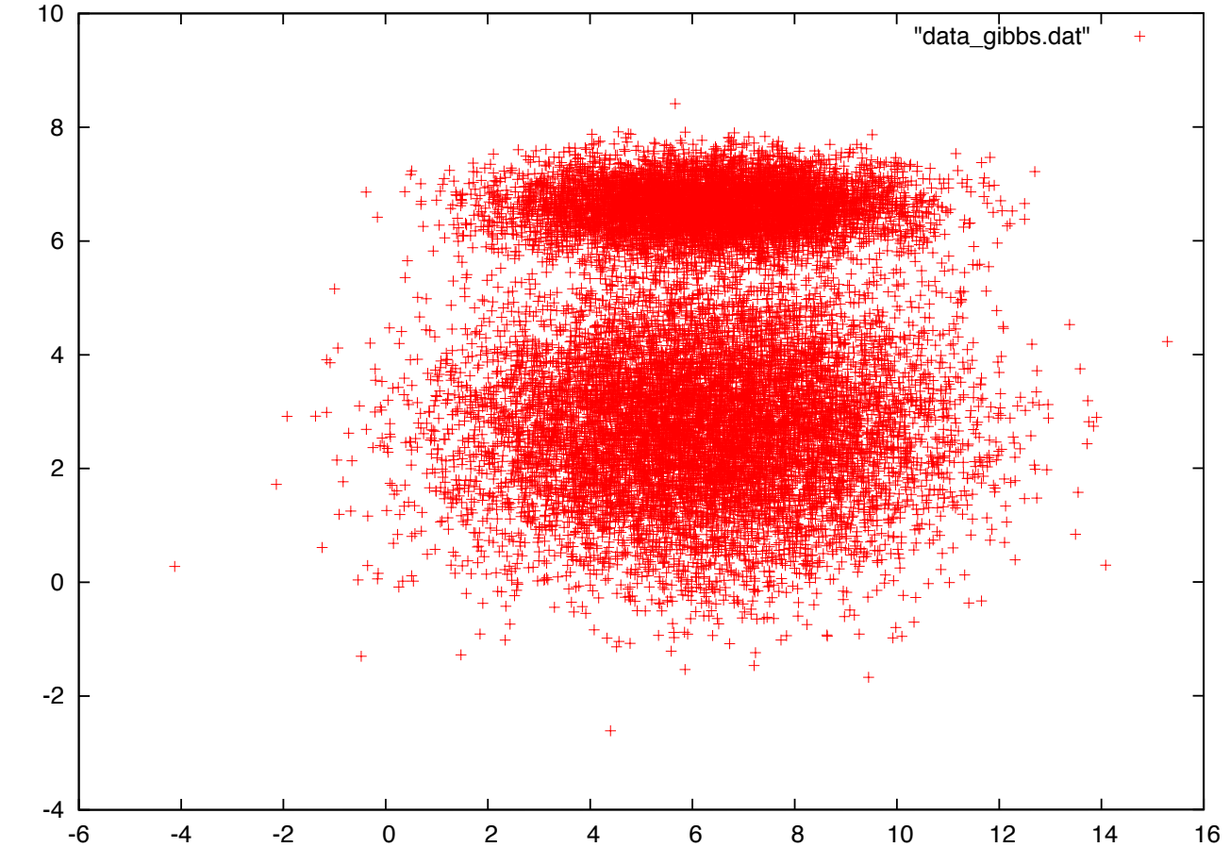
$$p(\boldsymbol{\pi} \mid \mathbf{z}) = \mathrm{Dir}(\{\alpha_k + \sum_{i=1}^{N} z_{ik}\}_{k=1}^{K})$$

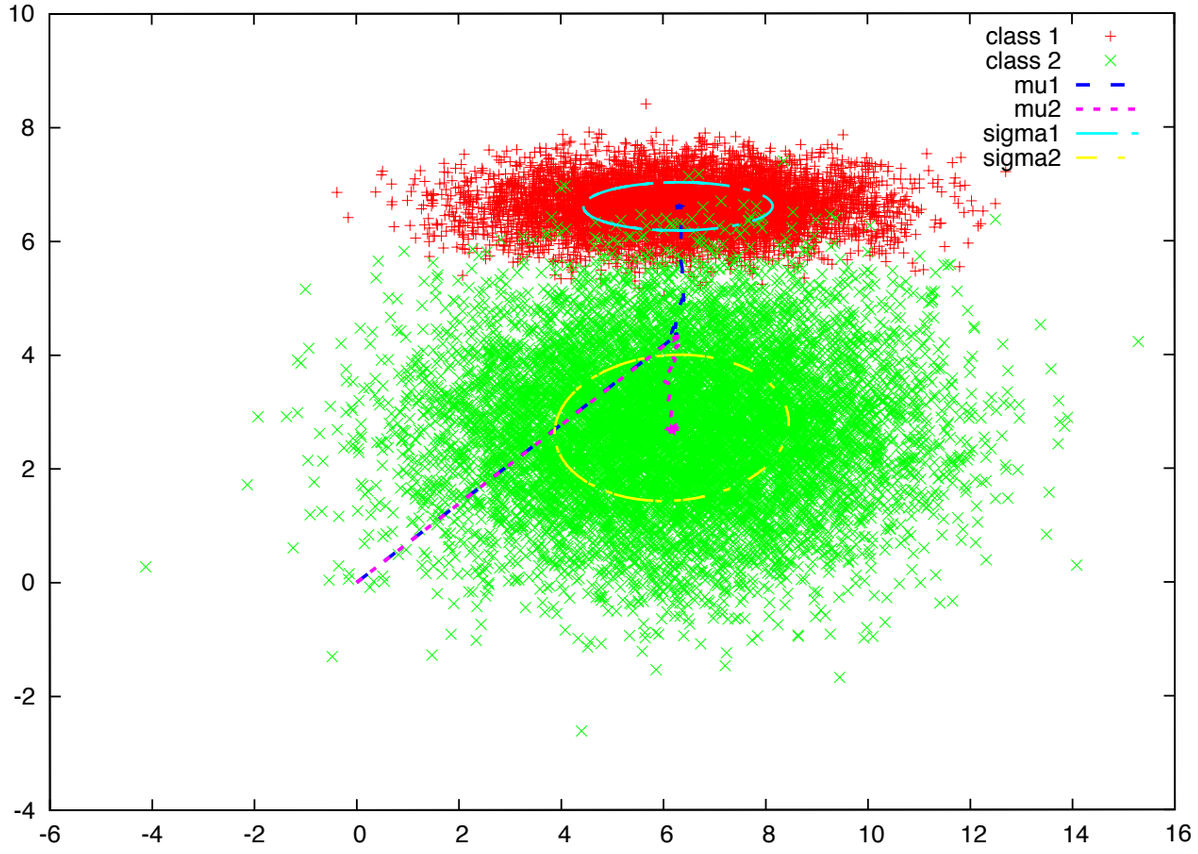
$$p(\boldsymbol{\mu}_k \mid \Sigma_k, Z, X) = \mathcal{N}(\boldsymbol{\mu}_k \mid \mathbf{m}_k, V_k) \quad \text{(linear-Gaussian)}$$

$$p(\Sigma_k \mid \boldsymbol{\mu}_k, Z, X) = \mathcal{IW}(\Sigma_k \mid S_k, \nu_k)$$

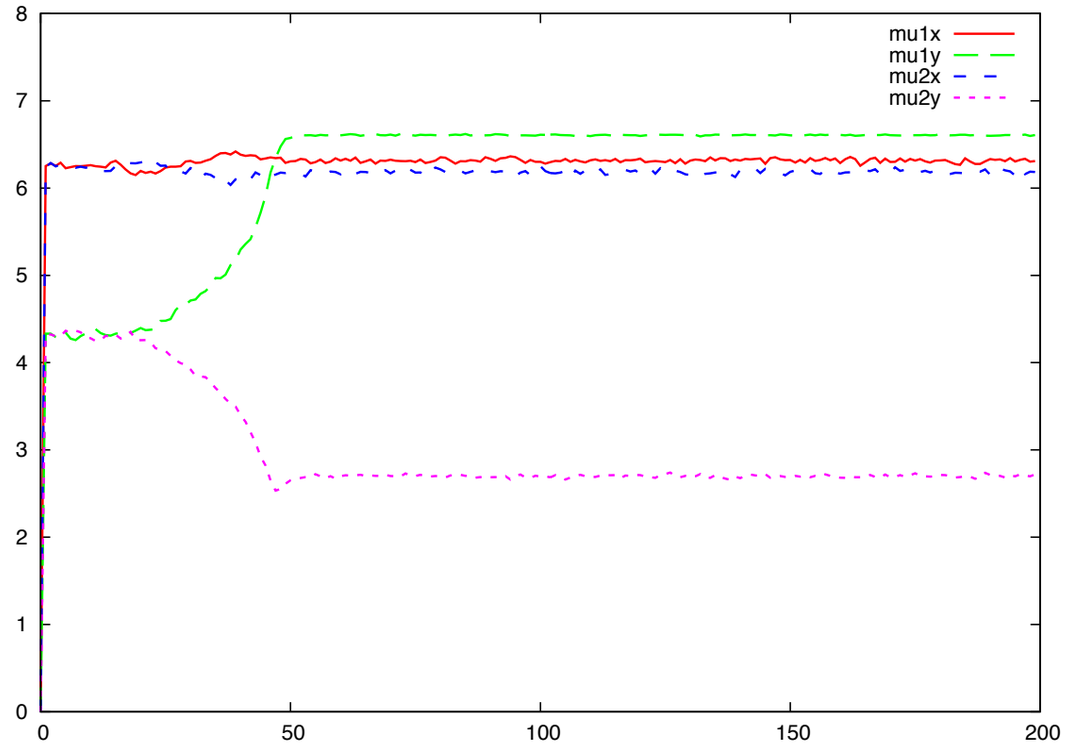
Gibbs Sampling for GMMs

- First, we initialize all variables
- Then we iterate over sampling from each conditional in turn
- In the end, we look at μ_k and Σ_k





How Often Do We Have To Sample?



- Here: after 50 sample rounds the values don't change any more
- In general, the **mixing time** τ_ϵ is related to the **eigen gap** $\gamma = \lambda_1 - \lambda_2$ of the transition matrix:

$$\tau_\epsilon \leq O(\frac{1}{\gamma} \log \frac{n}{\epsilon})$$

Gibbs Sampling is a Special Case of MH

- The proposal distribution in Gibbs sampling is

$$q(\mathbf{x}' \mid \mathbf{x}) = p(x'_i \mid \mathbf{x}_{-i})\mathbb{I}(\mathbf{x}'_{-i} = \mathbf{x}_{-i})$$

- This leads to an acceptance rate of:

$$\alpha = \frac{p(\mathbf{x}')q(\mathbf{x} \mid \mathbf{x}')}{p(\mathbf{x})q(\mathbf{x}' \mid \mathbf{x})} = \frac{p(x'_i \mid \mathbf{x}'_{-i})p(\mathbf{x}'_{-i})p(x_i \mid \mathbf{x}'_{-i})}{p(x_i \mid \mathbf{x}_{-i})p(\mathbf{x}_{-i})p(x'_i \mid \mathbf{x}_{-i})} = 1$$

- Although the acceptance is 100%, Gibbs sampling does not converge faster, as it only updates one variable at a time.

Summary

- Markov Chain Monte Carlo is a family of sampling algorithms that can sample from arbitrary distributions by moving in state space
- Most used methods are the Metropolis-Hastings (MH) and the Gibbs sampling method
- MH uses a proposal distribution and accepts a proposed state randomly
- Gibbs sampling does not use a proposal distribution, but samples from the full conditionals

11. Evaluation and Model Selection

Evaluation of Learning Methods

Very often, machine learning tries to find parameters of a model for a given data set.

But: Which parameters give a good model?

Intuitively, a good model behaves well on new, **unseen** data. This motivates the distinction of

Training data: used to generate different models

Validation data: used to adjust the model parameters so that their performance is optimal

Test data: used to evaluate the models with the optimized parameters

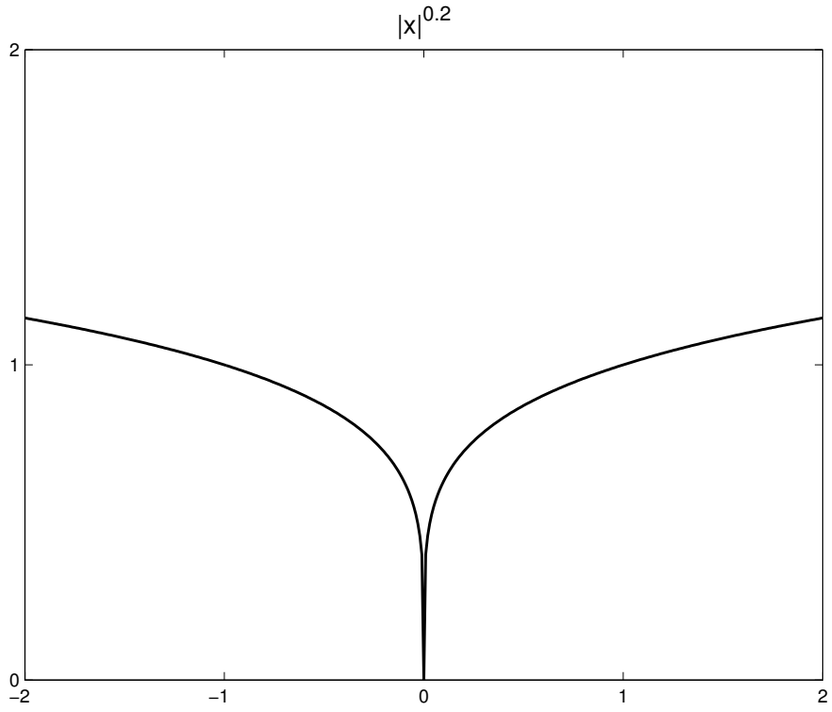
Loss Function

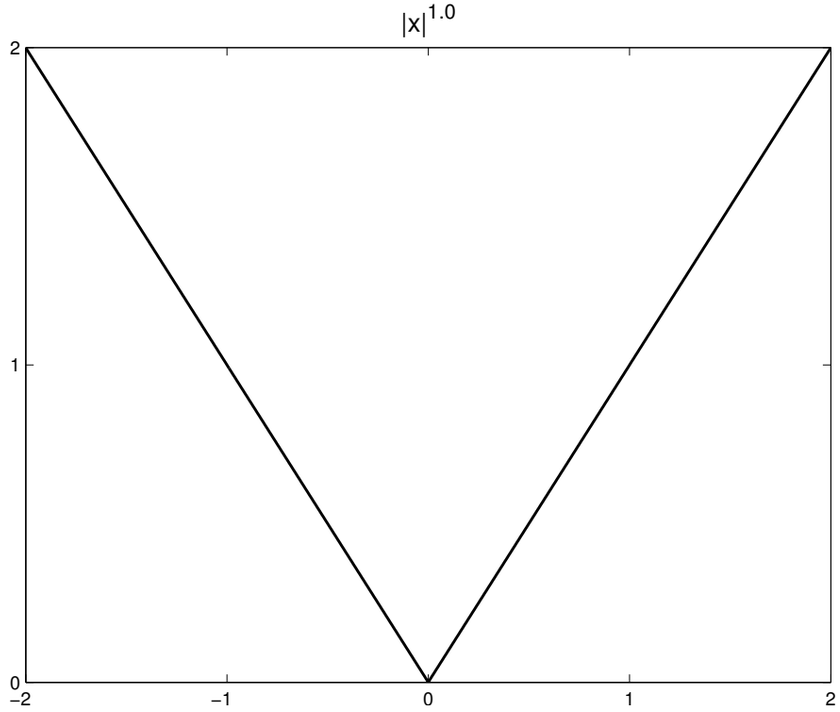
A common way to evaluate a learning algorithm (e.g. regression, classification) is to define a loss function:

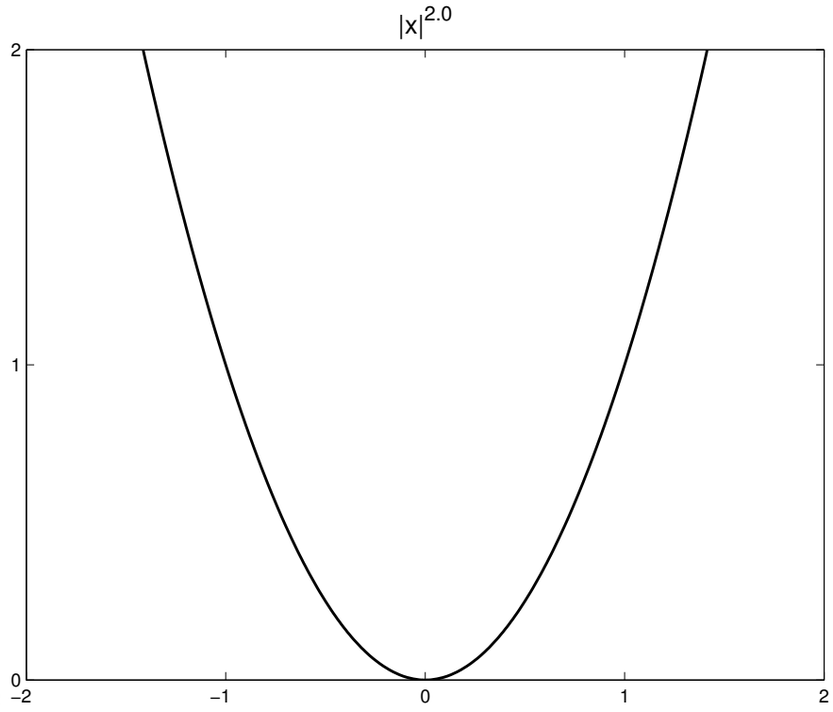
- In the case of regression, a common choice is the squared loss $L(\mathbf{w}, x, t) = (y(x, \mathbf{w}) - t)^2$
- In classification, where $y(x, \mathbf{w})$ and t are natural numbers, we use the 0/1-loss:

$$L(\mathbf{w}, x, t) = \begin{cases} 1 & \text{if } y(x, \mathbf{w}) \neq t \\ 0 & \text{otherwise} \end{cases}$$

Some Loss Functions







$$L(y, a) = |y - a|^q$$

a = “action”,
e.g. labeling

- Quadratic loss is useful for continuous parameters
- Its minimum is the posterior **mean** $\mathbb{E}[y \mid \mathbf{x}]$
- Absolute loss is less sensitive to outliers
- Its minimum is the **median** of the posterior

Model Selection

Model selection is used to find model parameters to optimize the classification result.

Possible methods:

- Minimizing the training error
- Hold-out testing
- Cross-validation
- Leave-one-out rule

Evaluation is done using an appropriate loss function.

Minimizing the Training Error

The training error is defined as:

$$E_T(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} L(\mathbf{w}, x_i, t_i)$$

Where (x_i, t_i) are all input/target value pairs from the training data set.

Problem: A model that minimizes the training error does not (necessarily) generalize well. It only behaves well on the data it was trained with.

Example: Polynomial regression with high model complexity (see above)

Hold-out Testing

Hold-out testing splits the data set up into

- A validation data set $\mathcal{S}_V$ of size K
- A smaller training data set $\mathcal{S}_T$ of size $N - K$

The model parameters are then selected so that the error

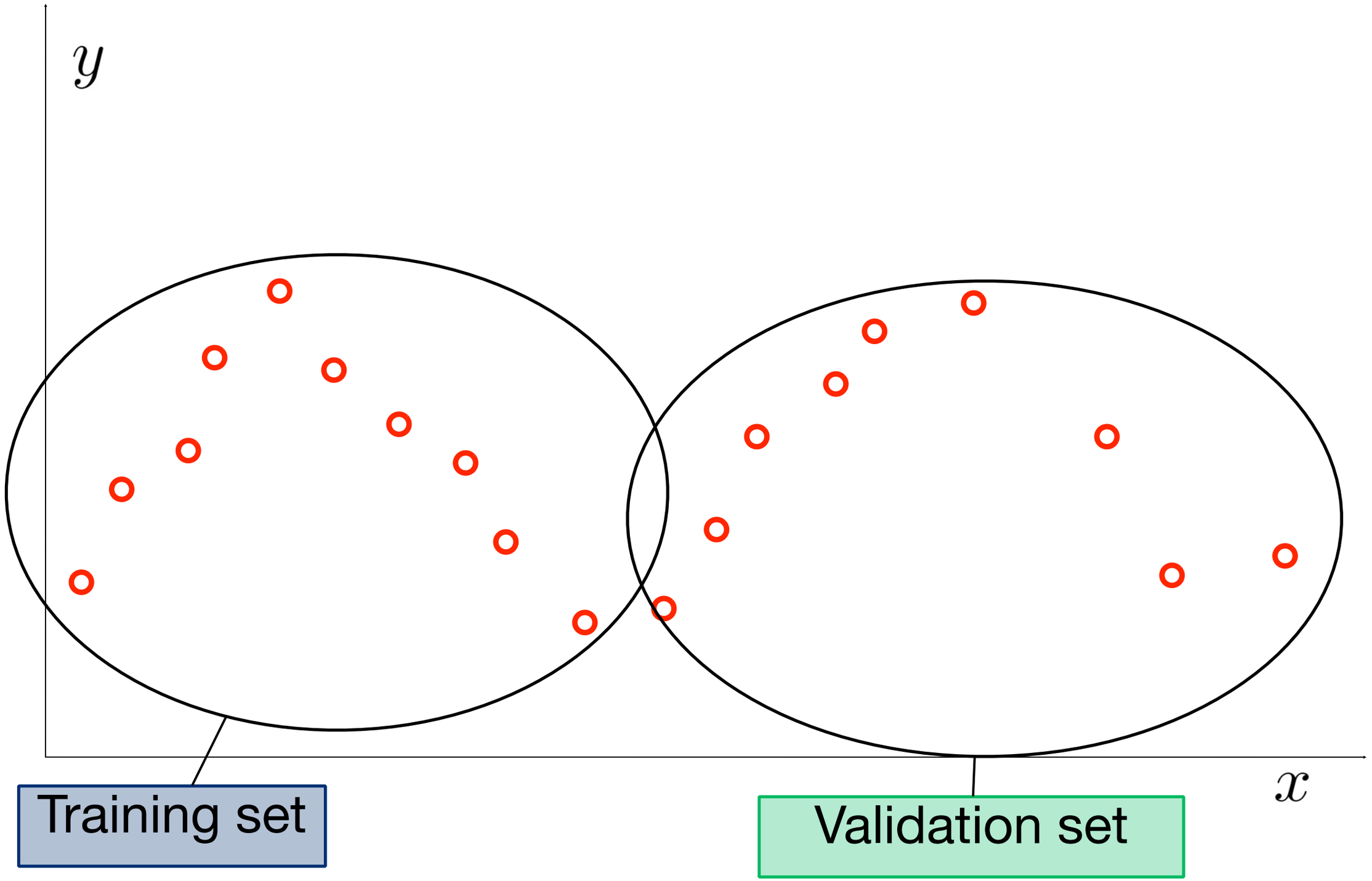
$$E_H(\mathbf{w}) = \frac{1}{K} \sum_{(x,t)\in\mathcal{S}_V} L(\mathbf{w}, x, t)$$

is minimized.

Problems: How big should be K ? Which elements should be chosen for evaluation?

Also: Iteration of the model design may lead to overfitting on the validation data

Hold-out Testing



Cross Validation

Idea of cross validation: Perform hold-out testing $\frac{N}{K}$ times on different evaluation (sub-)sets.

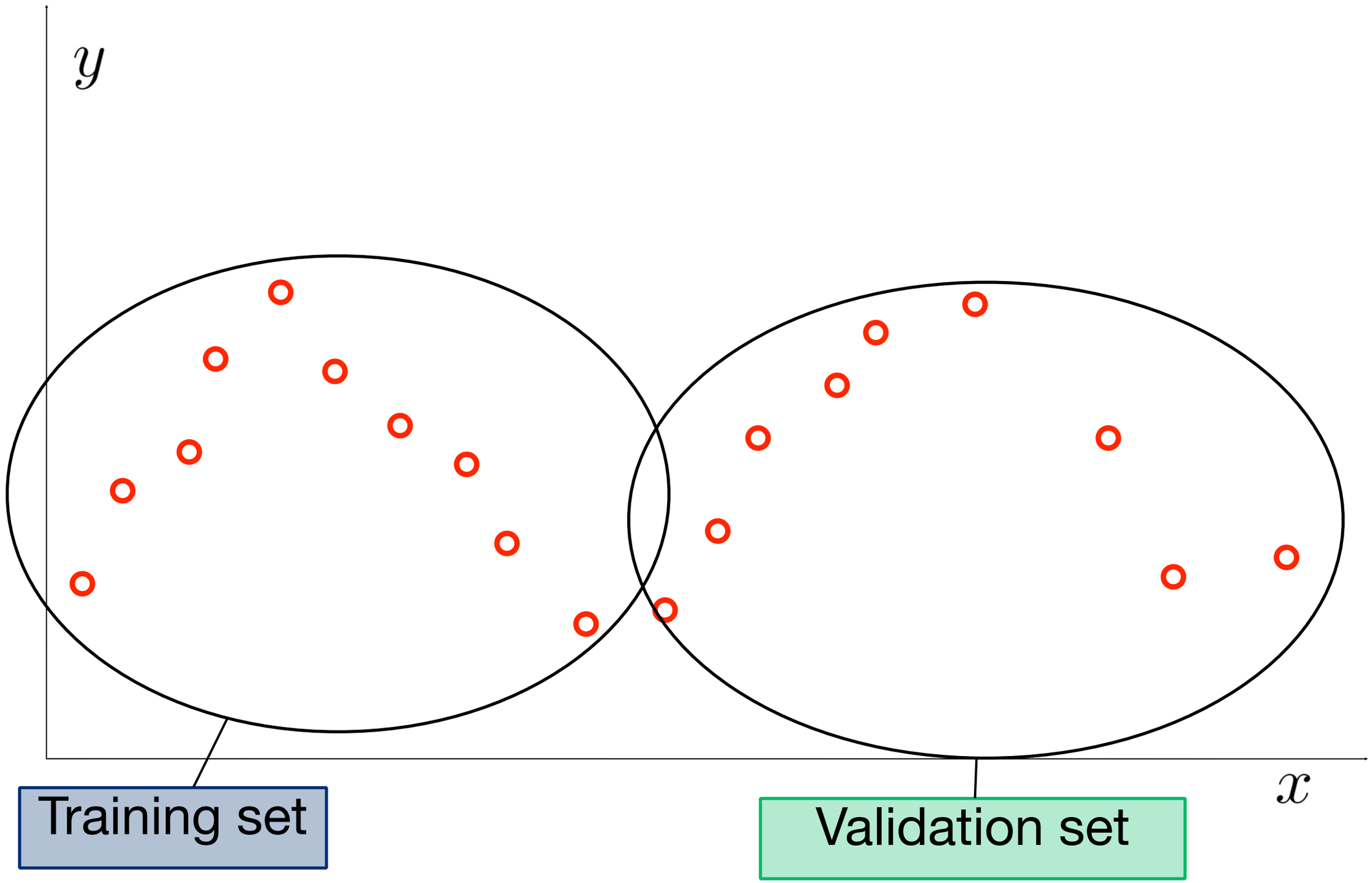
Validation subsets: $\mathcal{S}_V^1, \ldots, \mathcal{S}_V^M, \qquad M = \frac{N}{K}$

Error function:

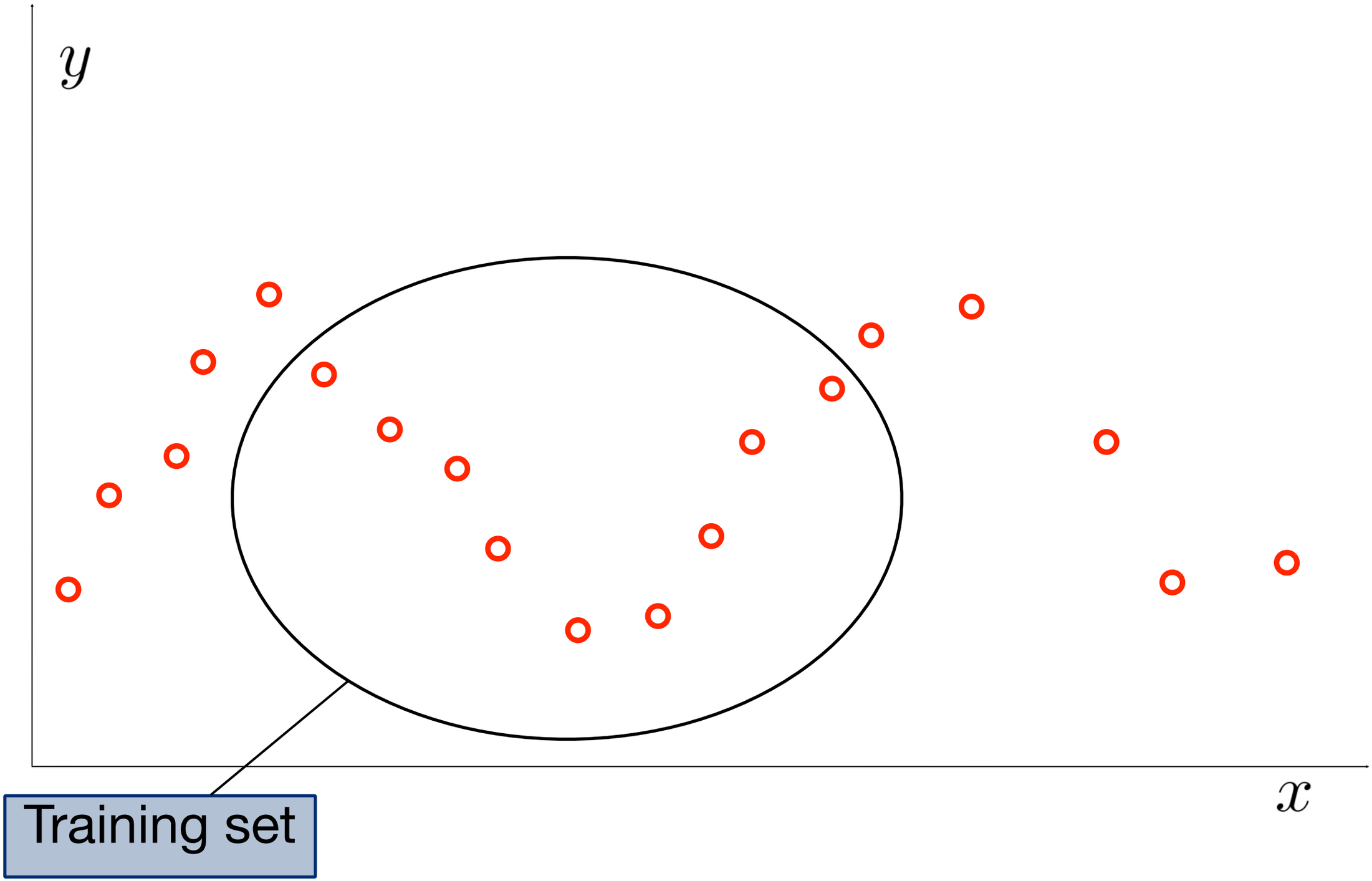
$$E_C(\mathbf{w}) = \frac{1}{K}\frac{1}{M}\sum_{i=1}^{M}\sum_{(x,t)\in\mathcal{S}_V^i} L(\mathbf{w}, x, t)$$

Minimizing this error function gives good results, but requires huge computational efforts. The training must be done M times.

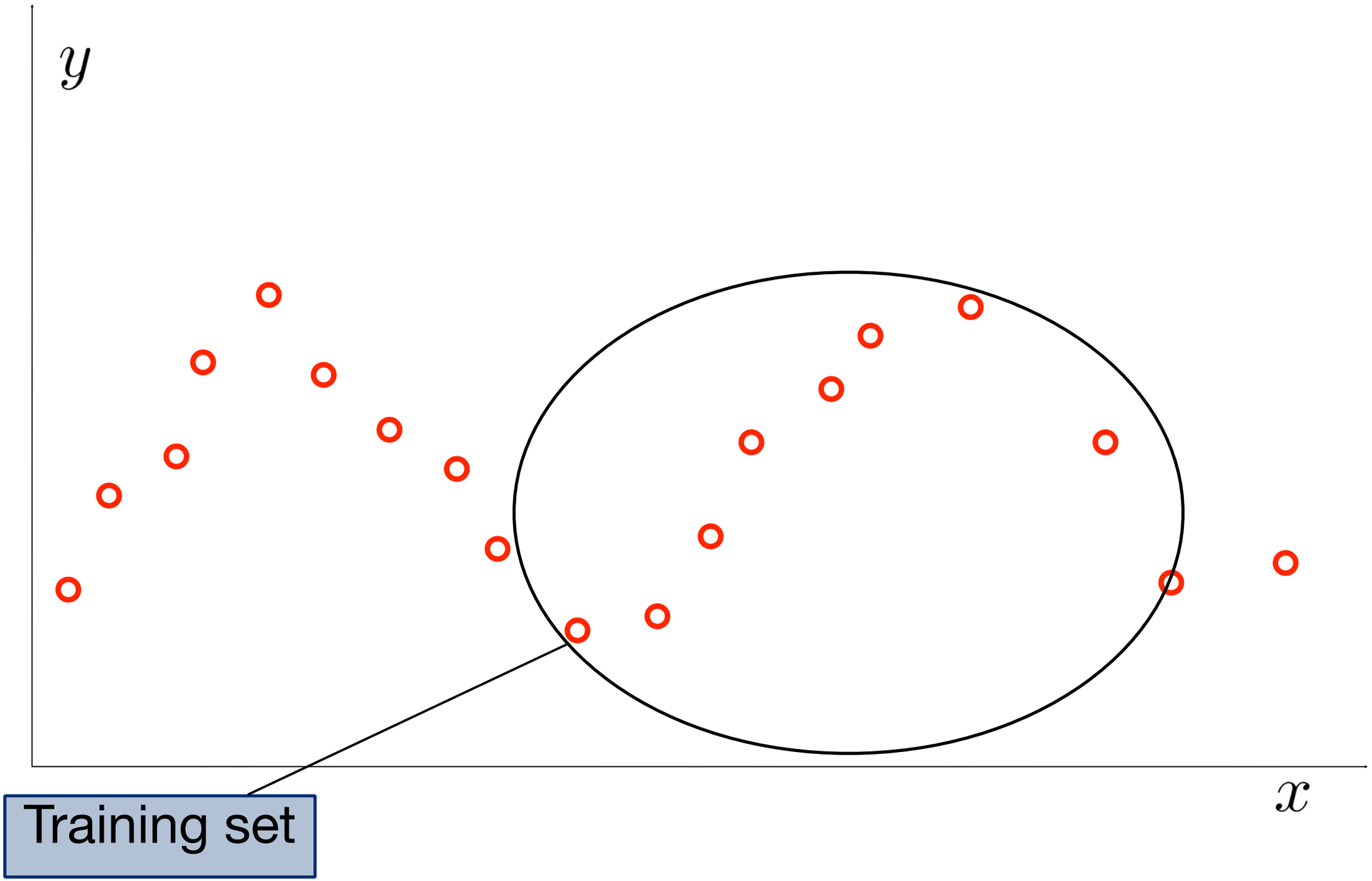
Cross-Validation



Cross-Validation



Cross-Validation



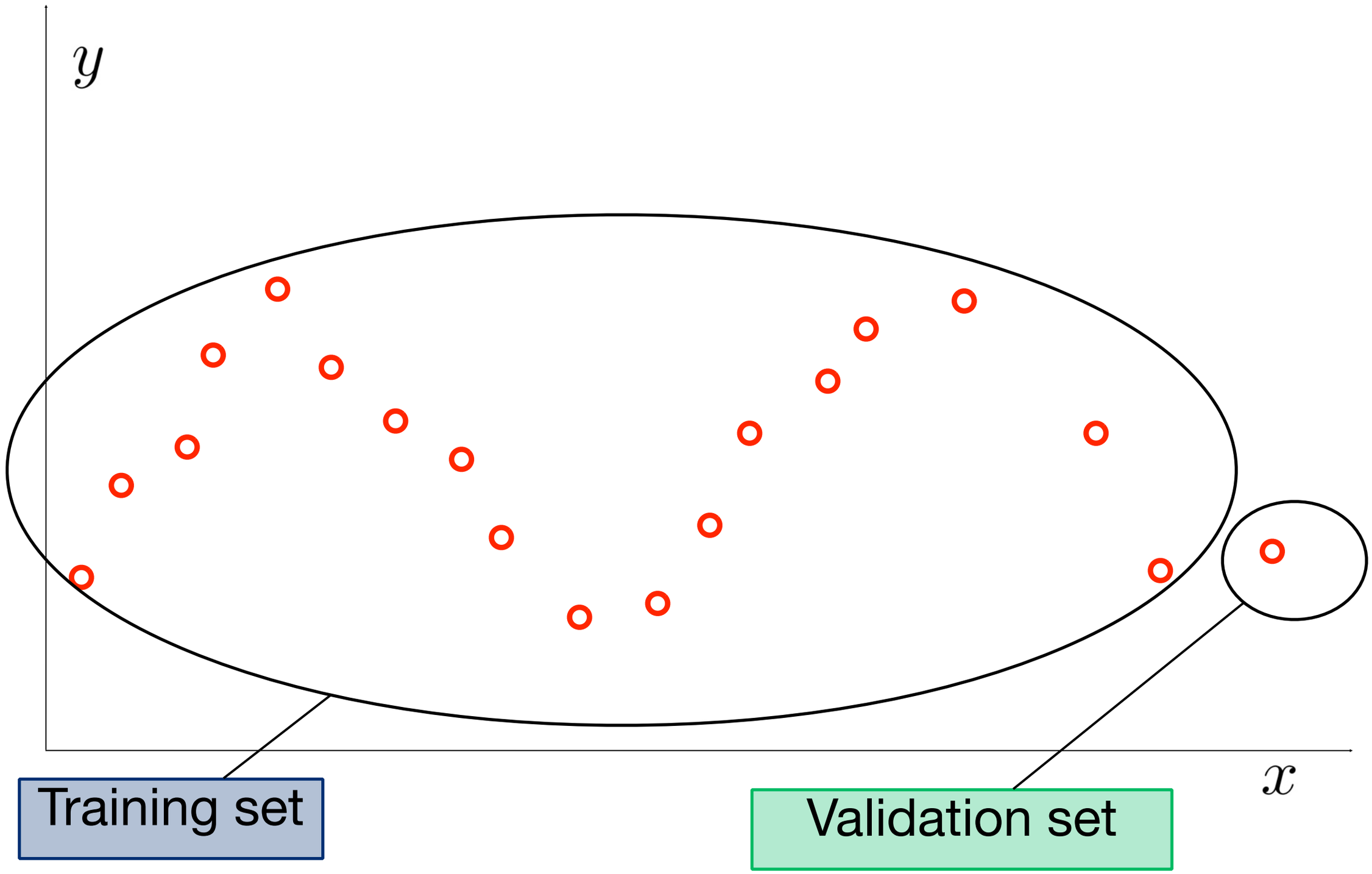
Leave-one-out Rule

Idea: do cross-validation with $\mathrm{K}=1$

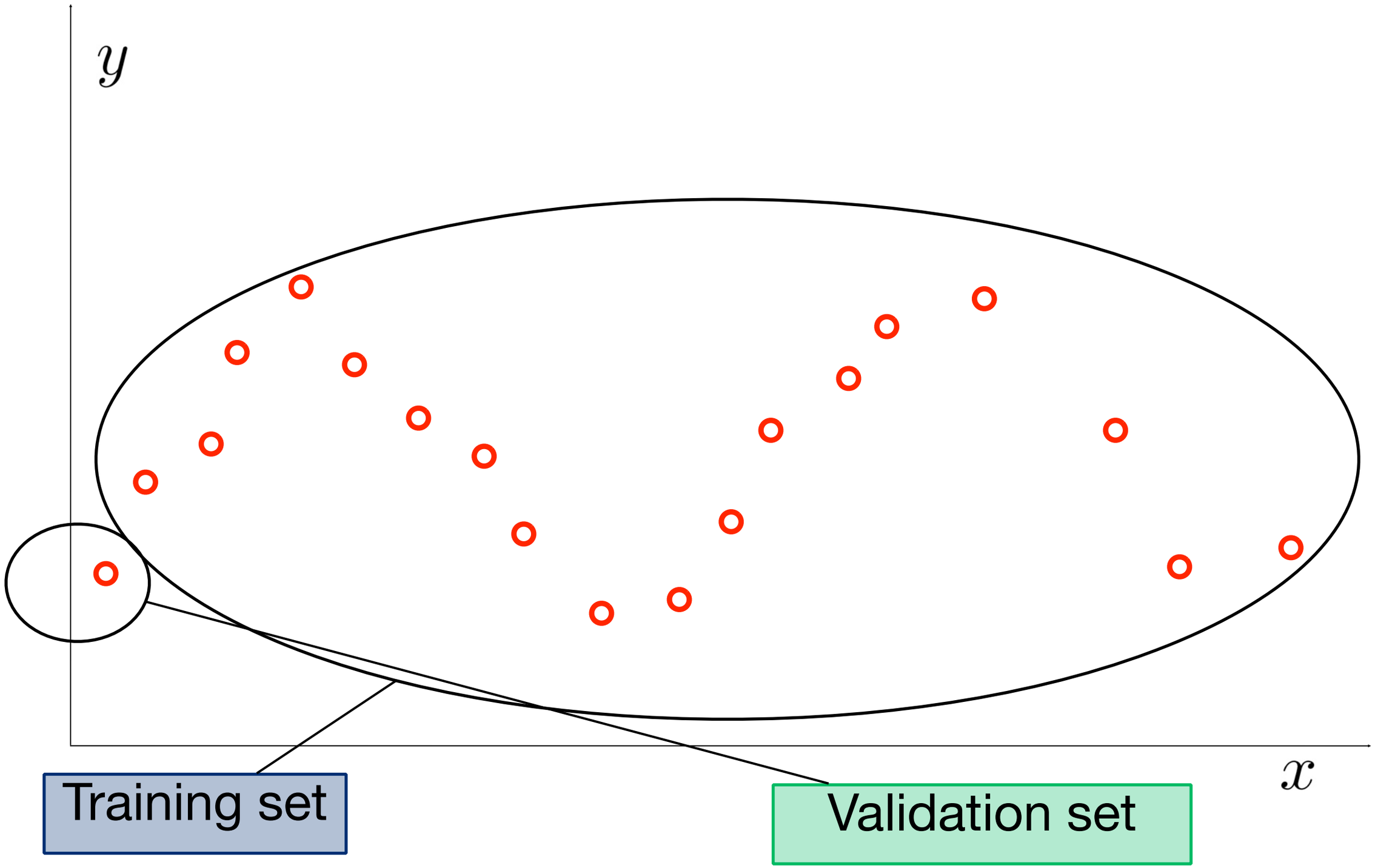
$$E_L(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{\mathcal{S}_T \setminus x_i} L(\mathbf{w}, x, t)$$

- Yields similarly good results compared to cross-validation
- Still requires to do the training N times
- Useful if the data set is particularly scarce

Leave-one-out Rule



Leave-one-out Rule



BIC and AIC

Observation: There is a trade-off between the obtained data likelihood (i.e. the quality of the fit) and the complexity M of the model.

Idea: Define a score function that balances both.

Two common examples are:

$$\ln p(\mathbf{t} \mid \mathbf{w}, \mathbf{x}) - M$$

"Akaike Information Criterion"

$$\ln p(\mathbf{t} \mid \mathbf{w}, \mathbf{x}) - \frac{1}{2} M \ln N$$

"Bayesian Information Criterion"

But: These criteria tend to favor too simple models.

Classifier Evaluation

Different methods to evaluate a classifier

- True-positive / false-positive rates
- Precision-recall diagram
- Receiving operator characteristics (ROC curves)
- Loss function

The evaluation of a classifier is needed to find the best classification parameters (e.g. for the kernel function)

Classifier Evaluation

Number of all data points: $N_P + N_N$	Classification result = -1	Classification result = 1
Correct label = -1	$\frac{T^-}{N_N}$ “True negative rate”	$\frac{F^+}{N_N}$ “False positive rate”
Correct label = 1	$\frac{F^-}{N_P}$ “False negative rate”	$\frac{T^+}{N_P}$ “True positive rate”

N_P Number of positive examples

N_N Number of negative examples

Classifier Evaluation

Terminology

- We define the **precision** as $$\text{prec} = \frac{T^+}{T^+ + F^+}$$

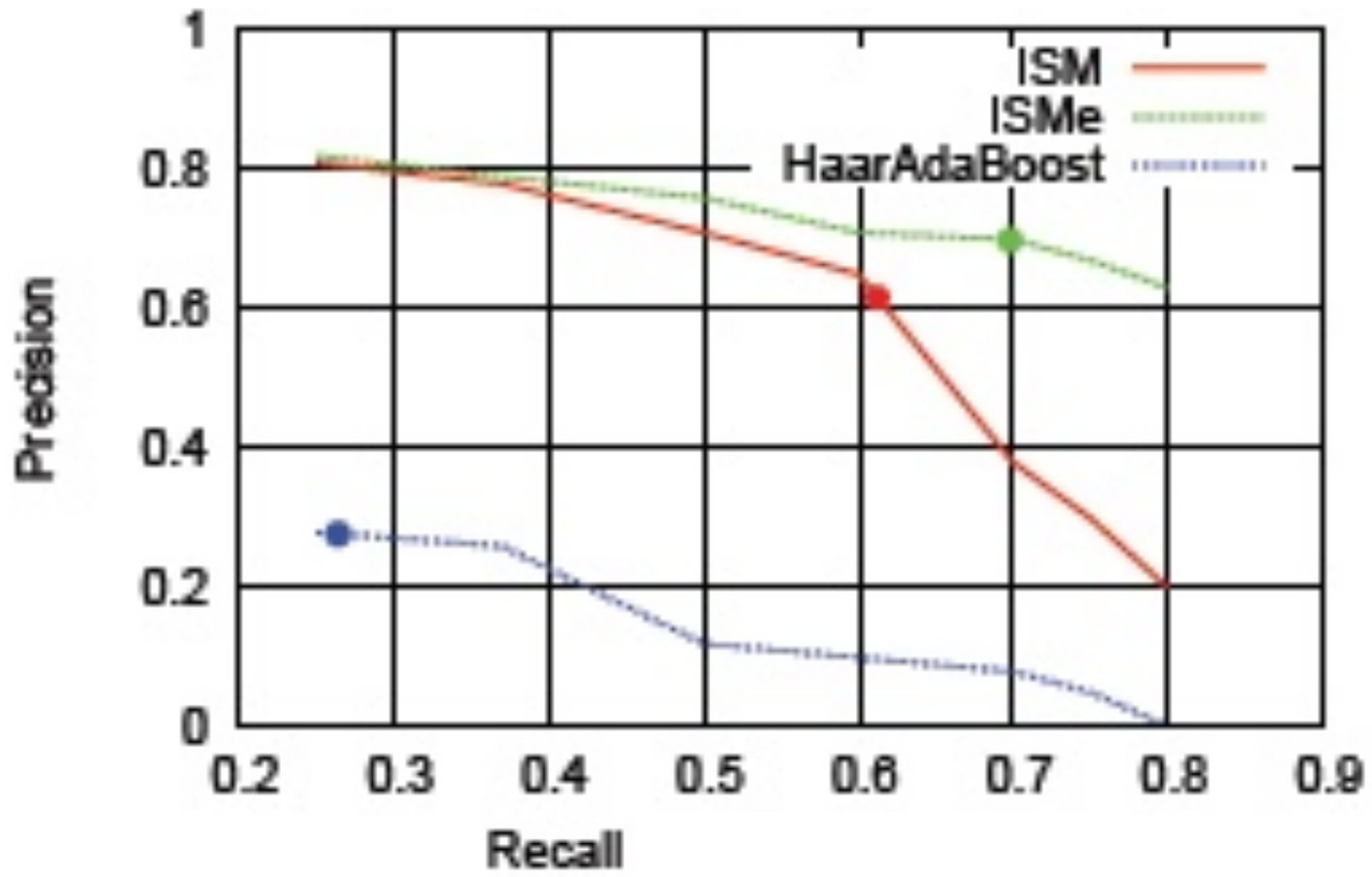
  intuitively: probability that the real label is positive in case the classifier returns “positive”

- The **recall** is defined as $$\text{rec} = \frac{T^+}{T^+ + F^-}$$

intuitively: probability that a “positive” labeled data point is detected as “positive” by the classifier.

Precision-Recall Curves

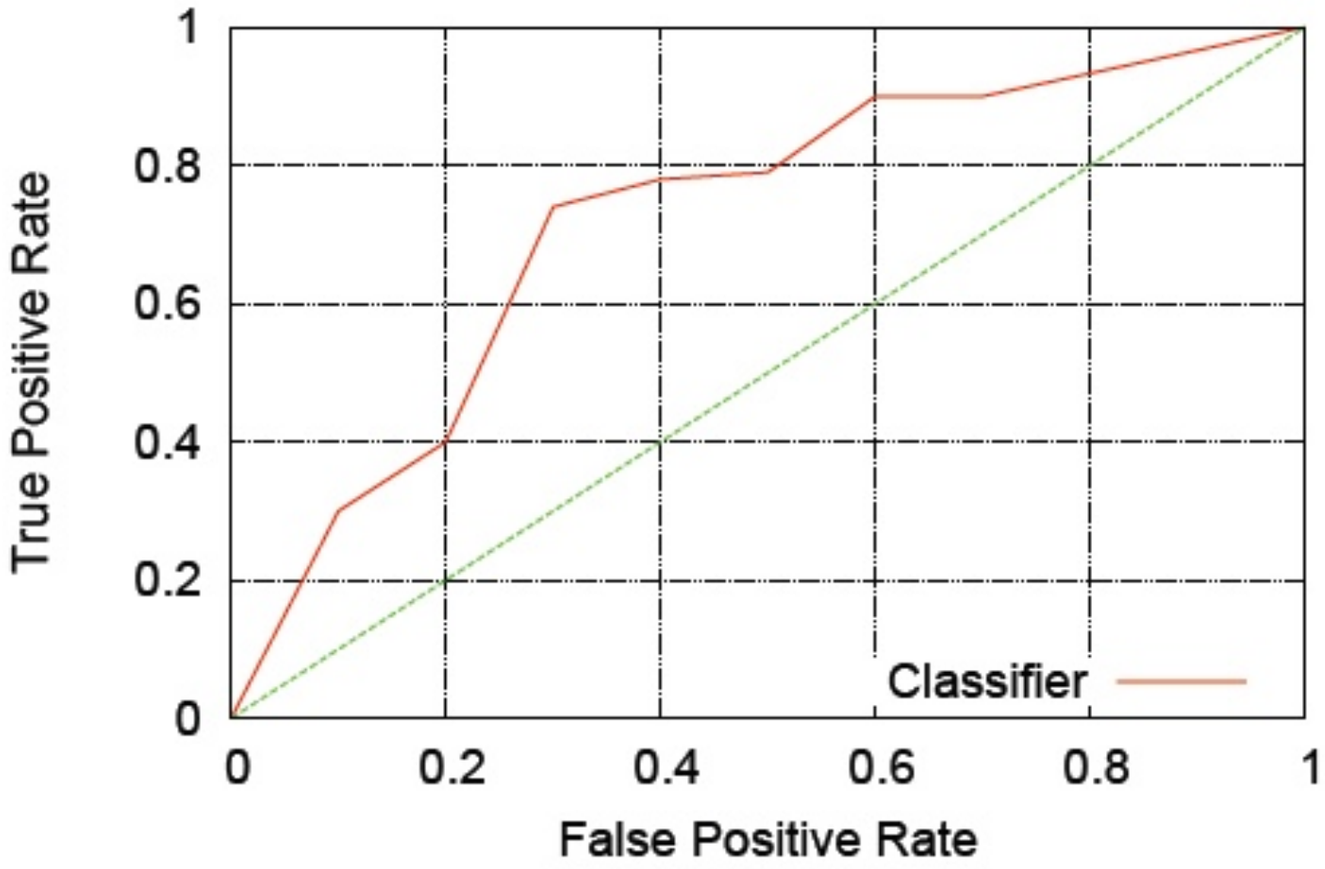
Usually, precision and recall are plotted into the same graph.



The optimal classifier parameters are obtained where precision equals recall (break-even-point).

Receiver Operating Characteristics

Receiving Operating Characteristics (ROC) curves plot the true positive rate vs. the false positive rate.



To find a good classifier we can search for a point where the slope is 1.

Loss Function

Another method to evaluate a classifier is defined by evaluating its **loss function**. The simplest loss function is the **0/1-loss:**

$$L(g, \mathbf{x}, y) = \begin{cases} 1 & \text{if } g(\mathbf{x}) \neq y \\ 0 & \text{otherwise} \end{cases}$$

Where g is a classifier, $\mathbf{x}$ is a feature vector and is the known class label of .

Important: The pair $(\mathbf{x}, y)$ should be taken from yn **evaluation data set** that i$\mathbf{x}$ different from the training set.