Introduction to Machine Learning

7. Kernels Methods

Geoff Gordon and Alex Smola
Carnegie Mellon University

http://alex.smola.org/teaching/cmu2013-10-701x
10-701

Carnegie Mellon University

Regression

"Under hypnosis you revealed that in your last eight lives you were ... er ... a cat."

Regression Estimation

- Find function f minimizing regression error

$$R[f] := \mathbf{E}_{x,y \sim p(x,y)} \left[l(y, f(x)) \right]$$

- Compute empirical average

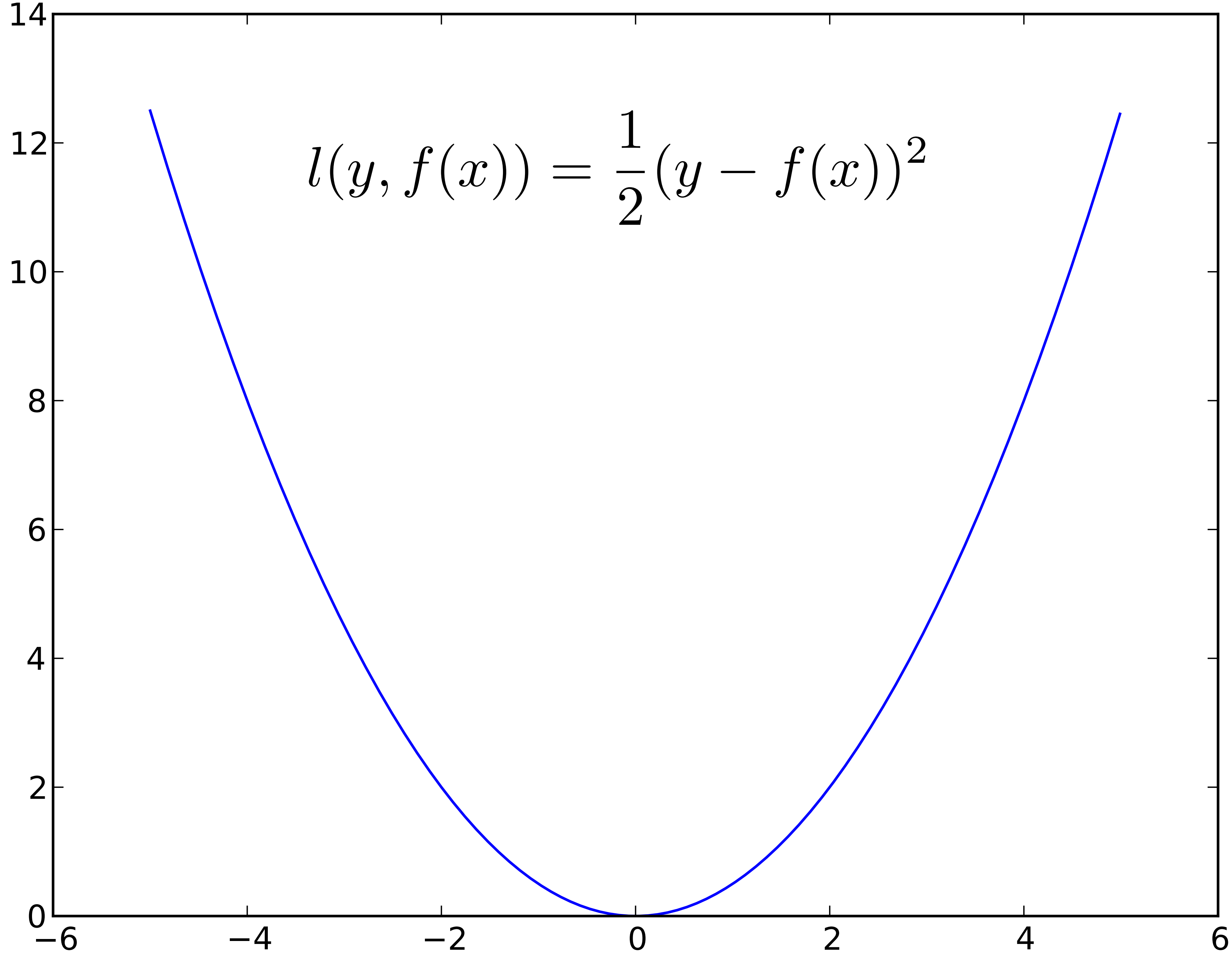
$$R_{\mathrm{emp}}[f] := \frac{1}{m} \sum_{i=1}^{m} l(y_i, f(x_i))$$

Overfitting as we minimize empirical error

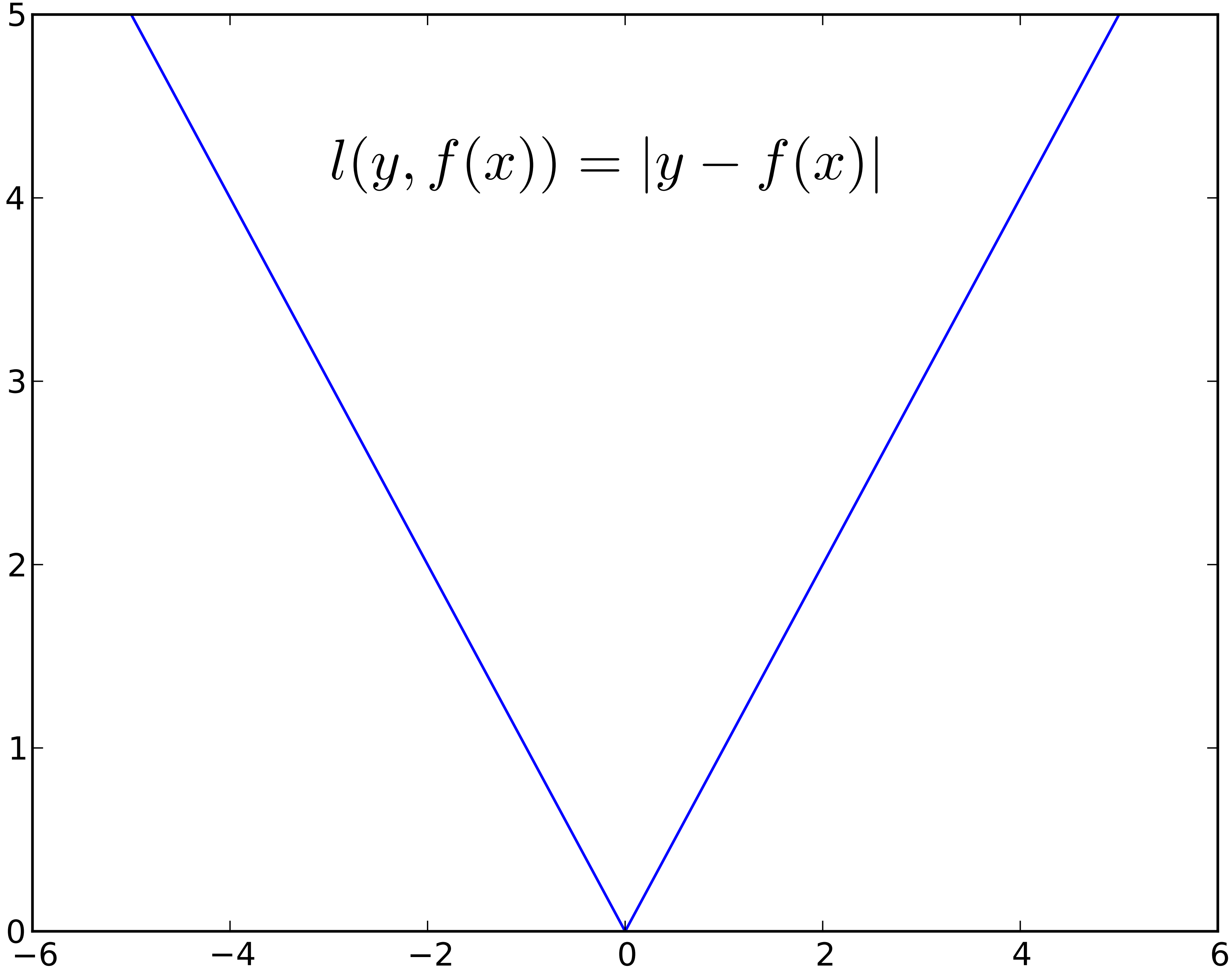
- Add regularization for capacity control

$$R_{\mathrm{reg}}[f] := \frac{1}{m} \sum_{i=1}^{m} l(y_i, f(x_i)) + \lambda \Omega[f]$$

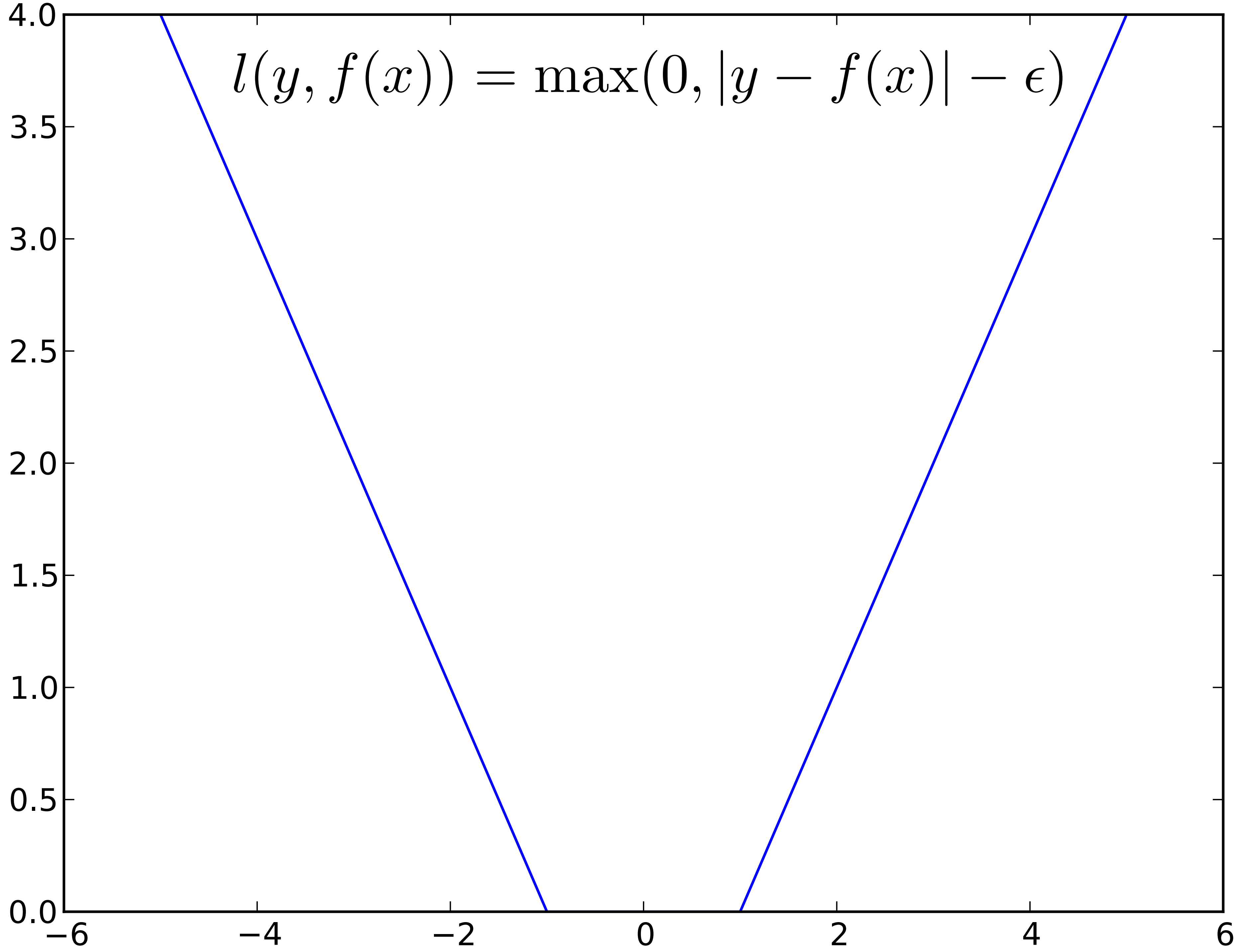
Squared loss



l1 loss



ε-insensitive Loss



Penalized least mean squares

- Optimization problem

$$\underset{w}{\text{minimize}} \frac{1}{2m} \sum_{i=1}^{m} (y_i - \langle x_i, w \rangle)^2 + \frac{\lambda}{2} \|w\|^2$$

- Solution

$$\partial_w [\ldots] = \frac{1}{m} \sum_{i=1}^{m} \left[x_i x_i^\top w - x_i y_i \right] + \lambda w$$

$$= \left[\frac{1}{m} X X^\top + \lambda \mathbf{1} \right] w - \frac{1}{m} X y = 0$$

$$\text{hence } w = \left[X X^\top + \lambda m \mathbf{1} \right]^{-1} X y$$

Outer product matrix in X

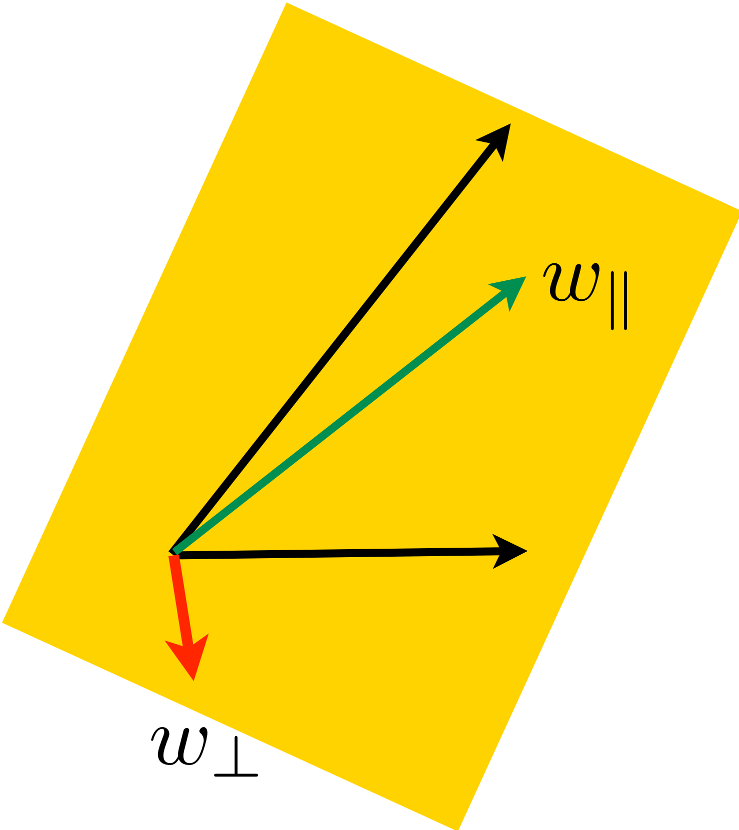
Conjugate Gradient
Sherman Morrison Woodbury

Penalized least mean squares ... now with kernels

- **Optimization problem**

$$\underset{w}{\text{minimize}} \frac{1}{2m} \sum_{i=1}^{m} (y_i - \langle \phi(x_i), w \rangle)^2 + \frac{\lambda}{2} \|w\|^2$$

- **Representer Theorem (Kimeldorf & Wahba, 1971)**



$$\|w\|^2 = \|w_{\|}\|^2 + \|w_{\perp}\|^2$$

empirical risk dependent

Penalized least mean squares ... now with kernels

- Optimization problem

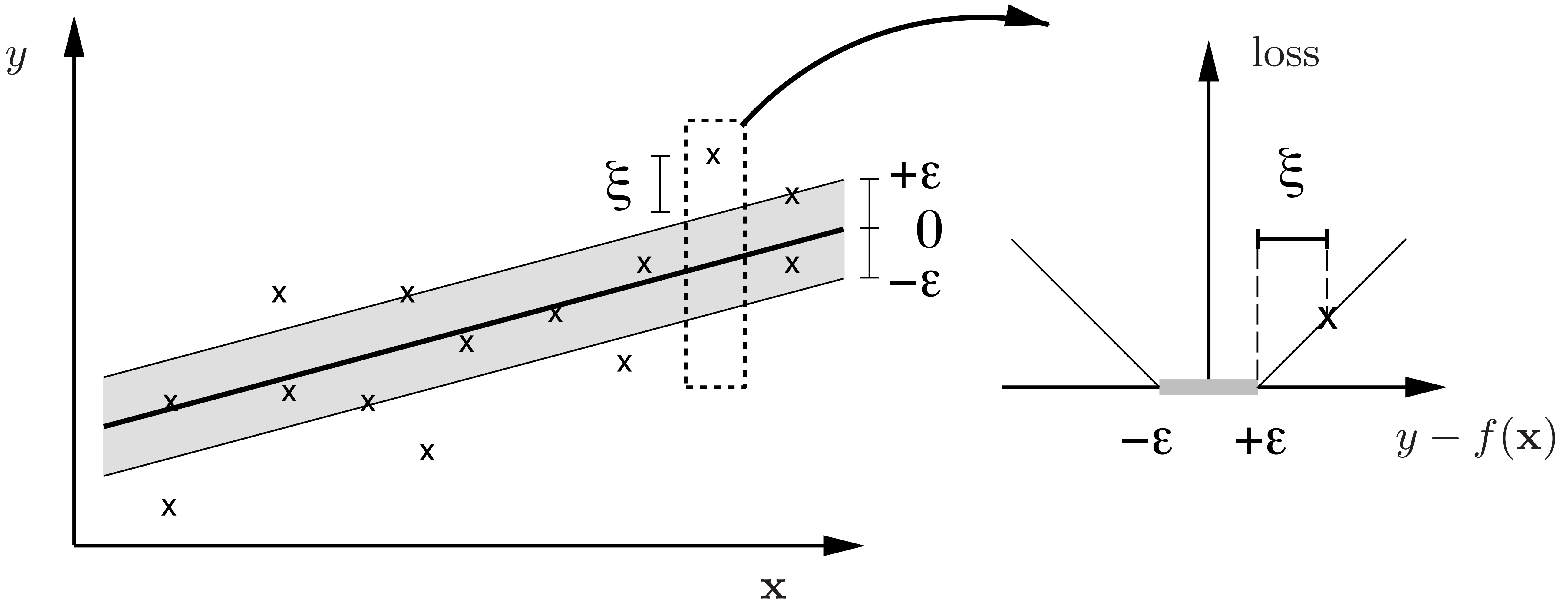
$$\underset{w}{\text{minimize}} \frac{1}{2m} \sum_{i=1}^{m} (y_i - \langle \phi(x_i), w \rangle)^2 + \frac{\lambda}{2} \|w\|^2$$

- Representer Theorem (Kimeldorf & Wahba, 1971)
  - Optimal solution is in span of data $w = \sum_i \alpha_i \phi(x_i)$
  - Proof - risk term only depends on data via $\phi(x_i)$
  - Regularization ensures that orthogonal part is 0
- Optimization problem in terms of w

$$\underset{\alpha}{\text{minimize}} \frac{1}{2m} \sum_{i=1}^{m} \left(y_i - \sum_j K_{ij} \alpha_j \right)^2 + \frac{\lambda}{2} \sum_{i,j} \alpha_i \alpha_j K_{ij}$$

solve for $\alpha = (K + m\lambda \mathbf{1})^{-1} y$ as linear system

SVM Regression
(ϵ-insensitive loss)



don't care about deviations within the tube

SVM Regression (ϵ-insensitive loss)

- **Optimization Problem (as constrained QP)**

$$\underset{w,b}{\text{minimize}} \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}[\xi_i + \xi_i^*]$$

$$\text{subject to } \langle w, x_i\rangle + b \leq y_i + \epsilon + \xi_i \text{ and } \xi_i \geq 0$$

$$\langle w, x_i\rangle + b \geq y_i - \epsilon - \xi_i^* \text{ and } \xi_i^* \geq 0$$

- **Lagrange Function**

$$L = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}[\xi_i + \xi_i^*] - \sum_{i=1}^{m}[\eta_i\xi_i + \eta_i^*\xi_i^*] +$$

$$\sum_{i=1}^{m}\alpha_i\left[\langle w, x_i\rangle + b - y_i - \epsilon - \xi_i\right] + \sum_{i=1}^{m}\alpha_i^*\left[y_i - \epsilon - \xi_i^* - \langle w, x_i\rangle - b\right]$$

SVM Regression (ϵ-insensitive loss)

- First order conditions

$$\partial_w L = 0 = w + \sum_i [\alpha_i - \alpha_i^*] x_i$$

$$\partial_b L = 0 = \sum_i [\alpha_i - \alpha_i^*]$$

$$\partial_{\xi_i} L = 0 = C - \eta_i - \alpha_i$$

$$\partial_{\xi_i^*} L = 0 = C - \eta_i^* - \alpha_i^*$$

- Dual problem

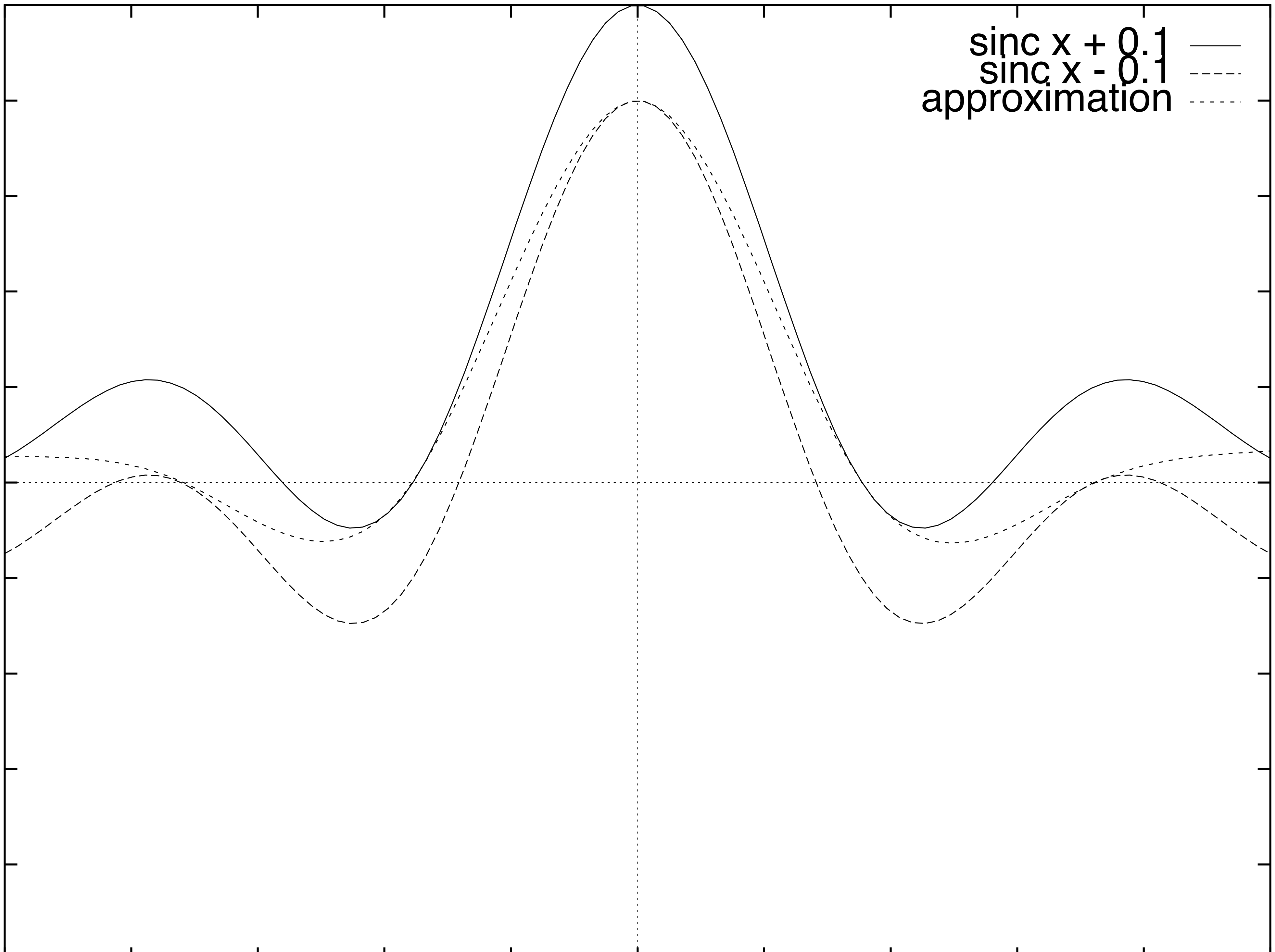
$$\underset{\alpha, \alpha^*}{\text{minimize}} \ \frac{1}{2}(\alpha - \alpha^*)^\top K (\alpha - \alpha^*) + \epsilon 1^\top (\alpha + \alpha^*) + y^\top (\alpha - \alpha^*)$$

$$\text{subject to } 1^\top (\alpha - \alpha^*) = 0 \text{ and } \alpha_i, \alpha_i^* \in [0, C]$$

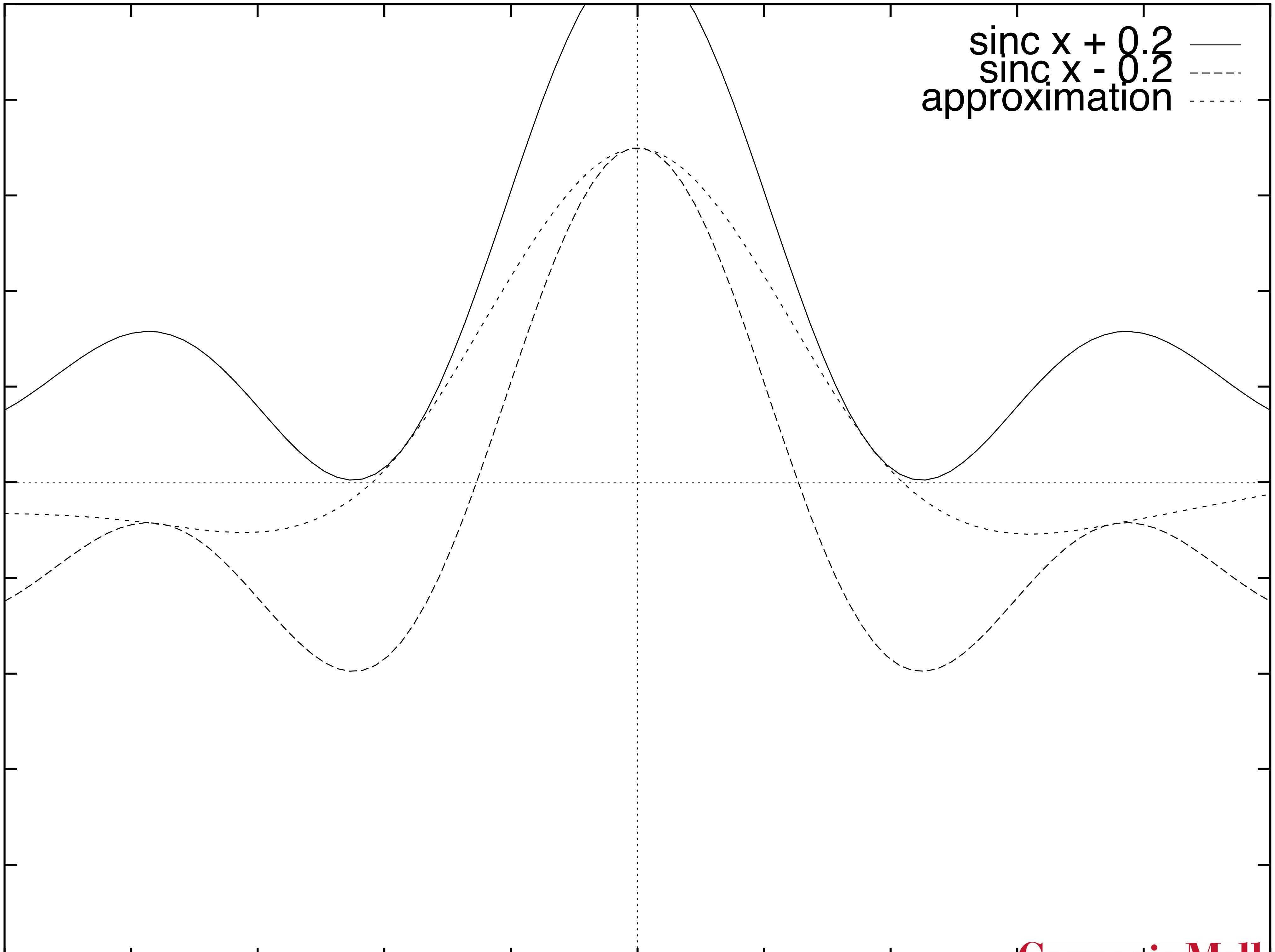
Properties

- Ignores ‘typical’ instances with small error
- Only upper or lower bound active at any time
- QP in 2n variables as cheap as SVM problem
- Robustness with respect to outliers
  - l1 loss yields same problem without epsilon
  - Huber’s robust loss yields similar problem but with added quadratic penalty on coefficients

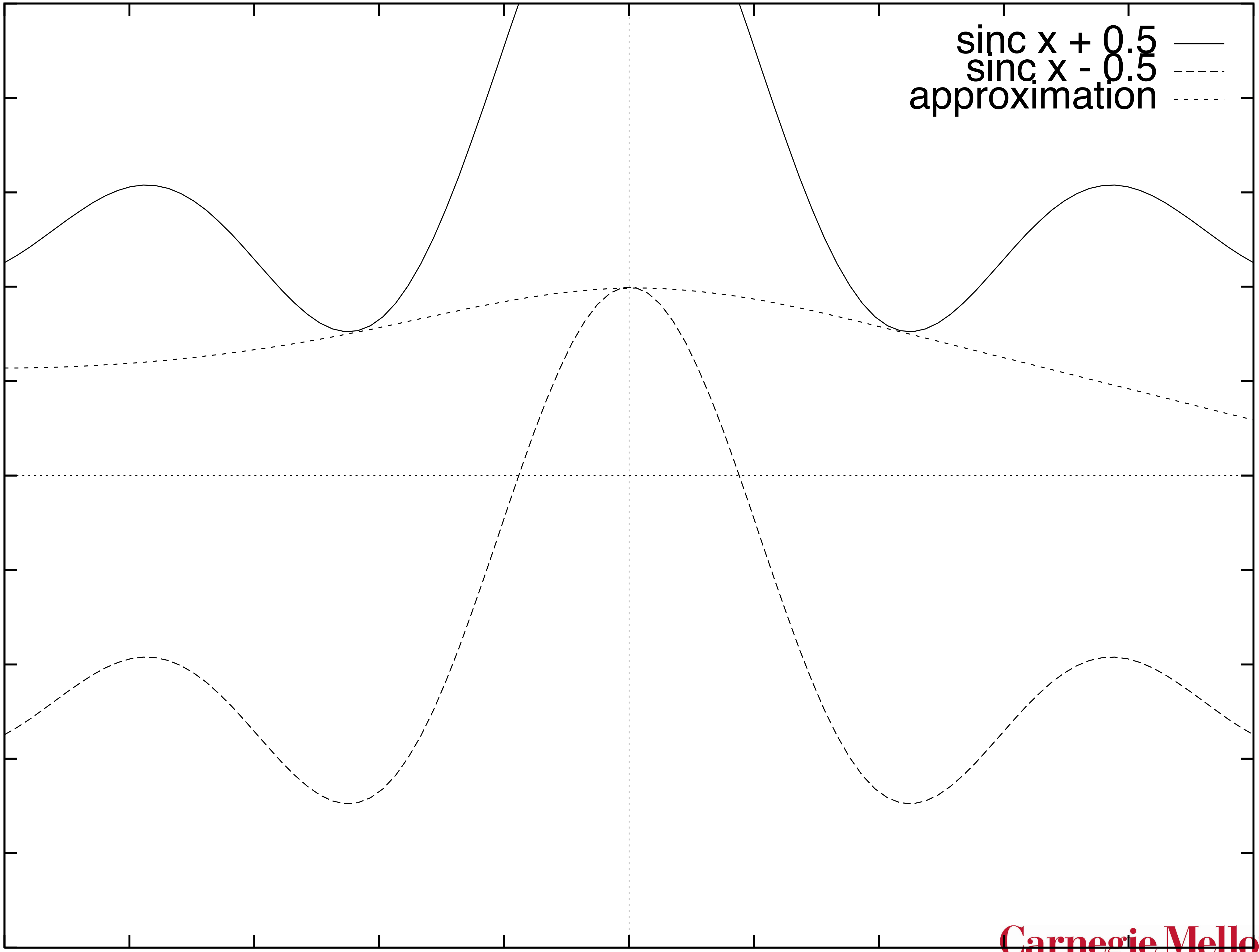
Regression example



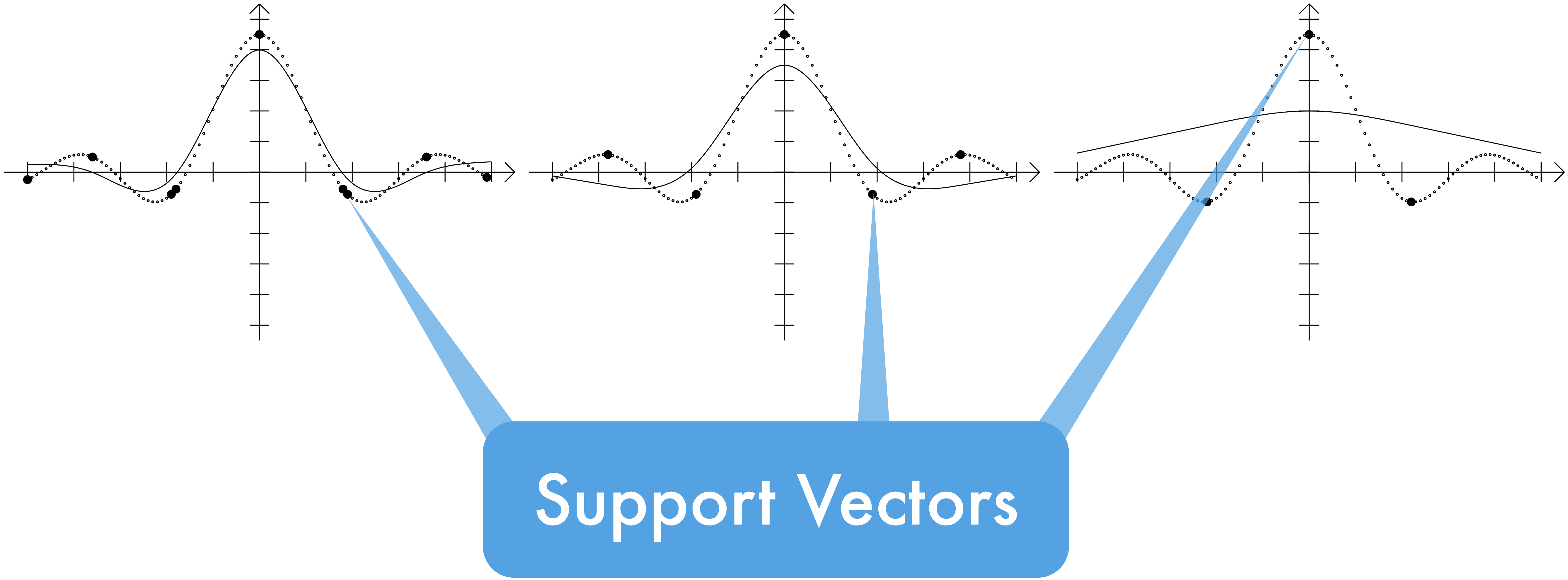
Regression example



Regression example

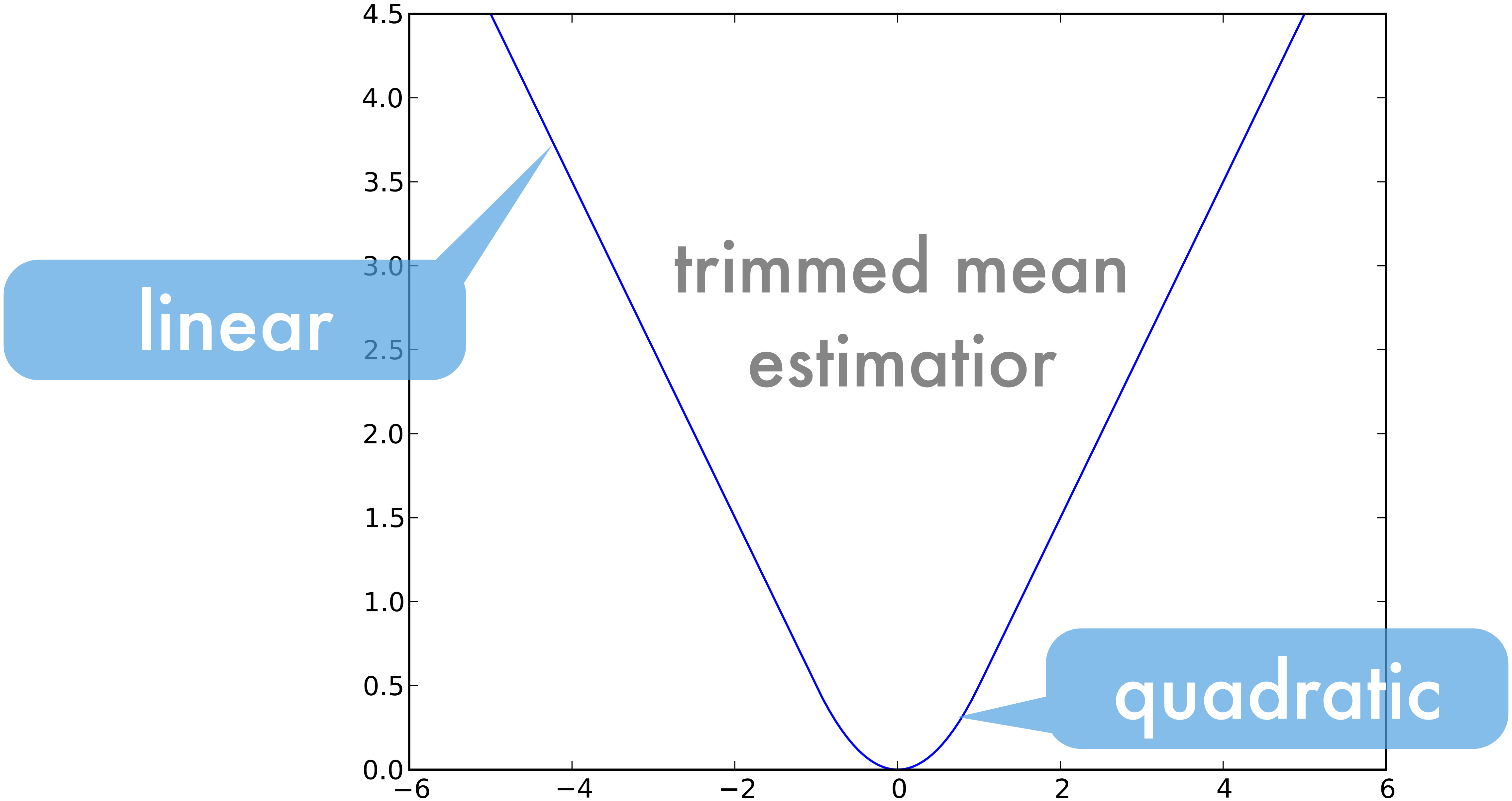


Regression example

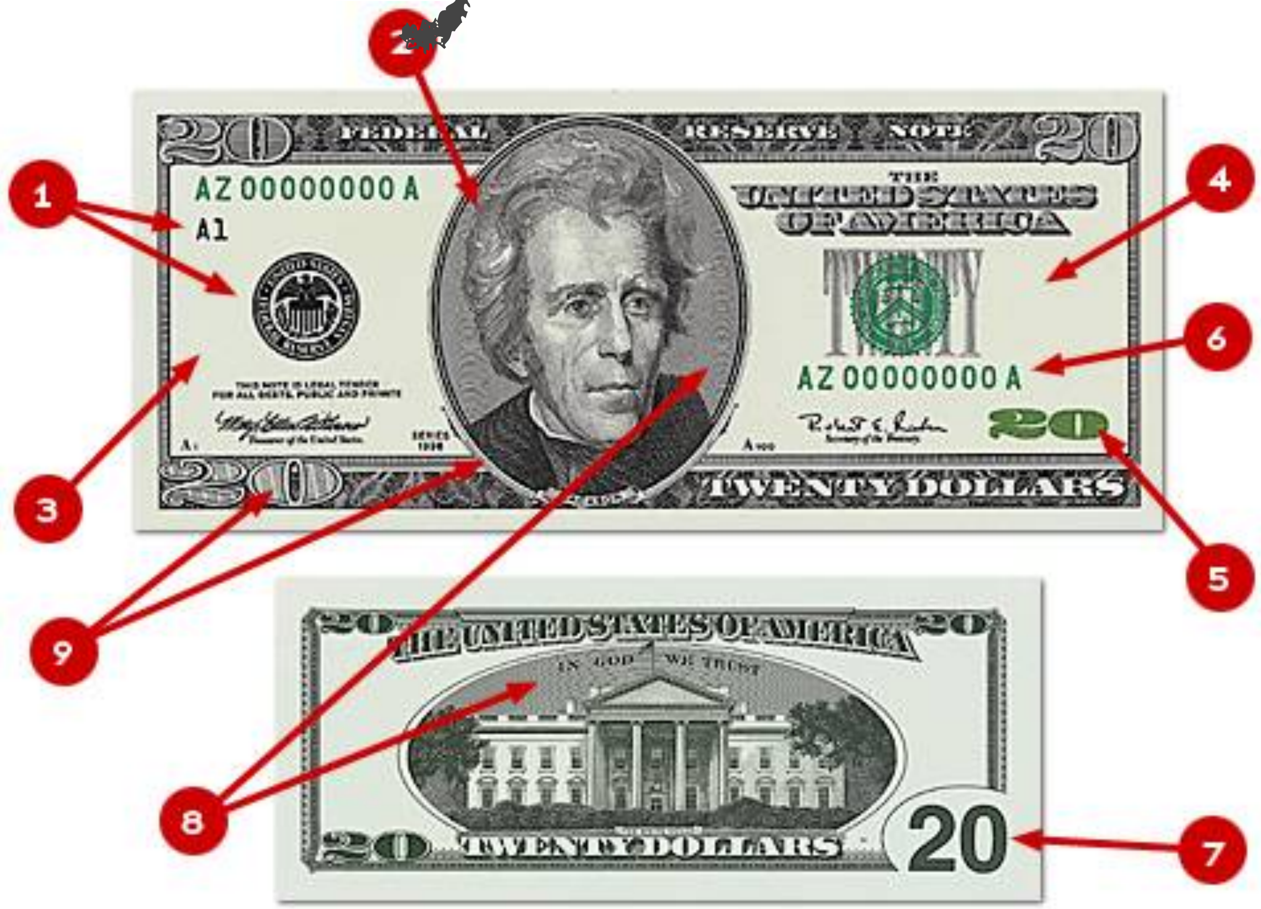


Huber's robust loss

$$l(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{if } |y - f(x)| < 1 \\ |y - f(x)| - \frac{1}{2} & \text{otherwise} \end{cases}$$



Novelty Detection



Basic Idea

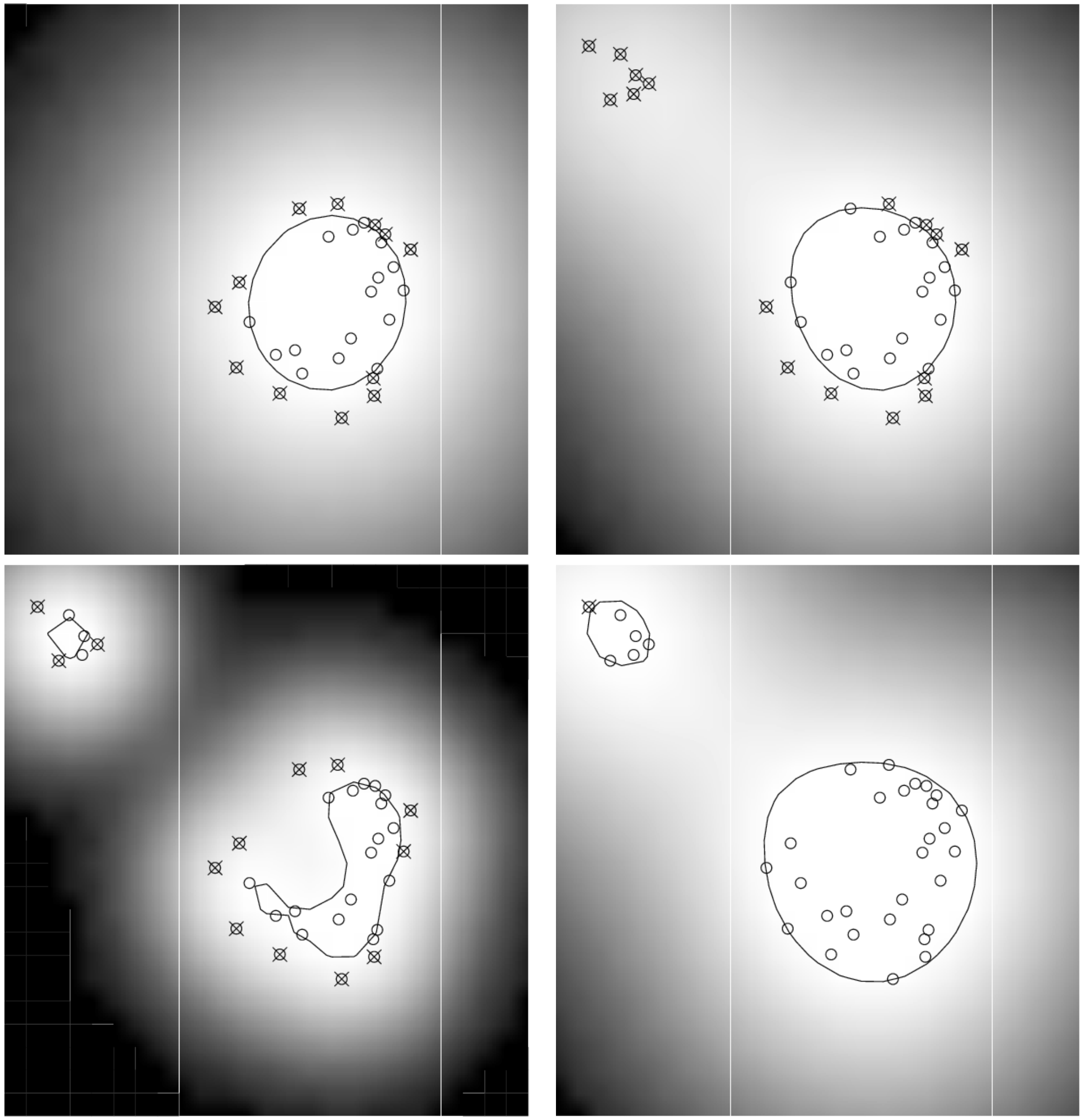
Data

Observations (x_i) generated from some $\mathrm{P}(x)$, e.g.,

- network usage patterns
- handwritten digits
- alarm sensors
- factory status

Task

Find unusual events, clean database, distinguish typical examples.



Applications

Network Intrusion Detection

Detect whether someone is trying to hack the network, downloading tons of MP3s, or doing anything else *unusual* on the network.

Jet Engine Failure Detection

You can't destroy jet engines just to see *how* they fail.

Database Cleaning

We want to find out whether someone stored bogus information in a database (typos, etc.), mislabelled digits, ugly digits, bad photographs in an electronic album.

Fraud Detection

Credit Cards, Telephone Bills, Medical Records

Self calibrating alarm devices

Car alarms (adjusts itself to where the car is parked), home alarm (furniture, temperature, windows, etc.)

Novelty Detection via Density Estimation

Key Idea

- Novel data is one that we don't see frequently.
- It must lie in low density regions.

Step 1: Estimate density

- Observations $x_1, \ldots, x_m$
- Density estimate via Parzen windows

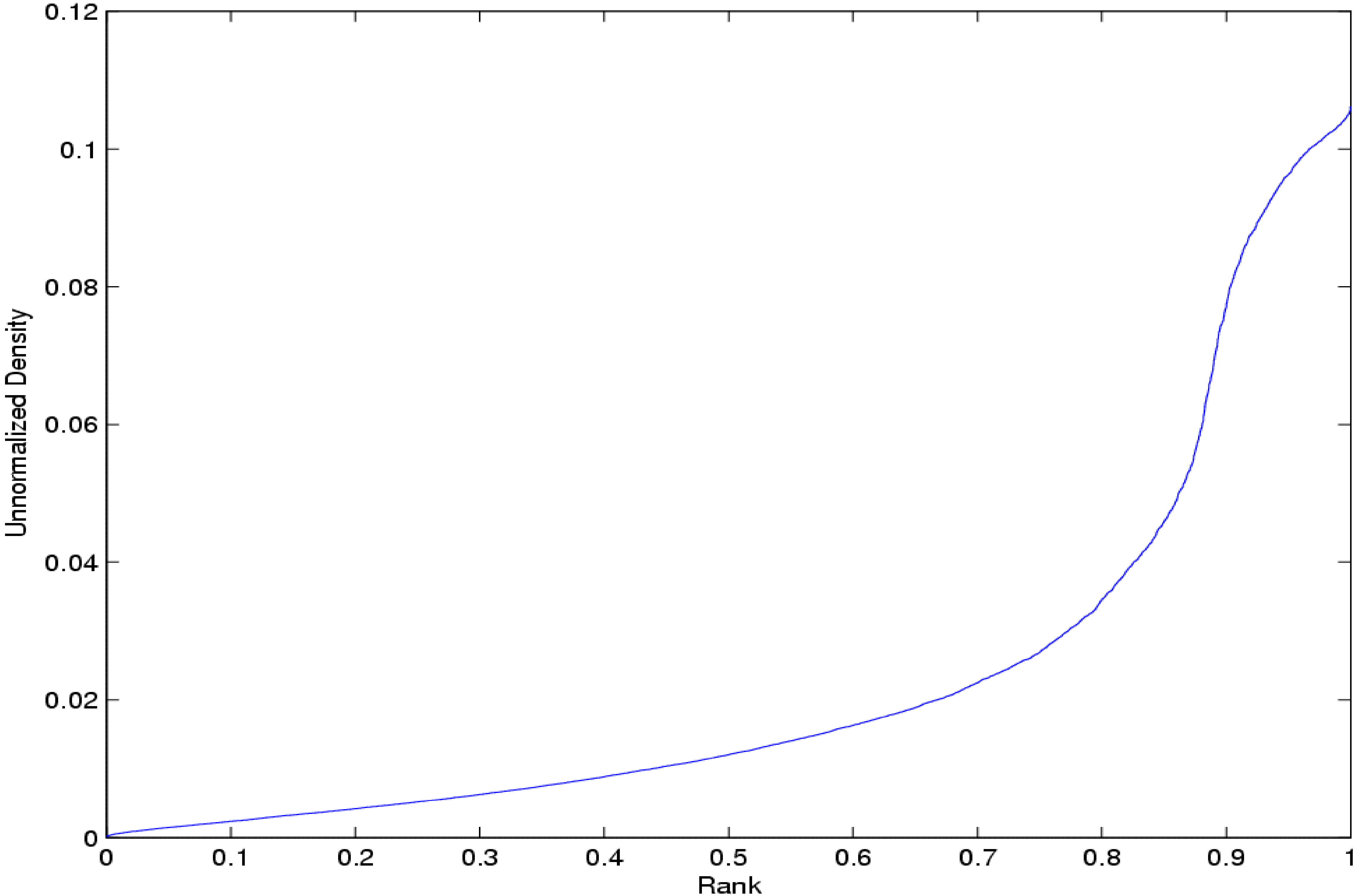
Step 2: Thresholding the density

- Sort data according to density and use it for rejection
- Practical implementation: compute

$$p(x_i) = \frac{1}{m} \sum_j k(x_i, x_j) \text{ for all } i$$

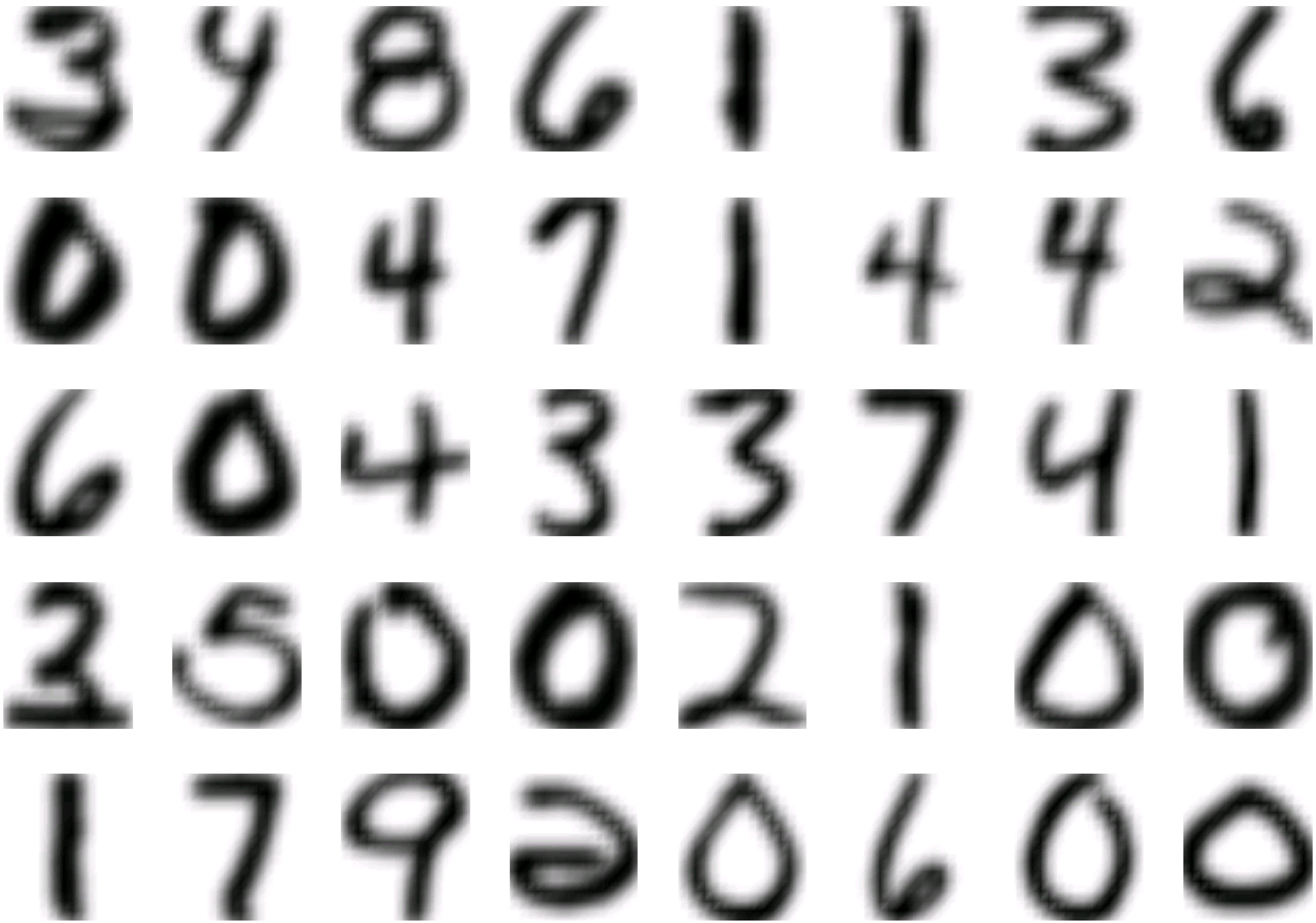
  and sort according to magnitude.
- Pick smallest $p(x_i)$ as novel points.

Order Statistics of Densities



iversity

Typical Data



Outliers

A better way

Problems

- We do not care about estimating the density properly in regions of high density (waste of capacity).
- We only care about the relative density for thresholding purposes.
- We want to eliminate a certain fraction of observations and tune our estimator specifically for this fraction.

Solution

- Areas of low density can be approximated as the **level set** of an auxiliary function. No need to estimate $p(x)$ directly — use proxy of $p(x)$.
- Specifically: find $f(x)$ such that x is novel if $f(x) \leq c$ where c is some constant, i.e. $f(x)$ describes the amount of novelty.

Problems with density estimation

- Exponential Family for density estimation

$$p(x|\theta) = \exp\left(\langle \phi(x), \theta \rangle - g(\theta)\right)$$

- MAP estimation

$$\underset{\theta}{\text{minimize}} \sum_i g(\theta) - \langle \phi(x_i), \theta \rangle + \frac{1}{2\sigma^2} \|\theta\|^2$$

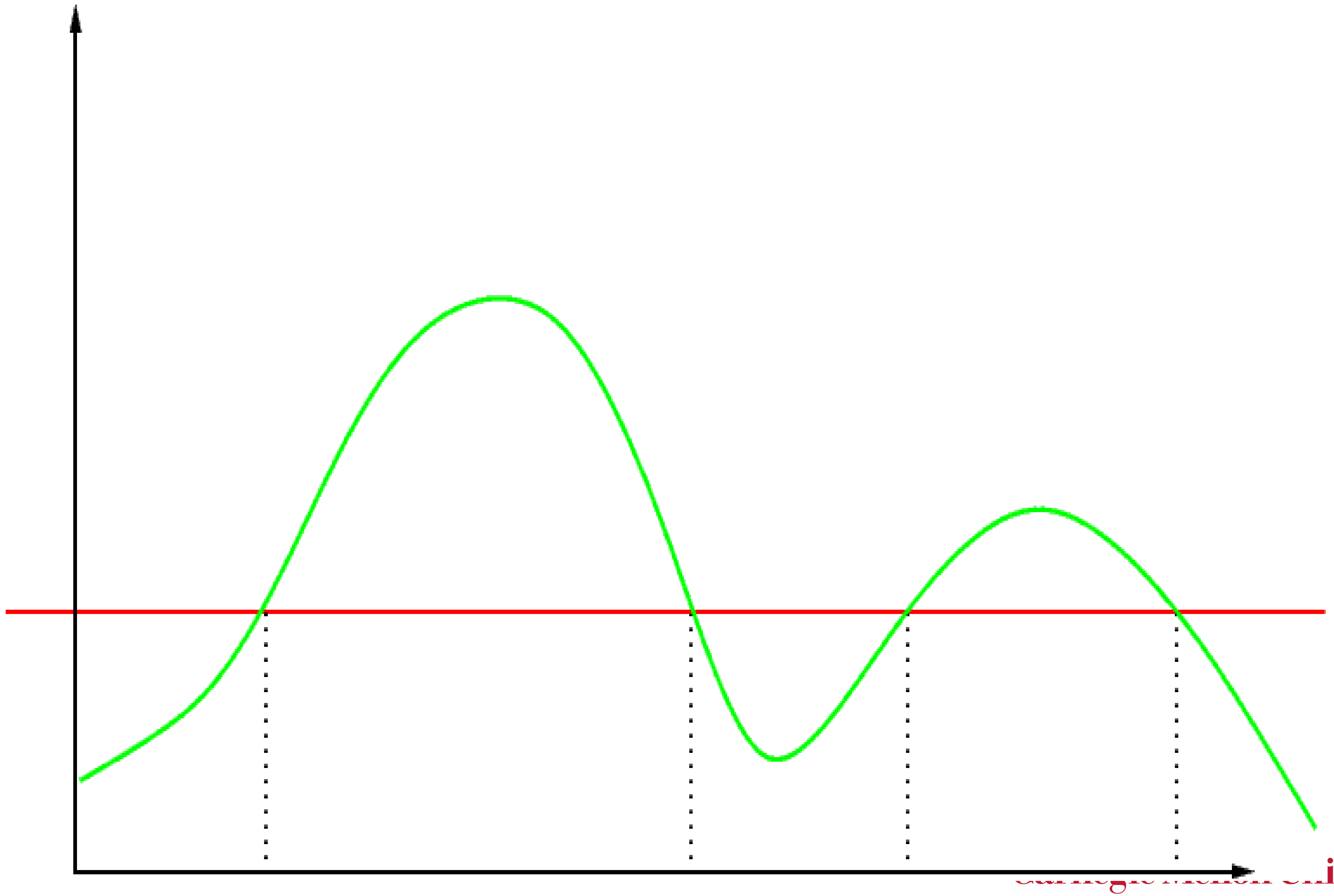
Advantages

- Convex optimization problem
- Concentration of measure

Problems

- Normalization $g(\theta)$ may be painful to compute
- For density estimation we need no normalized $p(x|\theta)$
- No need to perform particularly well in high density regions

Thresholding



Optimization Problem

Optimization Problem

MAP $$\sum_{i=1}^{m} -\log p(x_i|\theta) + \frac{1}{2\sigma^2}\|\theta\|^2$$

Novelty $$\sum_{i=1}^{m} \max\left(-\log \frac{p(x_i|\theta)}{\exp(\rho - g(\theta))}, 0\right) + \frac{1}{2}\|\theta\|^2$$

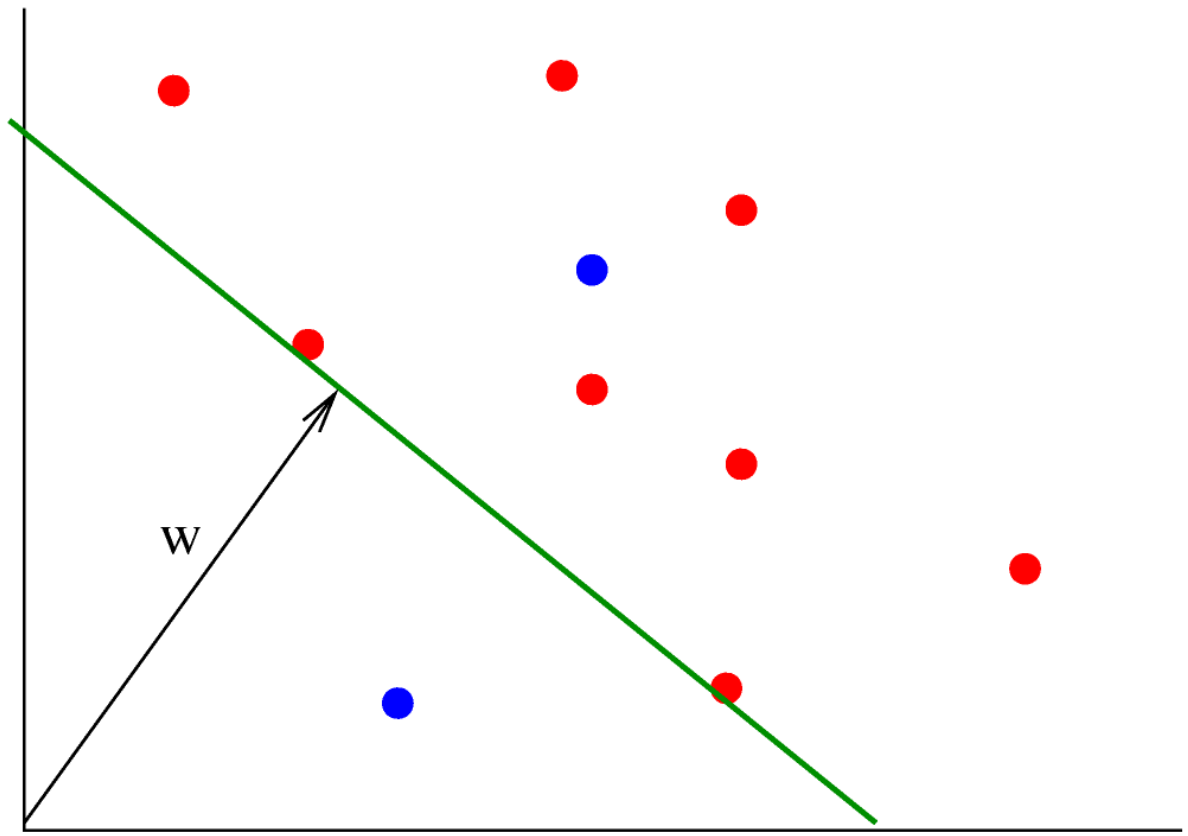
$$\sum_{i=1}^{m} \max(\rho - \langle\phi(x_i), \theta\rangle, 0) + \frac{1}{2}\|\theta\|^2$$

Advantages

- No normalization $g(\theta)$ needed
- No need to perform particularly well in high density regions (estimator focuses on low-density regions)
- Quadratic program

Maximum Distance Hyperplane

Idea Find hyperplane, given by $f(x) = \langle w, x \rangle + b = 0$ that has **maximum distance from origin** yet is still closer to the origin than the observations.



Hard Margin

$$\text{minimize} \quad \frac{1}{2}\|w\|^2$$

$$\text{subject to} \quad \langle w, x_i \rangle \geq 1$$

Soft Margin

$$\text{minimize} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi_i$$

$$\text{subject to} \quad \langle w, x_i \rangle \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

Optimization Problem

Primal Problem

minimize $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i$

subject to $\langle w, x_i\rangle - 1 + \xi_i \geq 0$ and $\xi_i \geq 0$

Lagrange Function L

- Subtract constraints, multiplied by Lagrange multipliers (α_i and η_i), from Primal Objective Function.
- Lagrange function L has **saddlepoint** at optimum.

$$L = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i\left(\langle w, x_i\rangle - 1 + \xi_i\right) - \sum_{i=1}^{m}\eta_i\xi_i$$

subject to $\alpha_i, \eta_i \geq 0$.

Dual Problem

Optimality Conditions

$$\partial_w L = w - \sum_{i=1}^m \alpha_i x_i = 0 \implies w = \sum_{i=1}^m \alpha_i x_i$$
$$\partial_{\xi_i} L = C - \alpha_i - \eta_i = 0 \implies \alpha_i \in [0, C]$$

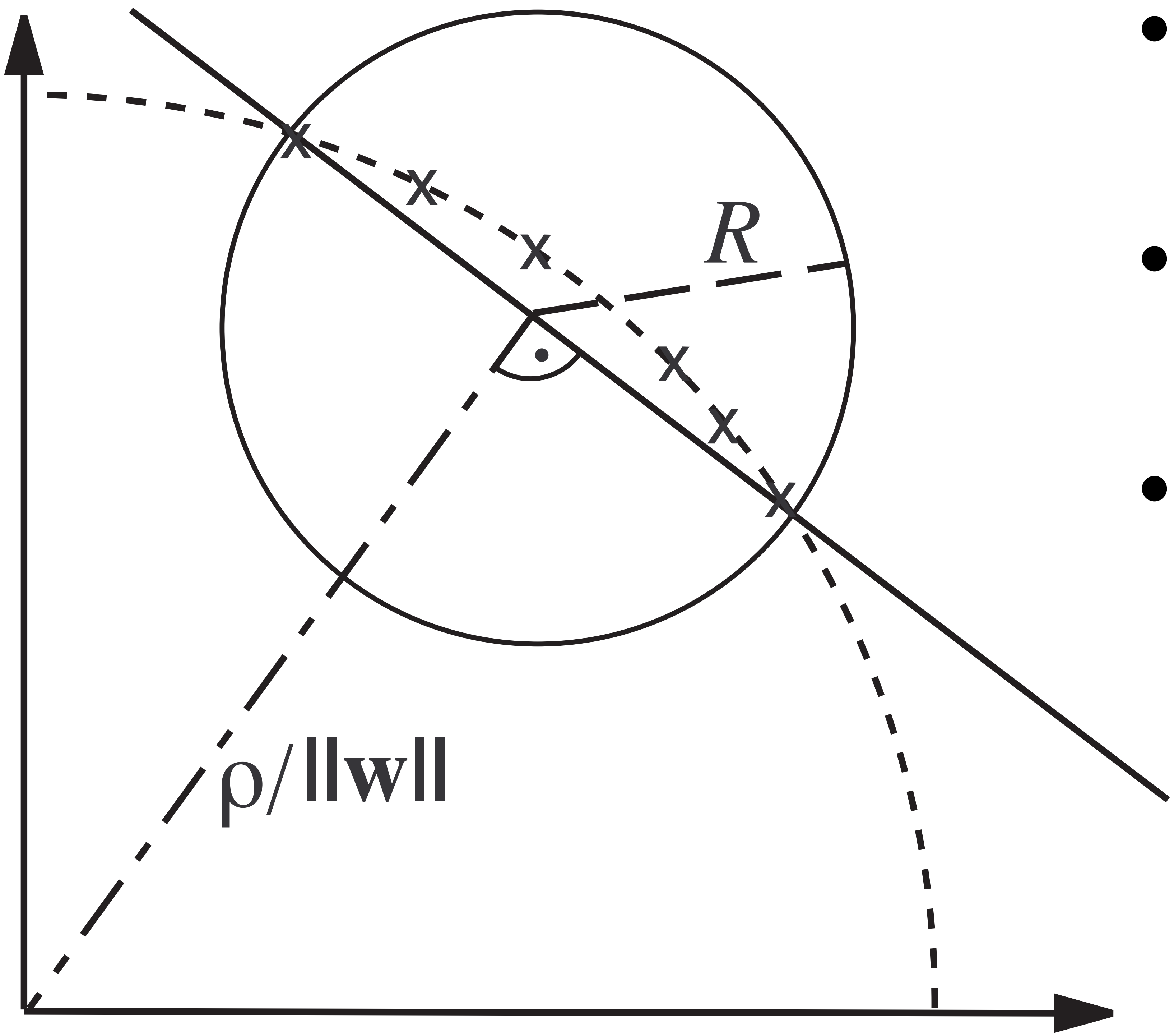
Now **substitute** the optimality conditions **back into** L.

Dual Problem

$$\text{minimize} \quad \frac{1}{2}\sum_{i=1}^m \alpha_i \alpha_j \langle x_i, x_j \rangle - \sum_{i=1}^m \alpha_i$$
$$\text{subject to} \quad \alpha_i \in [0, C]$$

All this is only possible due to the convexity of the primal problem.

Minimum enclosing ball



- Observations on surface of ball
- Find minimum enclosing ball
- Equivalent to single class SVM

Adaptive thresholds

Problem

- Depending on C, the number of novel points will vary.
- We would like to **specify the fraction** ν beforehand.

Solution

Use hyperplane separating data from the origin

$$H := \{x | \langle w, x \rangle = \rho\}$$

where the threshold ρ is **adaptive**.

Intuition

- Let the hyperplane shift by shifting ρ
- Adjust it such that the 'right' number of observations is considered novel.
- Do this automatically

Optimization Problem

Primal Problem

$$\text{minimize } \frac{1}{2}\|w\|^2 + \sum_{i=1}^{m} \xi_i - m\nu\rho$$

$$\text{where } \langle w, x_i \rangle - \rho + \xi_i \geq 0$$

$$\xi_i \geq 0$$

Dual Problem

$$\text{minimize } \frac{1}{2}\sum_{i=1}^{m} \alpha_i \alpha_j \langle x_i, x_j \rangle$$

$$\text{where } \alpha_i \in [0, 1] \text{ and } \sum_{i=1}^{m} \alpha_i = \nu m.$$

The ν-property theorem

- Optimization problem

$$\underset{w}{\text{minimize}} \;\; \frac{1}{2}\|w\|^2 + \sum_{i=1}^{m} \xi_i - m\nu\rho$$

$$\text{subject to } \langle w, x_i \rangle \geq \rho - \xi_i \text{ and } \xi_i \geq 0$$

- Solution satisfies
  - At most a fraction of ν points are novel
  - At most a fraction of (1-ν) points aren't novel
  - Fraction of points on boundary vanishes for large m (for non-pathological kernels)

Proof

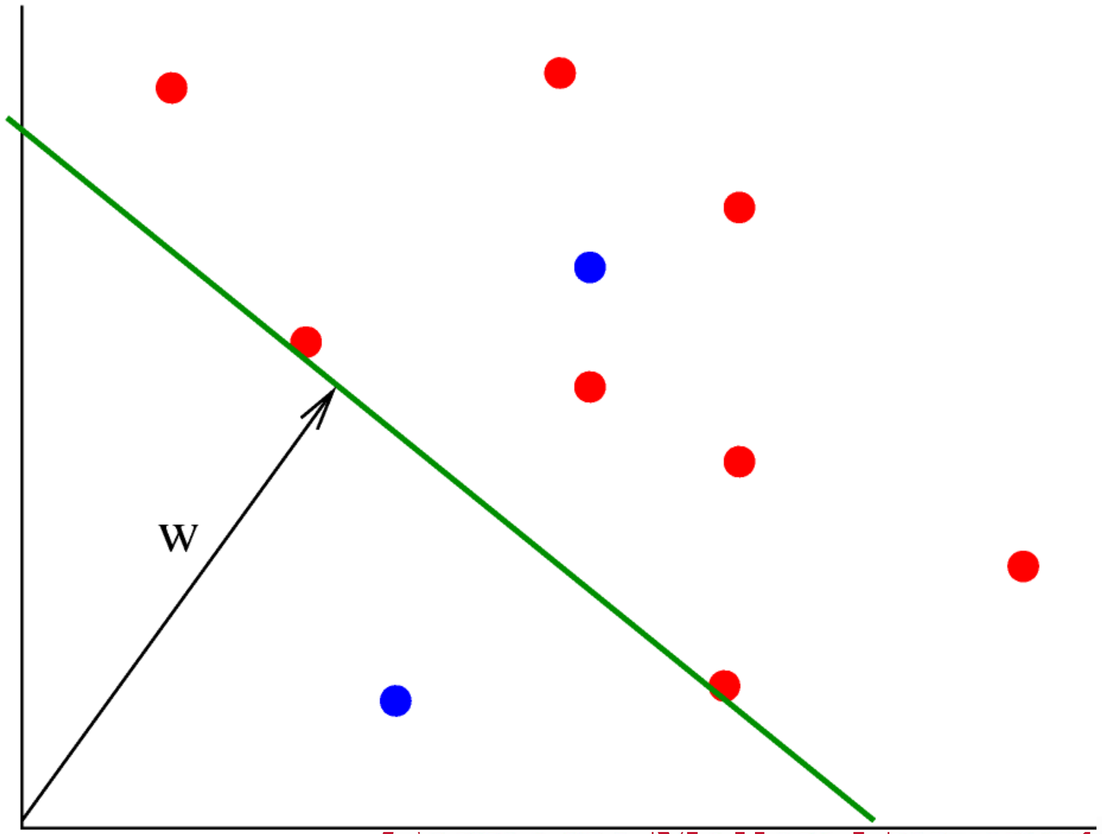
- Move boundary at optimality
  - For smaller threshold m- points on wrong side of margin contribute $\delta(m_- - \nu m) \leq 0$
  - For larger threshold m+ points not on 'good' side of margin yield

$$\delta(m_+ - \nu m) \geq 0$$

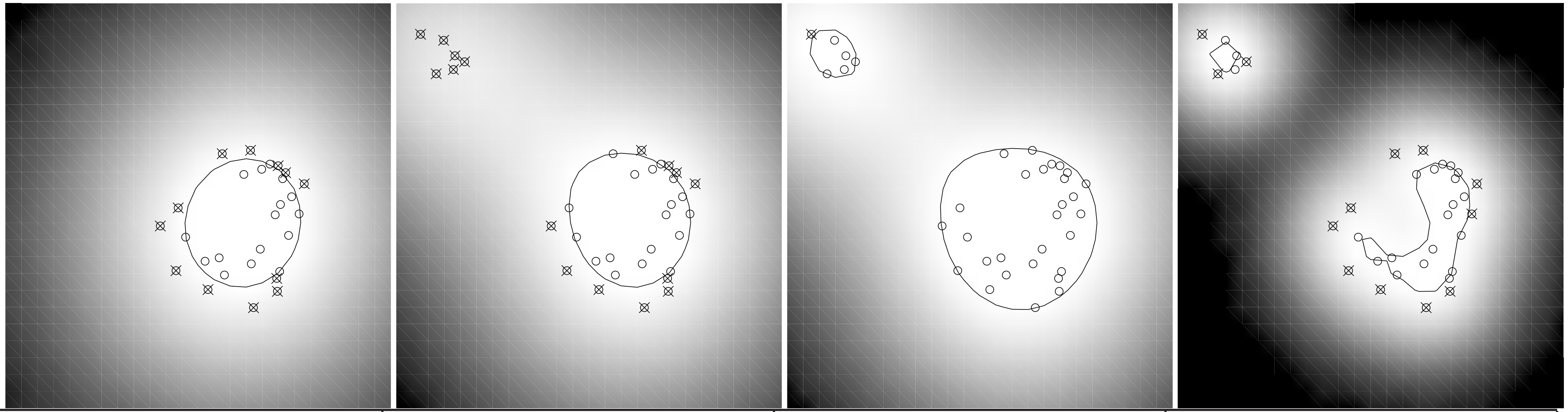
  - Combining inequalities

$$\frac{m_-}{m} \leq \nu \leq \frac{m_+}{m}$$

- Margin set of measure 0



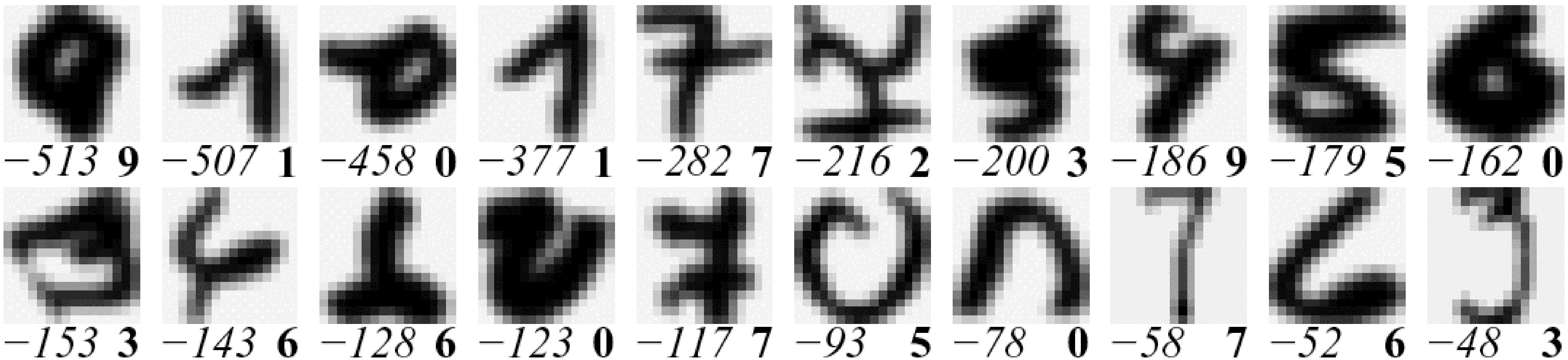
Toy example



ν, width c	0.5, 0.5	0.5, 0.5	0.1, 0.5	0.5, 0.1
frac. SVs/OLs	0.54, 0.43	0.59, 0.47	0.24, 0.03	0.65, 0.38
margin $\rho/\|\mathbf{w}\|$	0.84	0.70	0.62	0.48

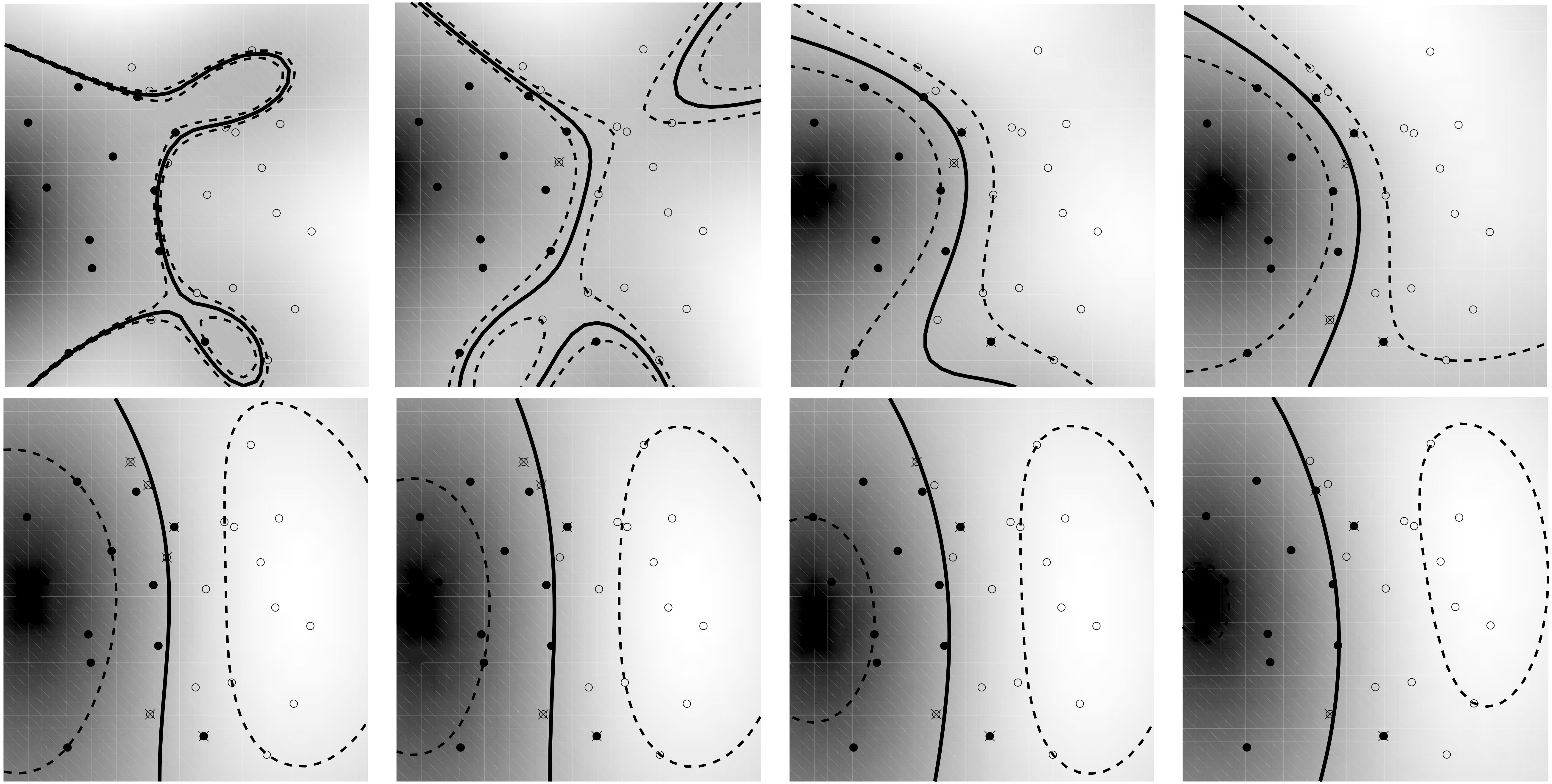
threshold and smoothness requirements

Novelty detection for OCR



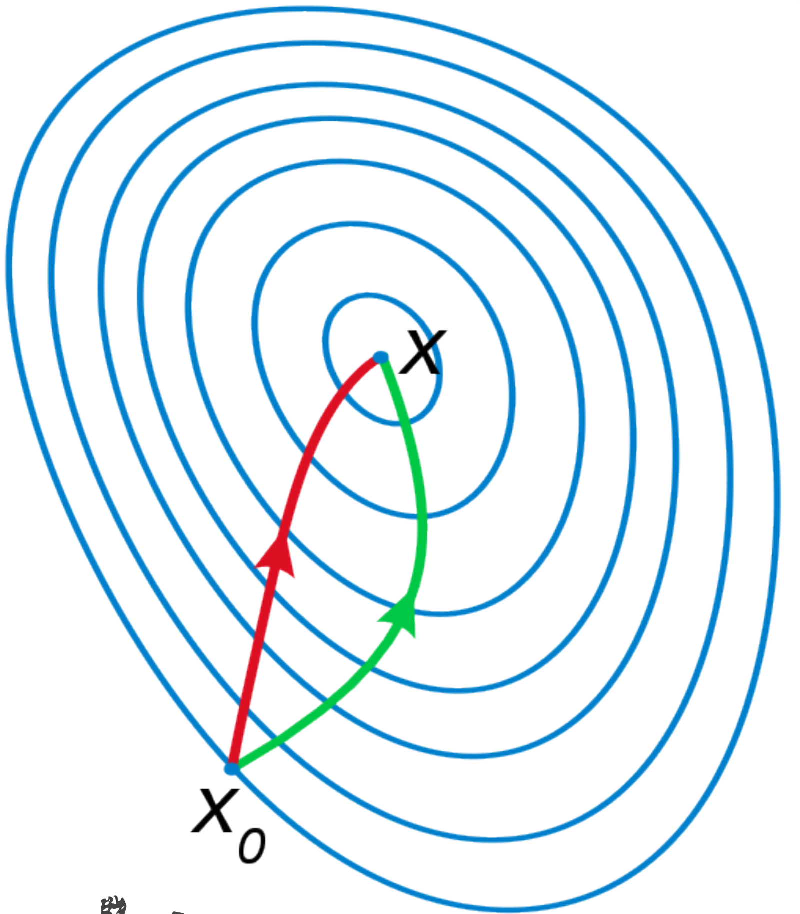
- Better estimates since we only optimize in low density regions.
- Specifically tuned for small number of outliers.
- Only estimates of a level-set.
- For $\nu = 1$ we get the Parzen-windows estimator back.

Classification with the ν-trick



changing kernel width and threshold

Convex Optimization



Carnegie Mellon

Selecting Variables

Constrained Quadratic Program

- Optimization Problem

$$\underset{\alpha}{\text{minimize}}\ \frac{1}{2}\alpha^\top Q\alpha + l^\top \alpha \text{ subject to } C\alpha + b \leq 0$$

  - Support Vector classification
  - Support Vector regression
  - Novelty detection
- Solving it
  - Off the shelf solvers for small problems
  - Solve sequence of subproblems
  - Optimization in primal space (the w space)

Subproblems

- Original optimization problem

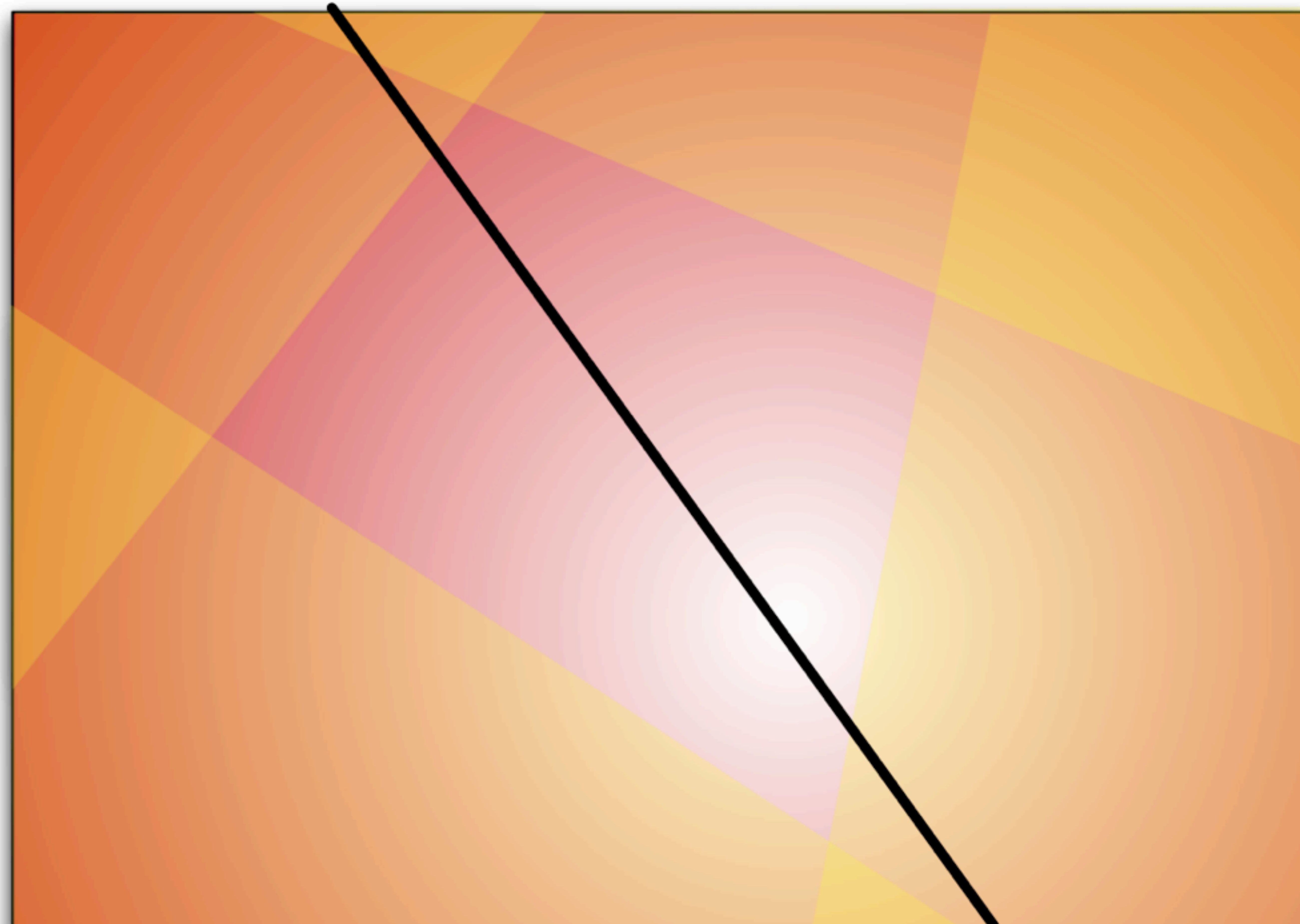
$$\underset{\alpha}{\text{minimize}} \frac{1}{2}\alpha^\top Q\alpha + l^\top\alpha \text{ subject to } C\alpha + b \leq 0$$

- Key Idea - solve subproblems one at a time and decompose into active and fixed set $\alpha = (\alpha_a, \alpha_f)$

$$\underset{\alpha}{\text{minimize}} \; \frac{1}{2}\alpha_a^\top Q_{aa}\alpha_a + [l_a + Q_{af}\alpha_f]^\top \alpha_a$$

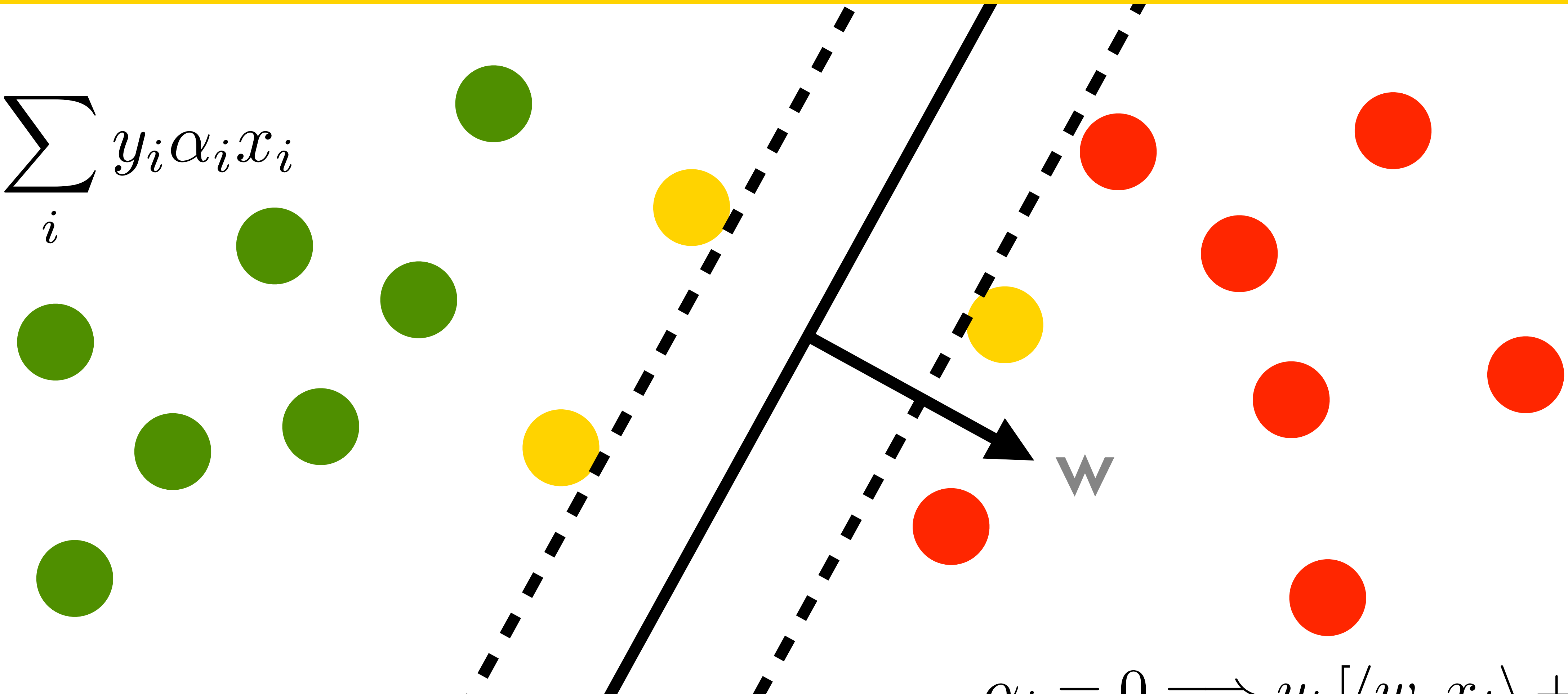
$$\text{subject to } C_a\alpha_a + [b + C_f\alpha_f] \leq 0$$

  - Subproblem is again a convex problem
  - Updating subproblems is cheap



Picking observations

$$w = \sum_i y_i \alpha_i x_i$$



$$\alpha_i \left[y_i \left[\langle w, x_i \rangle + b \right] + \xi_i - 1 \right] = 0$$

$$\eta_i \xi_i = 0$$

$$\alpha_i = 0 \Longrightarrow y_i \left[\langle w, x_i \rangle + b \right] \geq 1$$

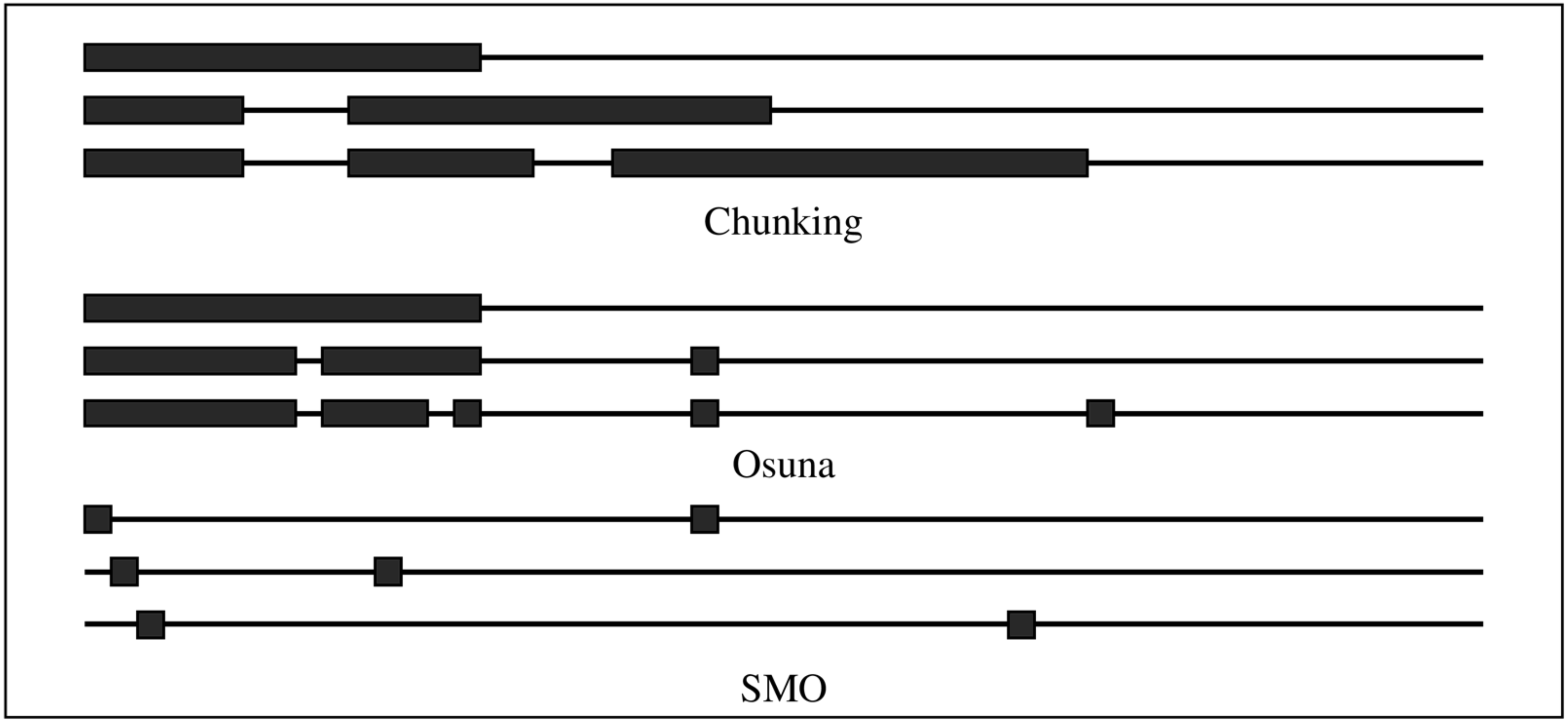
$$0 < \alpha_i < C \Longrightarrow y_i \left[\langle w, x_i \rangle + b \right] = 1$$

$$\alpha_i = C \Longrightarrow y_i \left[\langle w, x_i \rangle + b \right] \leq 1$$

- Most violated margin condition
- Points on the boundary
- Points with nonzero Lagrange multiplier that are correct

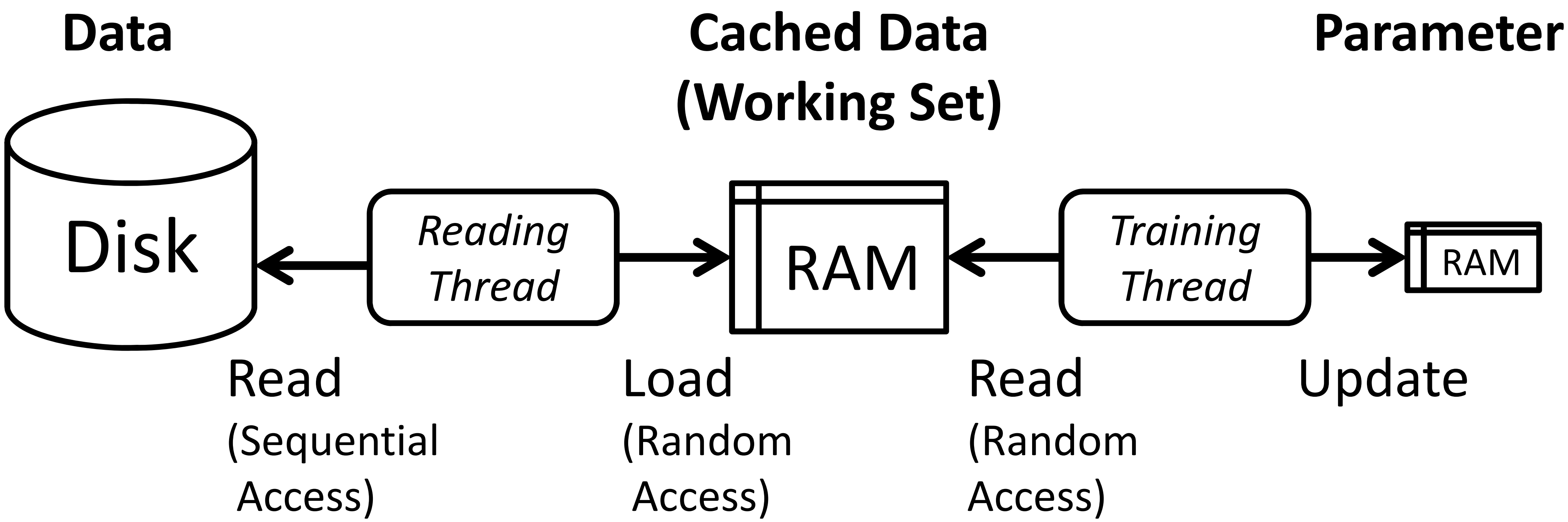
Selecting variables

- Incrementally increase (chunking)
- Select promising subset of actives (SVMLight)
- Select pairs of variables (SMO)



Being smart about hardware

- Data flow from disk to CPU

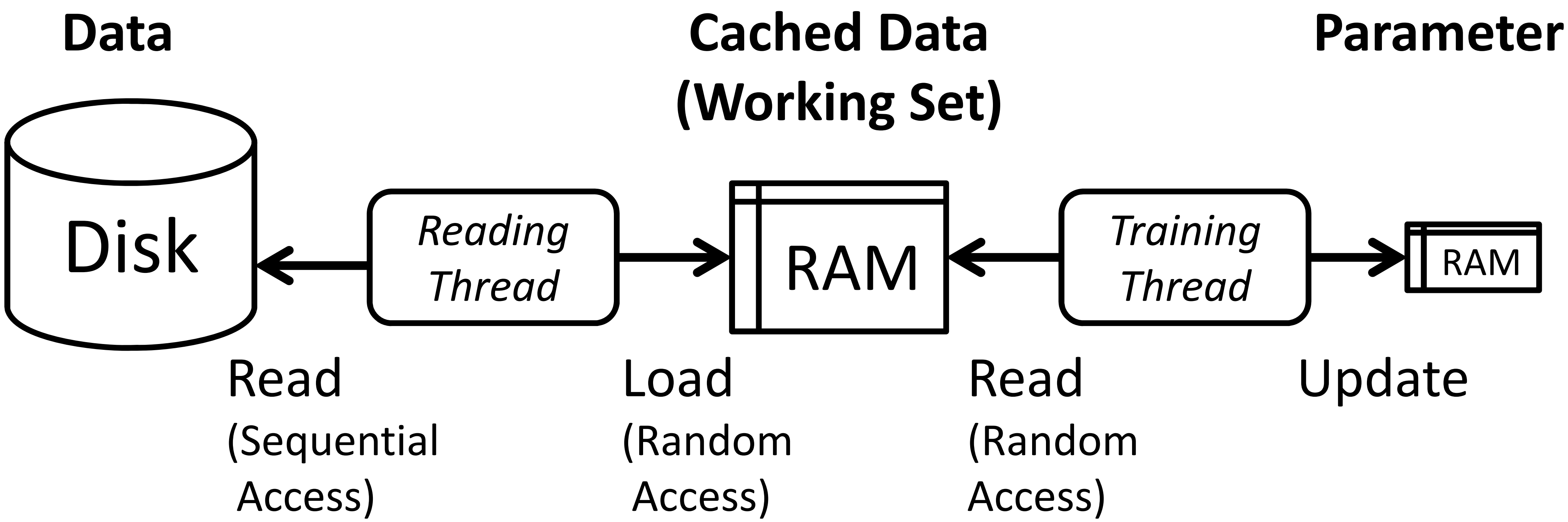


- IO speeds

System	Capacity	Bandwidth	IOPs
Disk	3TB	150MB/s	10^2
SSD	256GB	500MB/s	$5 \cdot 10^4$
RAM	16GB	30GB/s	10^8
Cache	16MB	100GB/s	10^9

Being smart about hardware

- Data flow from disk to CPU

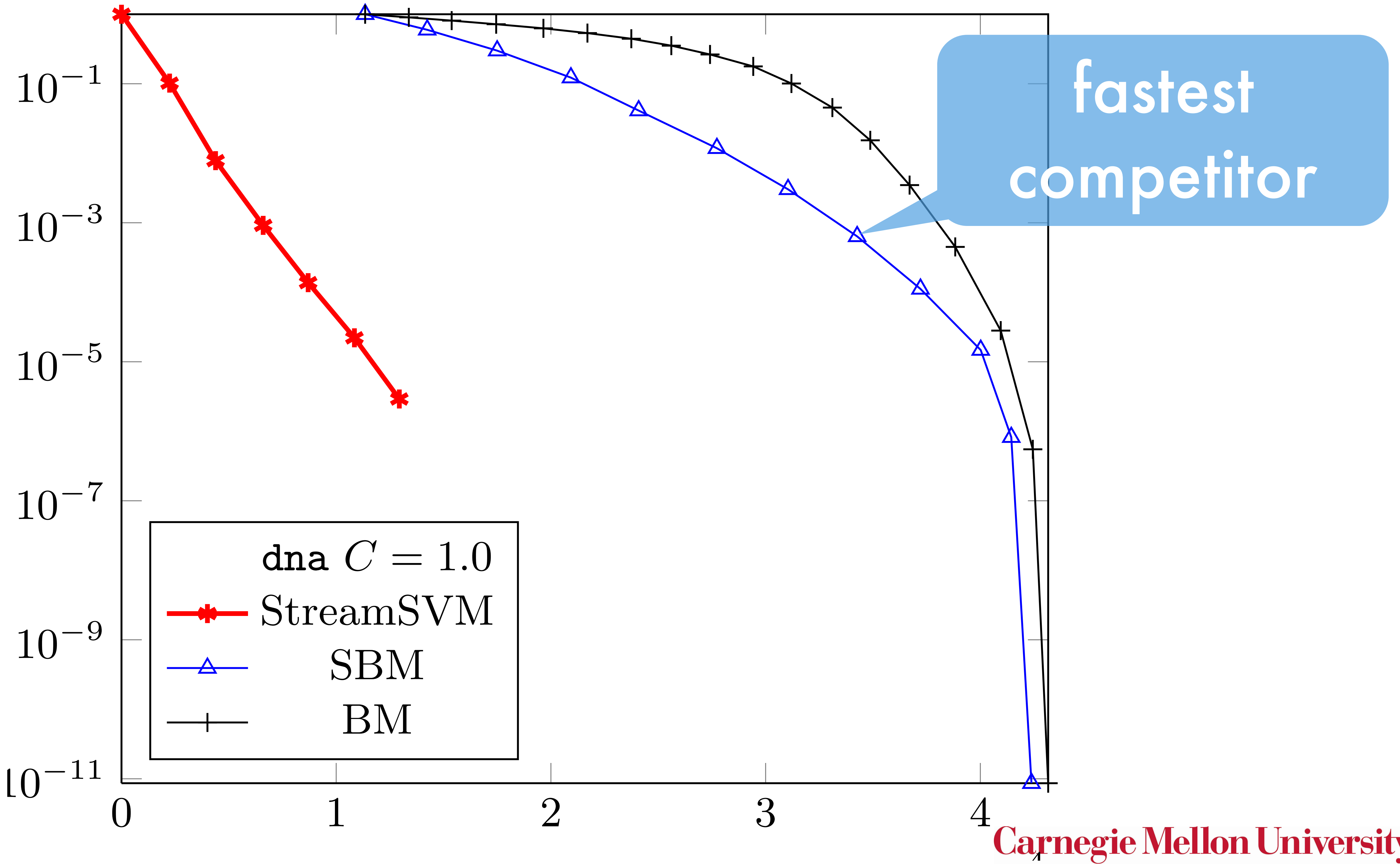


- IO speeds

reuse data

System	Capacity	Bandwidth	IOPs
Disk	3TB	150MB/s	10^2
SSD	256GB	500MB/s	$5 \cdot 10^4$
RAM	16GB	30GB/s	10^8
Cache	16MB	100GB/s	10^9

Runtime Example (Matsushima, Vishwanathan, Smola, 2012)



Regularization

Problems with Kernels

Myth

Support Vectors work because they map data into a high-dimensional feature space.

And your statistician (Bellmann) told you . . .

The higher the dimensionality, the more data you need

Example: Density Estimation

Assuming data in $[0, 1]^m$, 1000 observations in $[0, 1]$ give you on average 100 instances per bin (using binsize 0.1^m) but only $\frac{1}{100}$ instances in $[0, 1]^5$.

Worrying Fact

Some kernels map into an **infinite**-dimensional space, e.g., $k(x, x') = \exp(-\frac{1}{2\sigma^2}\|x - x'\|^2)$

Encouraging Fact

SVMs work well in practice . . .

Solving the Mystery

The Truth is in the Margins

Maybe the maximum margin requirement is what saves us when finding a classifier, i.e., we minimize $\|w\|^2$.

Risk Functional

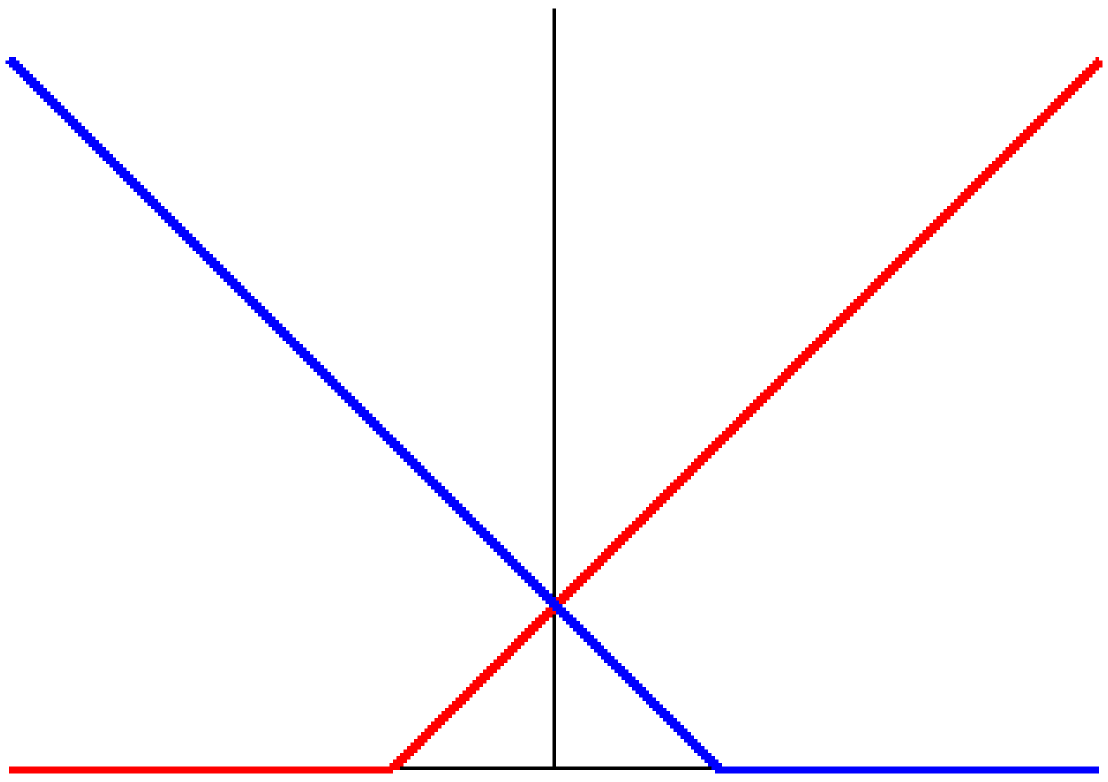
Rewrite the optimization problems in a unified form

$$R_{\text{reg}}[f] = \sum_{i=1}^{m} c(x_i, y_i, f(x_i)) + \Omega[f]$$

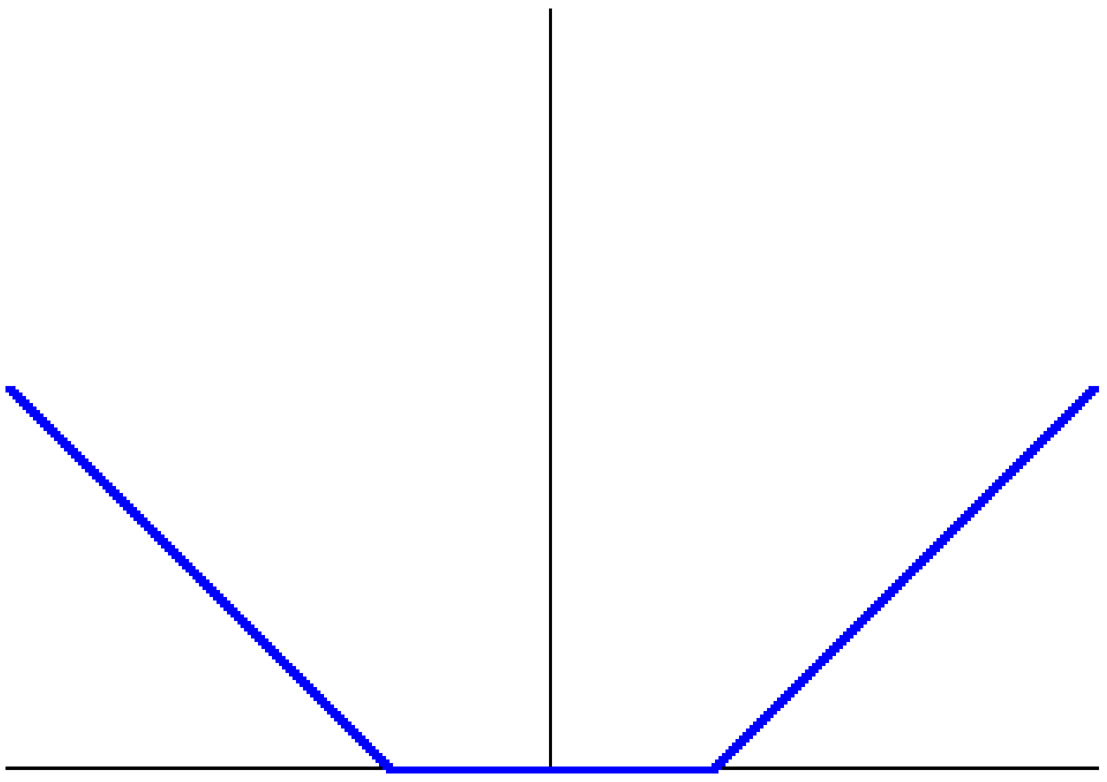
$c(x, y, f(x))$ is a loss function and $\Omega[f]$ is a regularizer.

- $\Omega[f] = \frac{\lambda}{2}\|w\|^2$ for linear functions.
- For classification $c(x, y, f(x)) = \max(0, 1 - yf(x))$.
- For regression $c(x, y, f(x)) = \max(0, |y - f(x)| - \epsilon)$.

Typical SVM loss



Soft Margin Loss



ε-insensitive Loss

Soft Margin Loss

Original Optimization Problem

$$\underset{w,\xi}{\text{minimize}} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i$$

subject to $y_i f(x_i) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all $1 \leq i \leq m$

Regularization Functional

$$\underset{w}{\text{minimize}} \quad \frac{\lambda}{2}\|w\|^2 + \sum_{i=1}^{m}\max(0, 1 - y_i f(x_i))$$

- For fixed f, clearly $\xi_i \geq \max(0, 1 - y_i f(x_i))$.
- For $\xi > \max(0, 1 - y_i f(x_i))$ we can decrease it such that the bound is matched and improve the objective function.
- Both methods are equivalent.

Why Regularization?

What we really wanted ...

Find some $f(x)$ such that the expected loss $\mathbf{E}[c(x, y, f(x))]$ is small.

What we ended up doing ...

Find some $f(x)$ such that the empirical average of the expected loss $\mathbf{E}_{\text{emp}}[c(x, y, f(x))]$ is small.

$$\mathbf{E}_{\text{emp}}[c(x, y, f(x))] = \frac{1}{m} \sum_{i=1}^{m} c(x_i, y_i, f(x_i))$$

However, just minimizing the empirical average does not guarantee anything for the expected loss (overfitting).

Safeguard against overfitting

We need to constrain the class of functions $f \in \mathfrak{F}$ somehow. Adding $\Omega[f]$ as a penalty does exactly that.

Some regularization ideas

Small Derivatives

We want to have a function f which is smooth on the entire domain. In this case we could use

$$\Omega[f] = \int_X \|\partial_x f(x)\|^2 dx = \langle \partial_x f, \partial_x f \rangle.$$

Small Function Values

If we have no further knowledge about the domain X, minimizing $\|f\|^2$ might be sensible, i.e.,

$$\Omega[f] = \|f\|^2 = \langle f, f \rangle.$$

Splines

Here we want to find f such that both $\|f\|^2$ and $\|\partial_x^2 f\|^2$ are small. Hence we can minimize

$$\Omega[f] = \|f\|^2 + \|\partial_x^2 f\|^2 = \langle (f, \partial_x^2 f), (f, \partial_x^2 f) \rangle$$

Regularization

Regularization Operators

We map f into some Pf, which is small for desirable f and large otherwise, and minimize

$$\Omega[f] = \|Pf\|^2 = \langle Pf, Pf \rangle.$$

For all previous examples we can find such a P.

Function Expansion for Regularization Operator

Using a linear function expansion of f in terms of some f_i, that is for $f(x) = \sum_i \alpha_i f_i(x)$ we can compute

$$\Omega[f] = \left\langle P \sum_i \alpha_i f_i(x), P \sum_j \alpha_j f_i(x) \right\rangle = \sum_{i,j} \alpha_i \alpha_j \langle Pf_i, Pf_j \rangle.$$

Regularization and Kernels

Regularization for $\Omega[f] = \frac{1}{2}\|w\|^2$

$$w = \sum_i \alpha_i \Phi(x_i) \Longrightarrow \|w\|^2 = \sum_{i,j} \alpha_i \alpha_j k(x_i, x_j)$$

This looks very similar to $\langle P f_i, P f_j \rangle$.

Key Idea

So if we could find a P and k such that

$$k(x, x') = \langle P k(x, \cdot), P k(x', \cdot) \rangle$$

we could show that using a kernel means that we are minimizing the empirical risk plus a regularization term.

Solution: Greens Functions

A sufficient condition is that k is the Greens Function of P^*P, that is $\langle P^* P k(x, \cdot), f(\cdot) \rangle = f(x)$.

One can show that this is necessary and sufficient.

Building Kernels

Kernels from Regularization Operators:

Given an operator P^*P, we can find k by solving the self consistency equation

$$\langle Pk(x,\cdot), Pk(x',\cdot)\rangle = k^\top(x,\cdot)(P^*P)k(x',\cdot) = k(x,x')$$

and take f to be the span of all $k(x,\cdot)$.

So we can find k for a given measure of smoothness.

Regularization Operators from Kernels:

Given a kernel k, we can find some P^*P for which the self consistency equation is satisfied.

So we can find a measure of smoothness for a given k.

Spectrum and Kernels

Effective Function Class

Keeping $\Omega[f]$ small means that $f(x)$ cannot take on arbitrary function values. Hence we study the function class

$$\mathfrak{F}_C = \left\{ f \,\middle|\, \frac{1}{2}\langle Pf, Pf \rangle \leq C \right\}$$

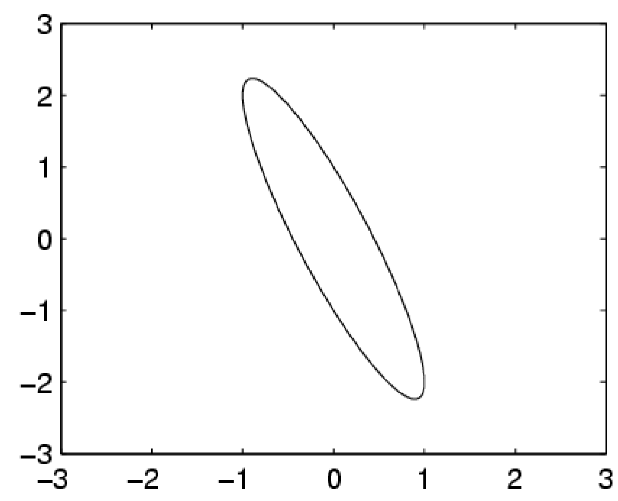
Example

For $f = \sum_i \alpha_i k(x_i, x)$ this implies $\frac{1}{2}\alpha^\top K \alpha \leq C$.

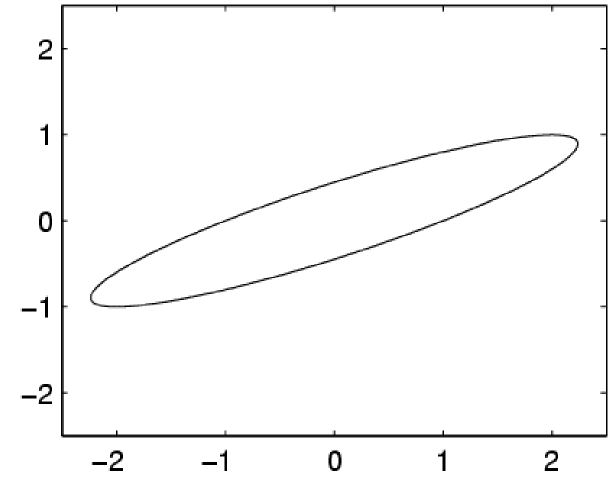
Kernel Matrix

$$K = \begin{bmatrix} 5 & 2 \\ 2 & 1 \end{bmatrix}$$

Coefficients



Function Values



Fourier Regularization

Goal

Find measure of smoothness that depends on the frequency properties of f and not on the position of f.

A Hint: Rewriting $\|f\|^2 + \|\partial_x f\|^2$

Notation: $\tilde{f}(\omega)$ is the Fourier transform of f.

$$\begin{aligned} \|f\|^2 + \|\partial_x f\|^2 &= \int |f(x)|^2 + |\partial_x f(x)|^2 dx \\ &= \int |\tilde{f}(\omega)|^2 + \omega^2 |\tilde{f}(\omega)|^2 d\omega \\ &= \int \frac{|\tilde{f}(\omega)|^2}{p(\omega)} d\omega \text{ where } p(\omega) = \frac{1}{1+\omega^2}. \end{aligned}$$

Idea

Generalize to arbitrary $p(\omega)$, i.e. $\Omega[f] := \frac{1}{2} \int \frac{|\hat{f}(\omega)|^2}{p(\omega)} d\omega$

Greens Function

Theorem

For regularization functionals $\Omega[f] := \frac{1}{2}\int \frac{|\hat{f}(\omega)|^2}{p(\omega)} d\omega$ the self-consistency condition

$$\langle Pk(x,\cdot), Pk(x',\cdot)\rangle = k^\top(x,\cdot)(P^*P)k(x',\cdot) = k(x,x')$$

is satisfied if k has $p(\omega)$ as its Fourier transform, i.e.,

$$k(x,x') = \int \exp(-i\langle\omega,(x-x')\rangle)p(\omega)d\omega$$

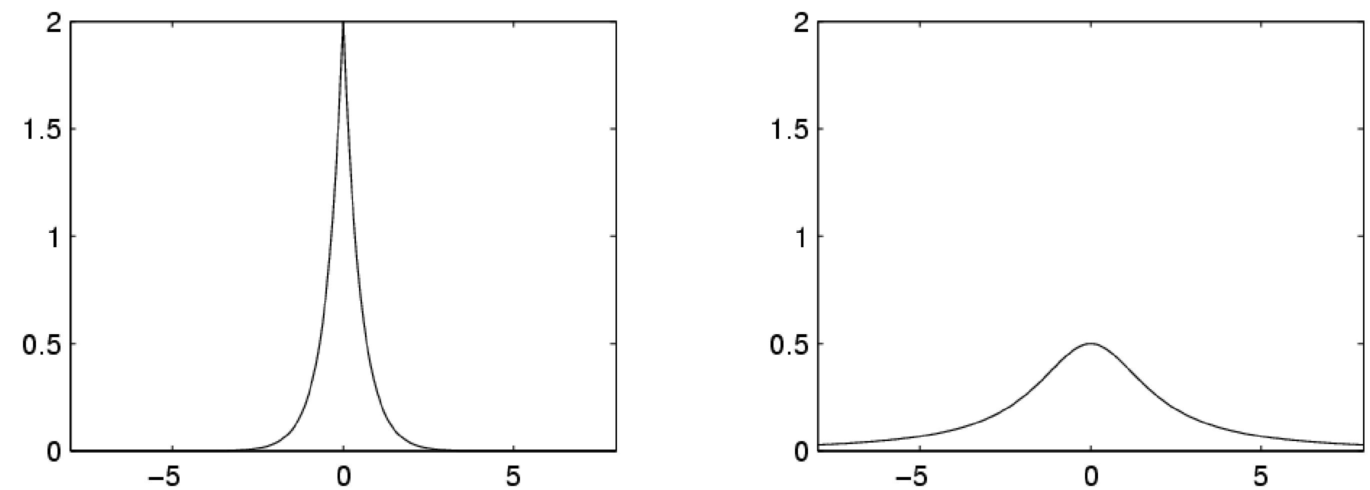
Consequences

- small $p(\omega)$ correspond to high penalty (regularization).
- $\Omega[f]$ is translation invariant, that is $\Omega[f(\cdot)] = \Omega[f(\cdot - x)]$.

Examples

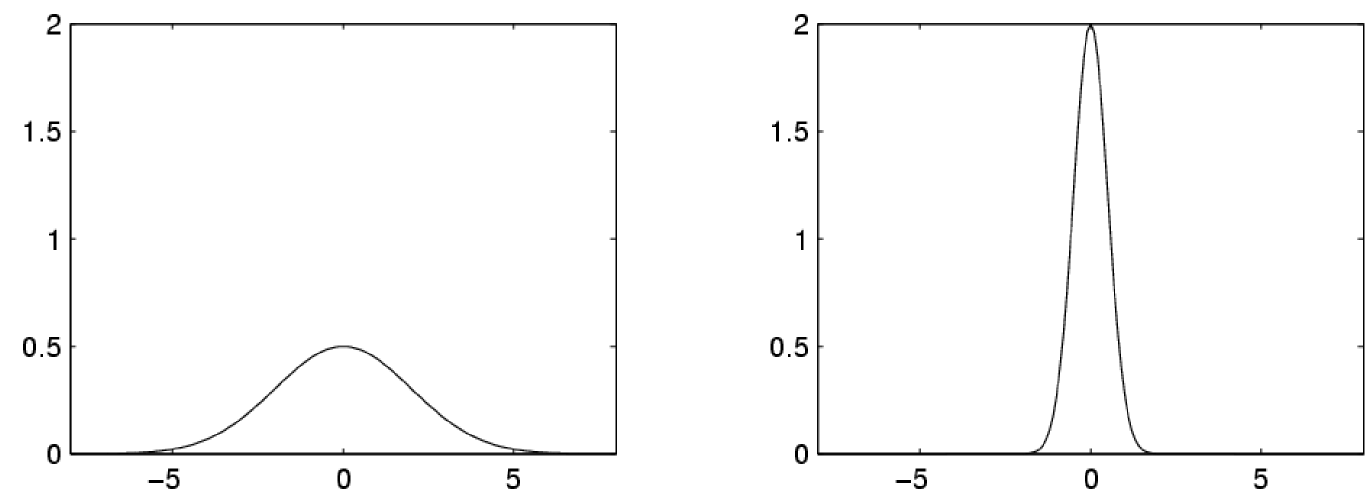
Laplacian Kernel

$$k(x, x') = \exp(-\|x - x'\|)$$
$$p(\omega) \propto (1 + \|\omega\|^2)^{-1}$$



Gaussian Kernel

$$k(x, x') = e^{-\frac{1}{2}\sigma^{-2}\|x-x'\|^2}$$
$$p(\omega) \propto e^{-\frac{1}{2}\sigma^{2}\|\omega\|^2}$$



Fourier transform of k shows regularization properties.
The more rapidly $p(\omega)$ decays, the more high frequencies are filtered out.

Rules of thumb

- Fourier transform is sufficient to check whether $k(x, x')$ satisfies Mercer's condition: only check if $\tilde{k}(\omega) \geq 0$.
- Example: $k(x, x') = \operatorname{sinc}(x - x')$.
  $\tilde{k}(\omega) = \chi_{[-\pi,\pi]}(\omega)$, hence k is a proper kernel.
- Width of kernel often more important than type of kernel (short range decay properties matter).
- Convenient way of incorporating prior knowledge, e.g.: for speech data we could use the autocorrelation function.
- Sum of derivatives becomes polynomial in Fourier space.

Polynomial Kernels

Functional Form

$$k(x, x') = \kappa(\langle x, x' \rangle)$$

Series Expansion

Polynomial kernels admit an expansion in terms of Legendre polynomials (L_n^N: order n in $\mathbb{R}^N$).

$$k(x, x') = \sum_{n=0}^{\infty} b_n L_n(\langle x, x' \rangle)$$

Consequence:

L_n (and their rotations) form an orthonormal basis on the unit sphere, P^*P is rotation invariant, and P^*P is diagonal with respect to L_n. In other words

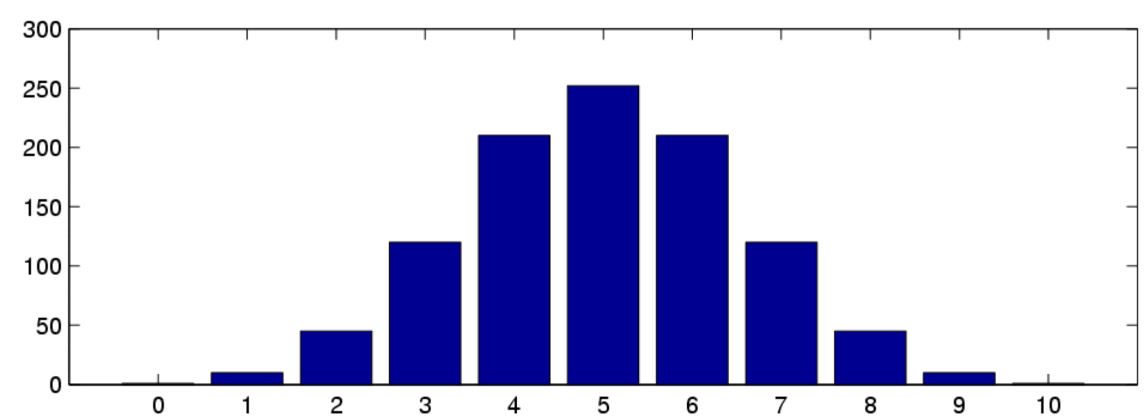
$$(P^*P)L_n(\langle x, \cdot \rangle) = b_n^{-1} L_n(\langle x, \cdot \rangle)$$

Polynomial Kernels

- Decay properties of b_n determine smoothness of functions specified by $k(\langle x, x' \rangle)$.
- For $N \to \infty$ all terms of L_n^N but x^n vanish, hence a Taylor series $k(x, x') = \sum_i a_i \langle x, x' \rangle^i$ gives a good guess.

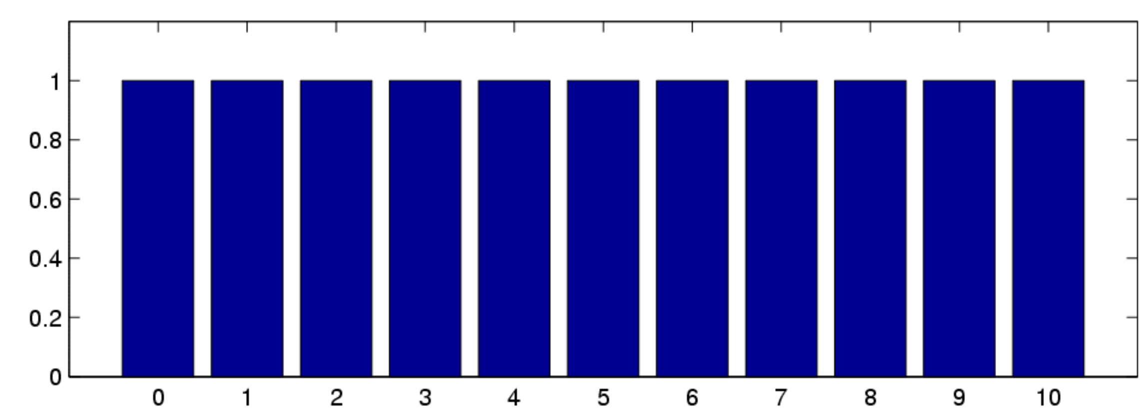
Inhomogeneous Polynomial

$$k(x, x') = (\langle x, x' \rangle + 1)^p$$
$$a_n = \binom{p}{n} \text{ if } n \leq p$$



Vovk's Real Polynomial

$$k(x, x') = \frac{1 - \langle x, x' \rangle^p}{1 - (\langle x, x' \rangle)}$$
$$a_n = 1 \text{ if } n < p$$



Mini Summary

Regularized Risk Functional

- From Optimization Problems to Loss Functions
- Regularization
- Safeguard against Overfitting

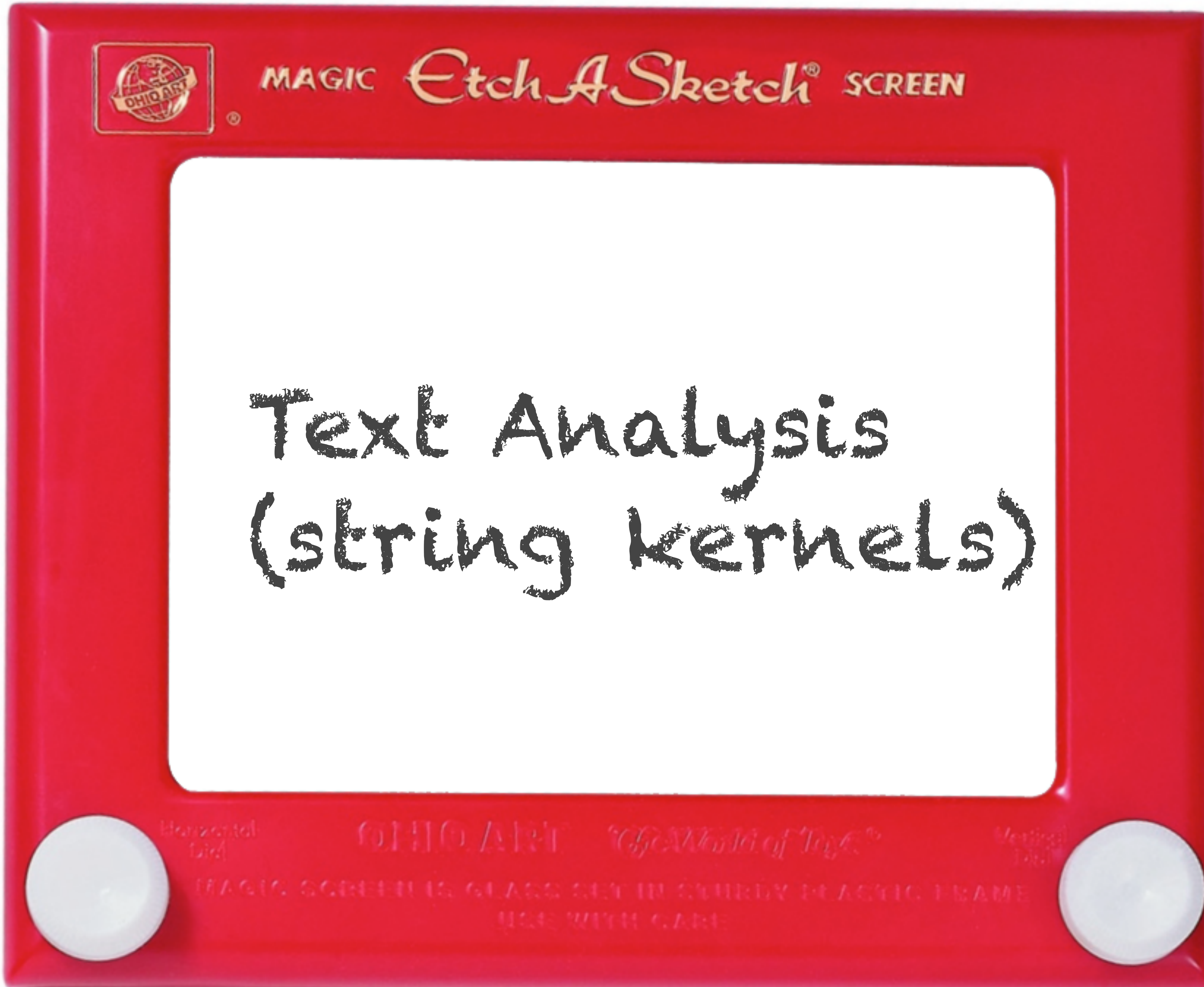
Regularization and Kernels

- Examples of Regularizers
- Regularization Operators
- Greens Functions and Self Consistency Condition

Fourier Regularization

- Translation Invariant Regularizers
- Regularization in Fourier Space
- Kernel is inverse Fourier Transformation of Weight

Polynomial Kernels and Series Expansions



Text Analysis (string kernels)

String Kernel (pre)History

The Kernel Perspective

- Design a kernel implementing good features

$$k(x, x') = \langle \phi(x), \phi(x') \rangle \text{ and } f(x) = \langle \phi(x), w \rangle = \sum_i \alpha_i k(x_i, x)$$

- Many variants
  - Bag of words (AT&T labs 1995, e.g. Vapnik)
  - Matching substrings (Haussler, Watkins 1998)
  - Spectrum kernel (Leslie, Eskin, Noble, 2000)
  - Suffix tree (Vishwanathan, Smola, 2003)
  - Suffix array (Teo, Vishwanathan, 2006)
  - Rational kernels (Mohri, Cortes, Haffner, 2004 ...)

Bag of words

- At least since 1995 known in AT&T labs

$$k(x, x') = \sum_w n_w(x) n_w(x') \text{ and } f(x) = \sum_w \omega_w n_w(x')$$

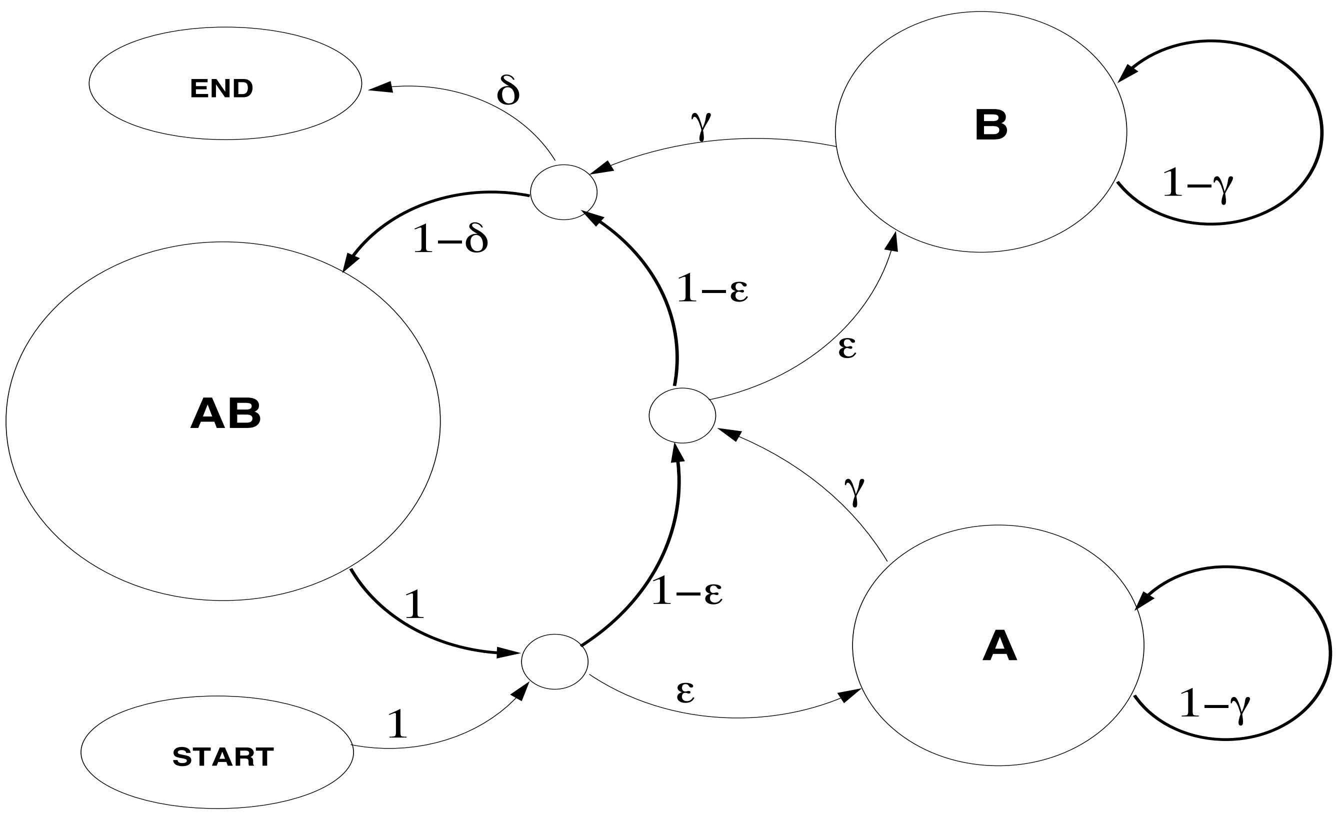
`(to be or not to be)` → `(be:2, or:1, not:1, to:2)`

- Joachims 1998: Use sparse vectors
- Haffner 2001: Inverted index for faster training
- Lots of work on feature weighting (TF/IDF)
- Variants of it deployed in many spam filters

Substring (mis)matching

- Watkins 1998+99 (dynamic alignment, etc)
- Haussler 1999 (convolution kernels)

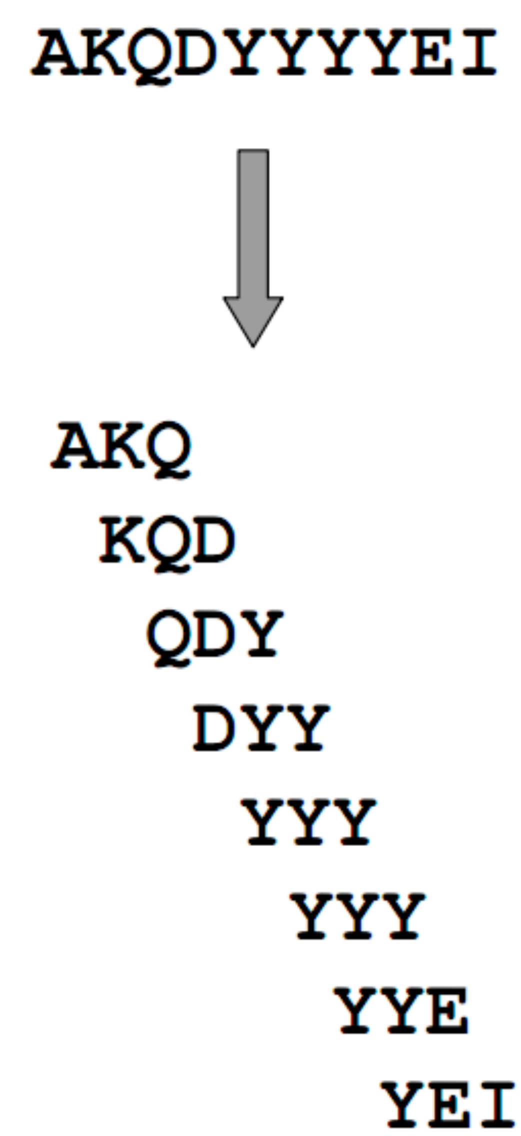
$$k(x, x') = \sum_{w \in x} \sum_{w' \in x'} \kappa(w, w')$$

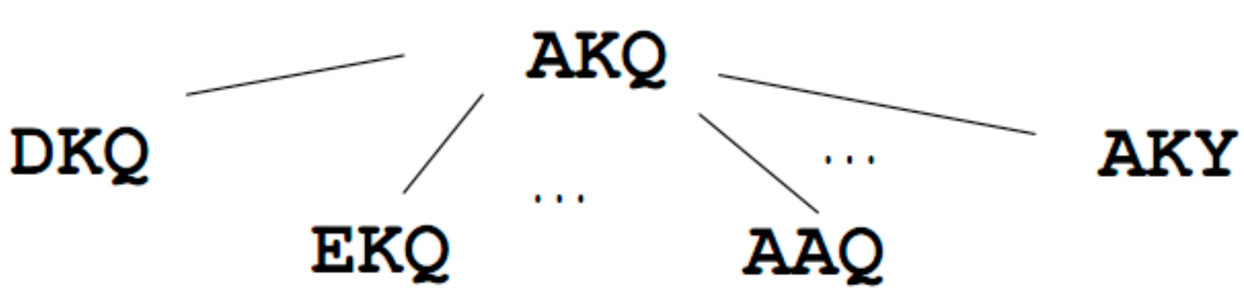


- In general O(x x′) runtime
  (e.g. Cristianini, Shawe-Taylor, Lodhi, 2001)
- Dynamic programming solution for pair-HMM

Spectrum Kernel

- Leslie, Eskin, Noble & coworkers, 2002
- Key idea is to focus on features directly
  - Linear time operation to get features
  - Limited amount of mismatch (exponential in number of missed chars)
  - Explicit feature construction (good & fast for DNA sequences)



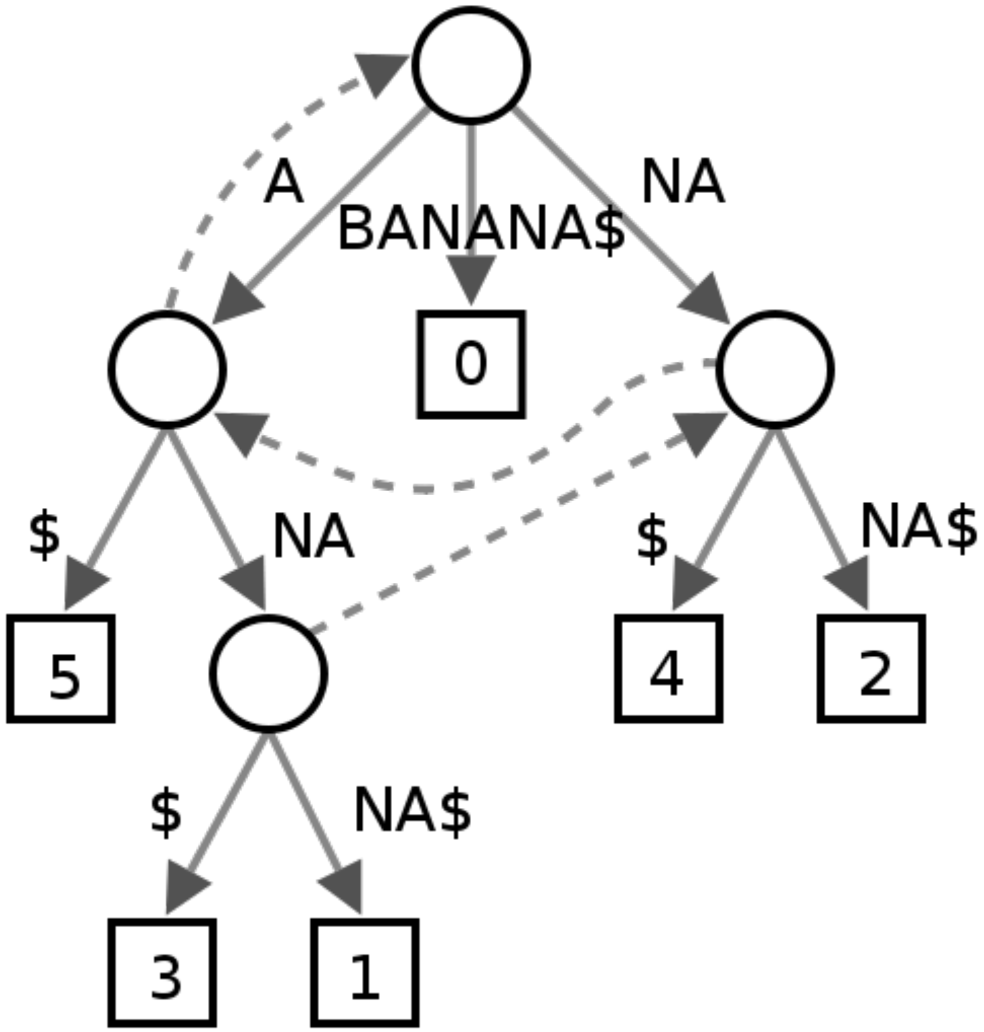


Suffix Tree Kernel

- Vishwanathan & Smola, 2003 (O(x + x′) time)
- Mismatch-free kernel + arbitrary weights

$$k(x, x') = \sum_w \omega_w n_w(x) n_w(x')$$

- Linear time construction (Ukkonen, 1995)
- Find matches for second string in linear time (Chang & Lawler, 1994)
- Precompute weights on path



Are we done?

- Large vocabulary size
- Need to build dictionary
- Approximate matches are still a problem
- Suffix tree/array is storage inefficient (40-60x)
- Realtime computation
- Memory constraints (keep in RAM)
- Difficult to implement

Multitask Learning

Multitask Learning

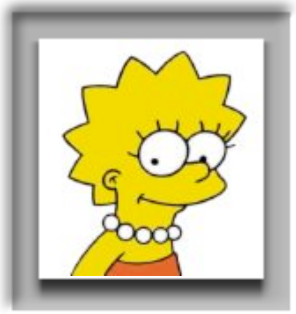
From: bat <kilian@gmail.com>
Subject: **hey whats up check this meds place out**
Date: April 6, 2009 10:50:13 PM PDT
To: Kilian Weinberger
Reply-To: bat <kilian@gmail.com>

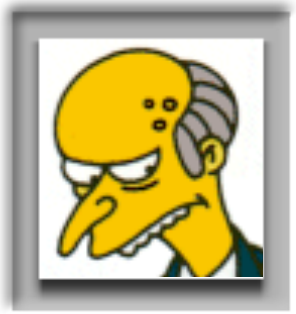
Your friend (kilian@gmail.com) has sent you a link to the following Scout.com story:
Savage Hall Ground-Breaking Celebration

Get Vicodin, Valium, Xanax, Viagra, Oxycontin, and much more. Absolutely No Prescription Required. Over Night Shipping! Why should you be risking dealing with shady people. Check us out today!
http://jenkinste[illegible]3.blogspot.com

The University of Toledo will hold a ground-breaking celebration to kick-off the UT Athletics Complex and Savage Hall renovation project on Wednesday, December 12th at Savage Hall.

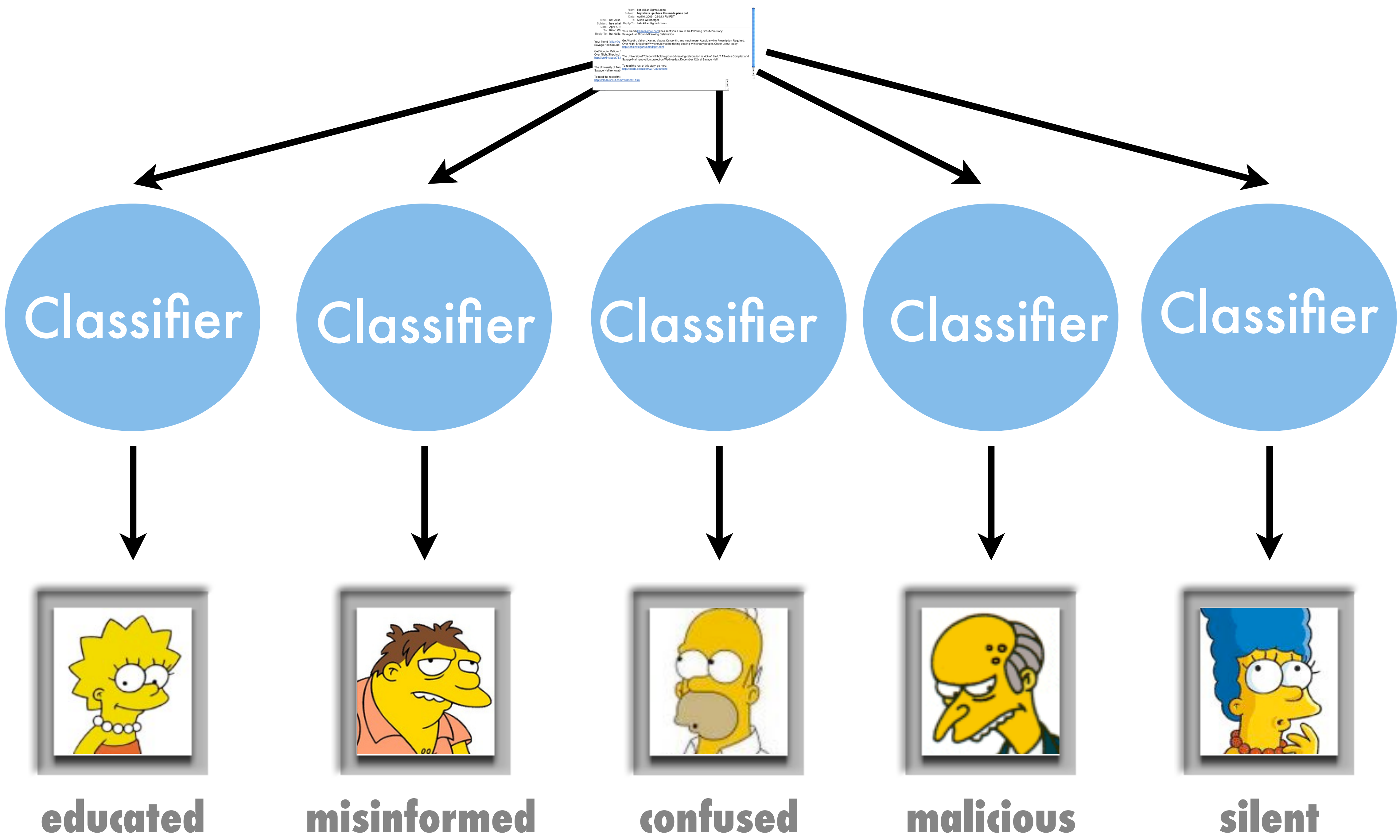
To read the rest of this story, go here:
http://toledo.scout.com/2/708390.html



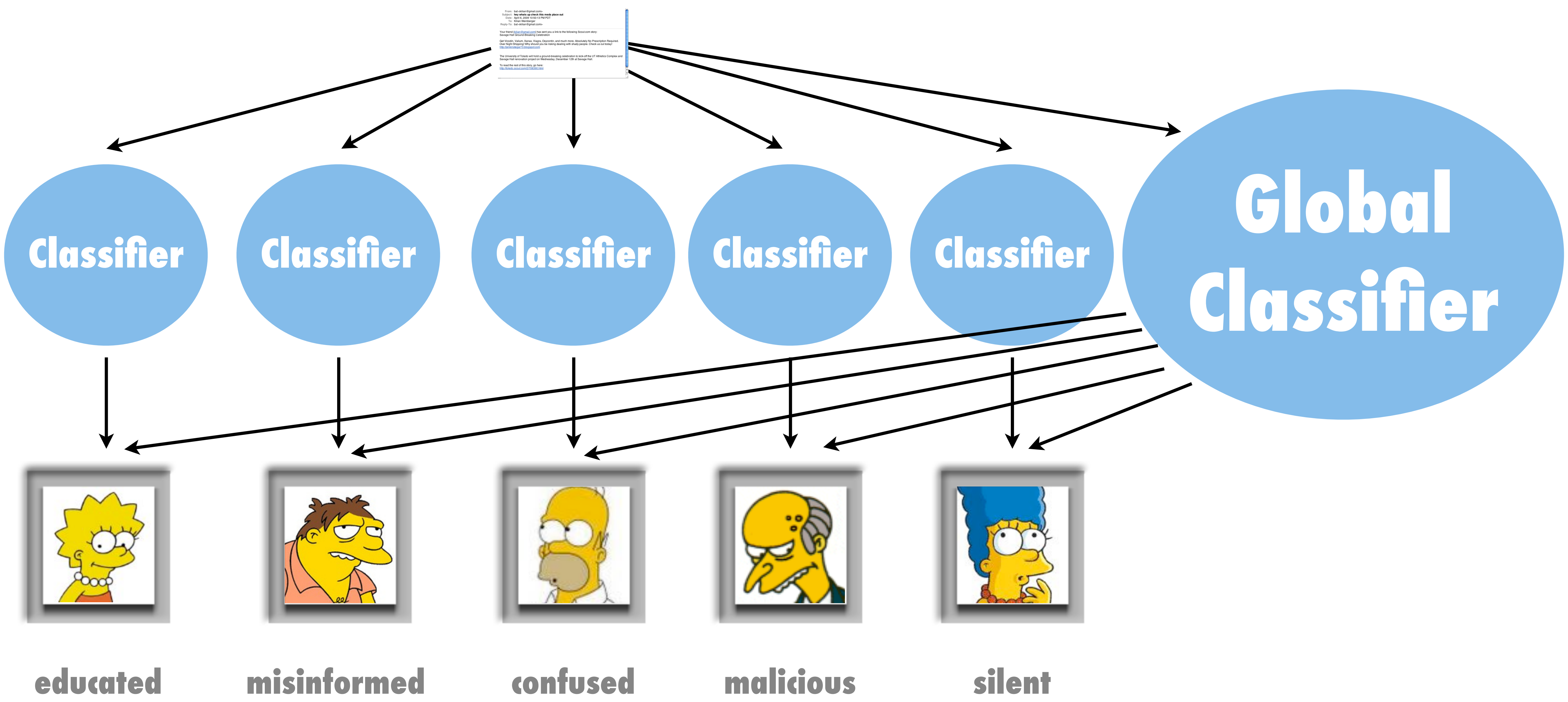




Multitask Learning

Multitask Learning



Multitask Learning



Carnegie Mellon University

Collaborative Classification

- **Primal representation**

$$f(x, u) = \langle \phi(x), w \rangle + \langle \phi(x), w_u \rangle = \langle \phi(x) \otimes (1 \oplus e_u), w \rangle$$

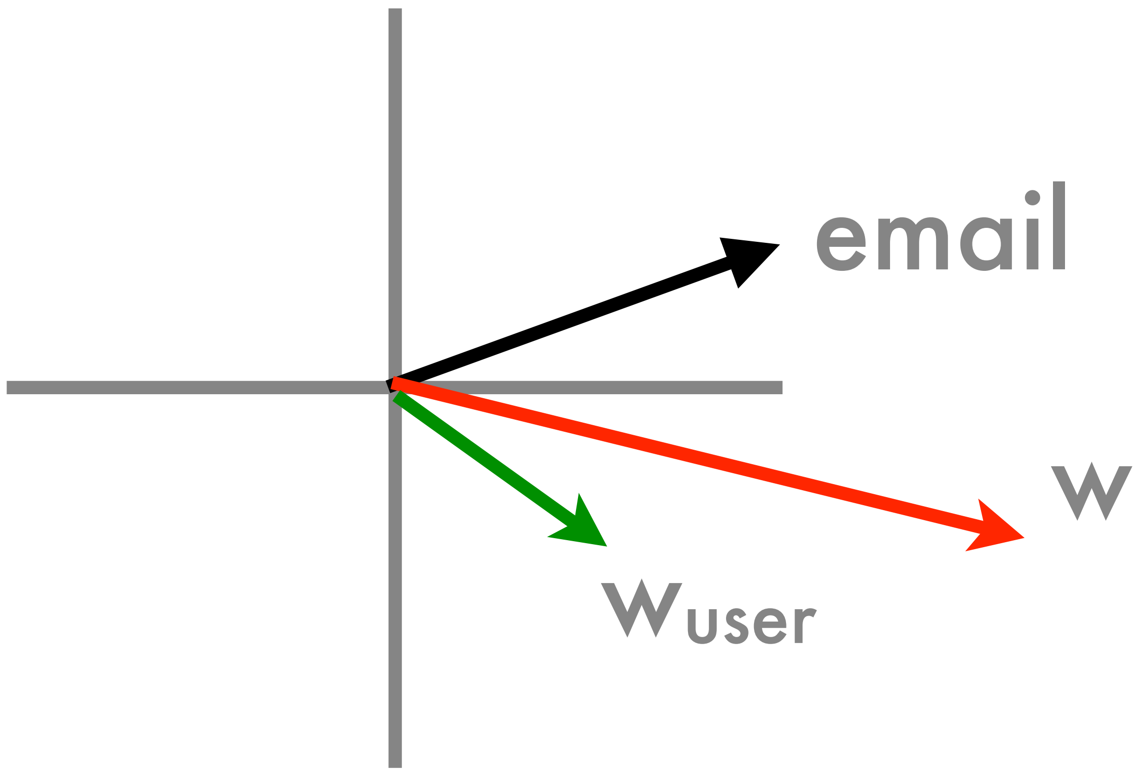
Kernel representation

$$k((x, u), (x', u')) = k(x, x')[1 + \delta_{u,u'}]$$

Multitask kernel (e.g. Pontil & Michelli, Daume). Usually does not scale well ...

- **Problem -** dimensionality is 10^{13}. That is 40TB of space

Collaborative Classification



- **Primal representation**

$$f(x, u) = \langle \phi(x), w \rangle + \langle \phi(x), w_u \rangle = \langle \phi(x) \otimes (1 \oplus e_u), w \rangle$$

Kernel representation

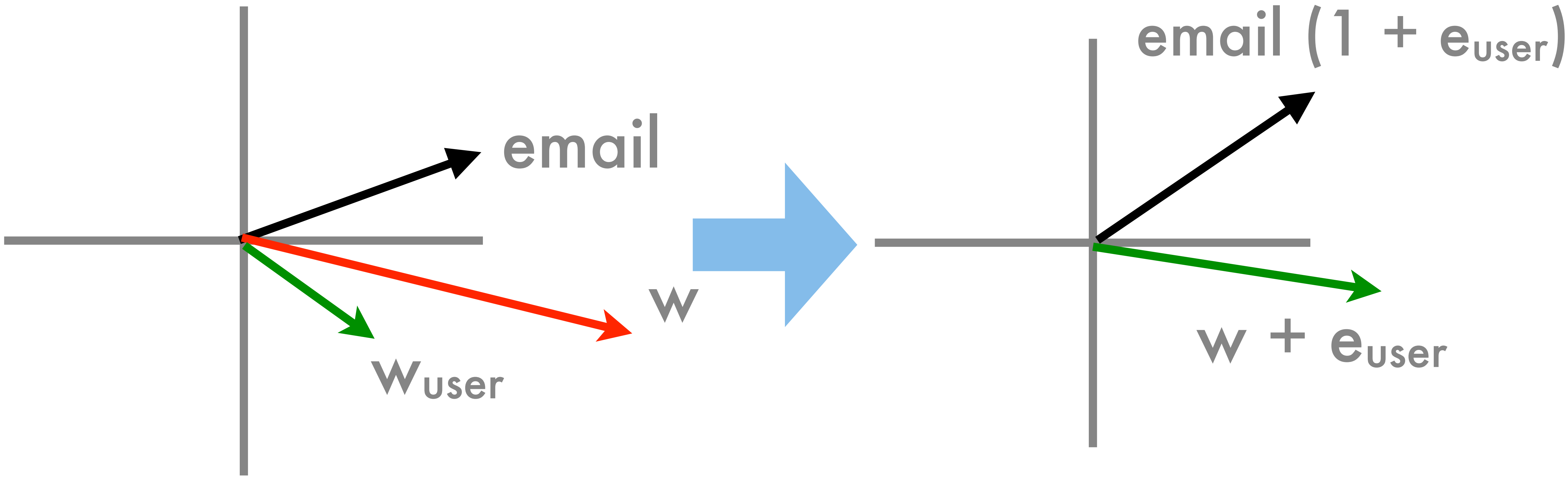
$$k((x, u), (x', u')) = k(x, x')[1 + \delta_{u,u'}]$$

Multitask kernel (e.g. Pontil & Michelli, Daume). Usually does not scale well ...

- **Problem -** dimensionality is 10^{13}. That is 40TB of space

Carnegie Mellon University

Collaborative Classification



- **Primal representation**

$$f(x,u) = \langle \phi(x), w \rangle + \langle \phi(x), w_u \rangle = \langle \phi(x) \otimes (1 \oplus e_u), w \rangle$$

Kernel representation

$$k((x,u),(x',u')) = k(x,x')[1 + \delta_{u,u'}]$$

Multitask kernel (e.g. Pontil & Michelli, Daume). Usually does not scale well ...

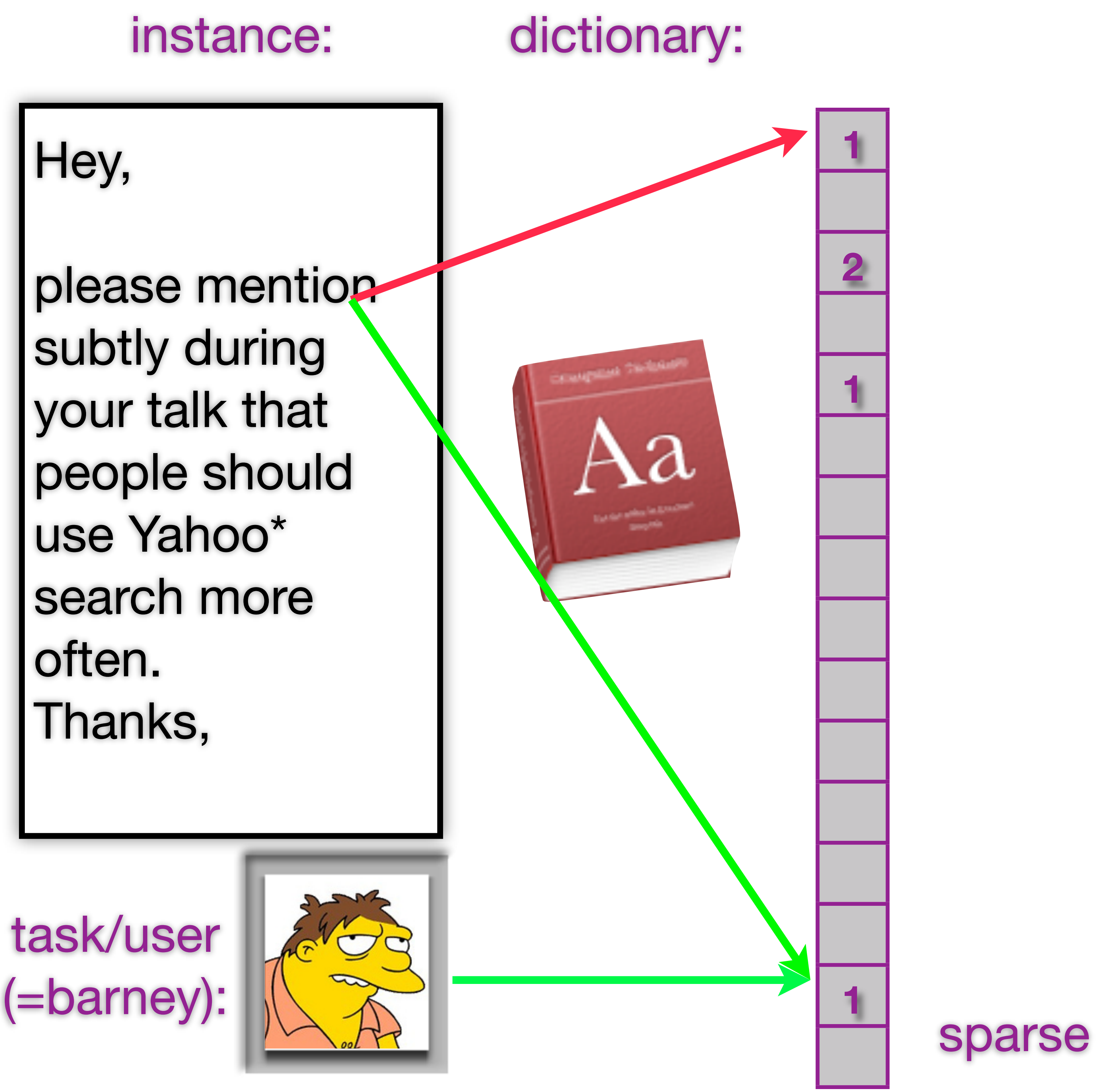
- **Problem -** dimensionality is 10^{13}. That is 40TB of space

Hashing

Hash Kernels

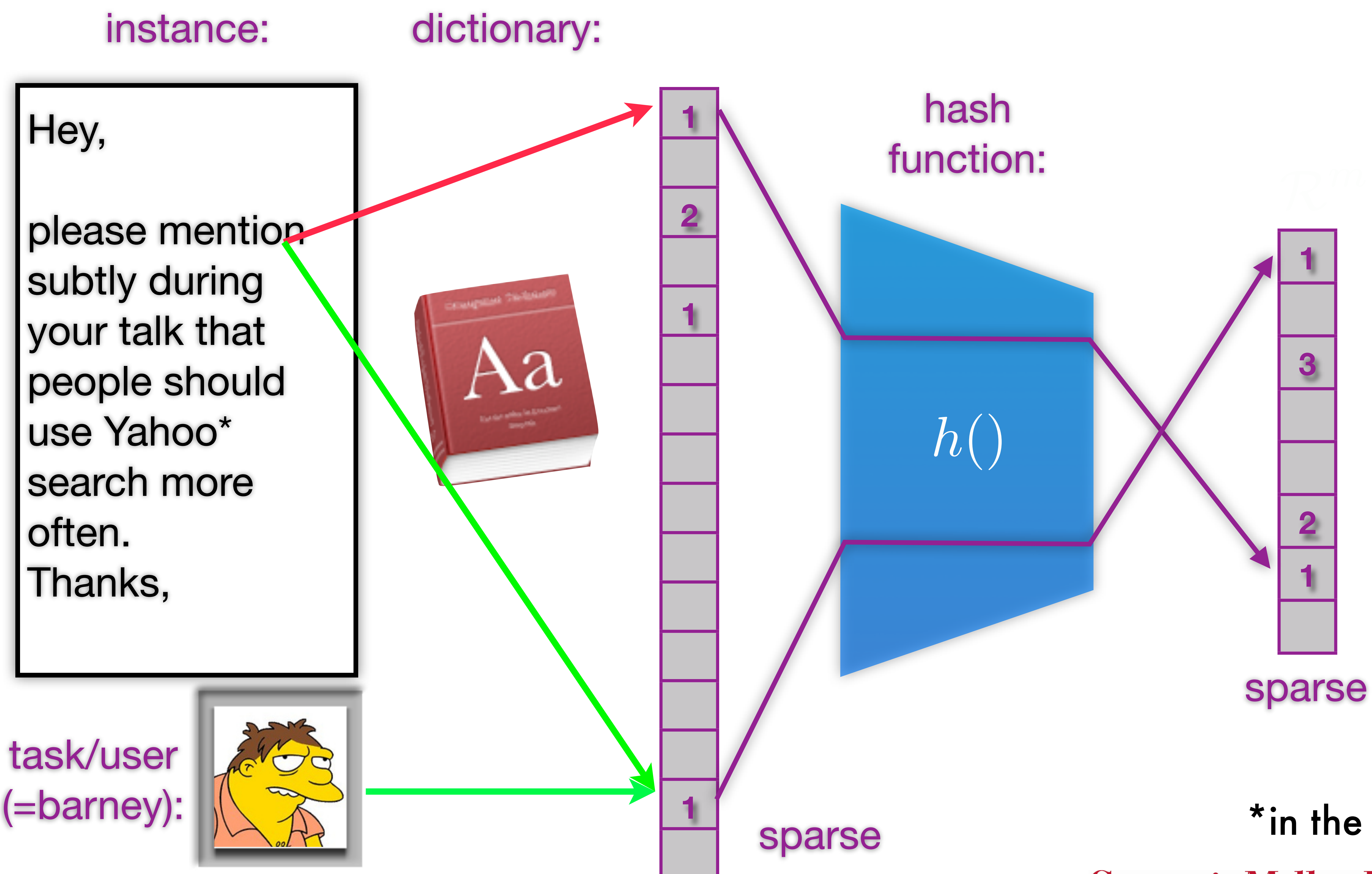
*in the old days

Hash Kernels



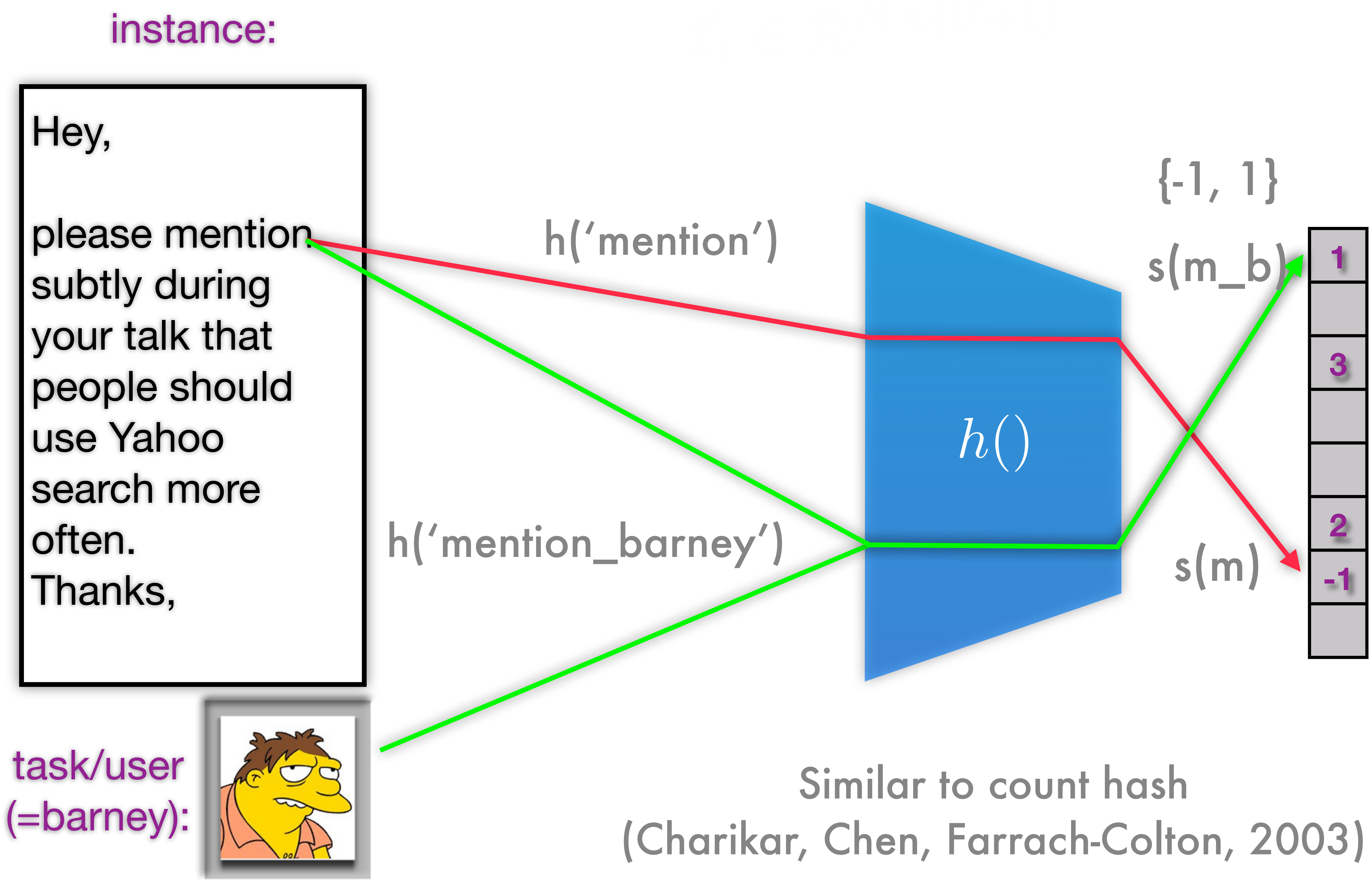
*in the old days

Hash Kernels



*in the old days

Hash Kernels



Similar to count hash
(Charikar, Chen, Farrach-Colton, 2003)

Advantages of hashing

Advantages of hashing

- No dictionary!
  - Content drift is no problem
  - All memory used for classification
  - Finite memory guarantee (via online learning)

Advantages of hashing

- No dictionary!
  - Content drift is no problem
  - All memory used for classification
  - Finite memory guarantee (via online learning)
- No Memory needed for projection. (vs LSH)

Carnegie Mellon University

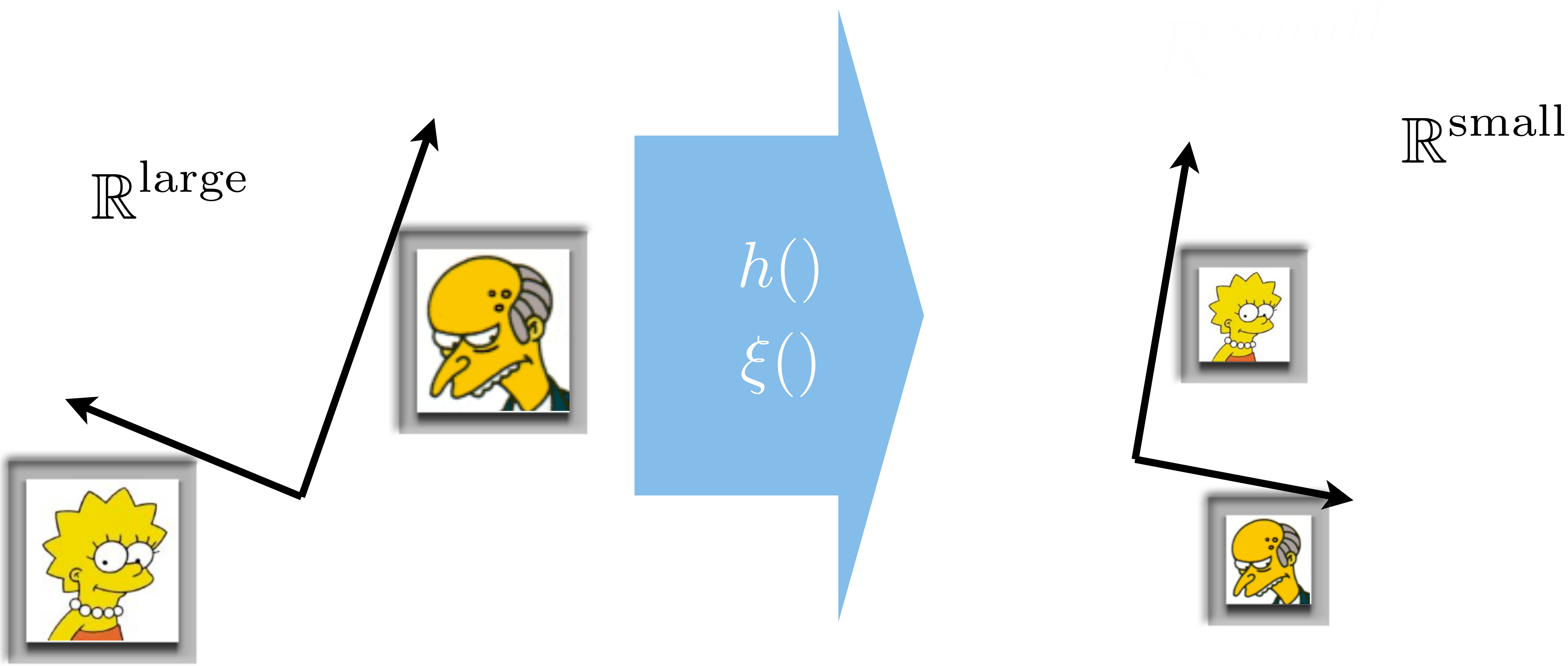
Advantages of hashing

- No dictionary!
  - Content drift is no problem
  - All memory used for classification
  - Finite memory guarantee (via online learning)
- No Memory needed for projection. (vs LSH)
- Implicit mapping into high dimensional space!

Advantages of hashing

- No dictionary!
  - Content drift is no problem
  - All memory used for classification
  - Finite memory guarantee (via online learning)
- No Memory needed for projection. (vs LSH)
- Implicit mapping into high dimensional space!
- It is sparsity preserving! (vs LSH)

Approximate Orthogonality



We can do multi-task learning!

Guarantees

- For a random hash function the inner product vanishes with high probability via

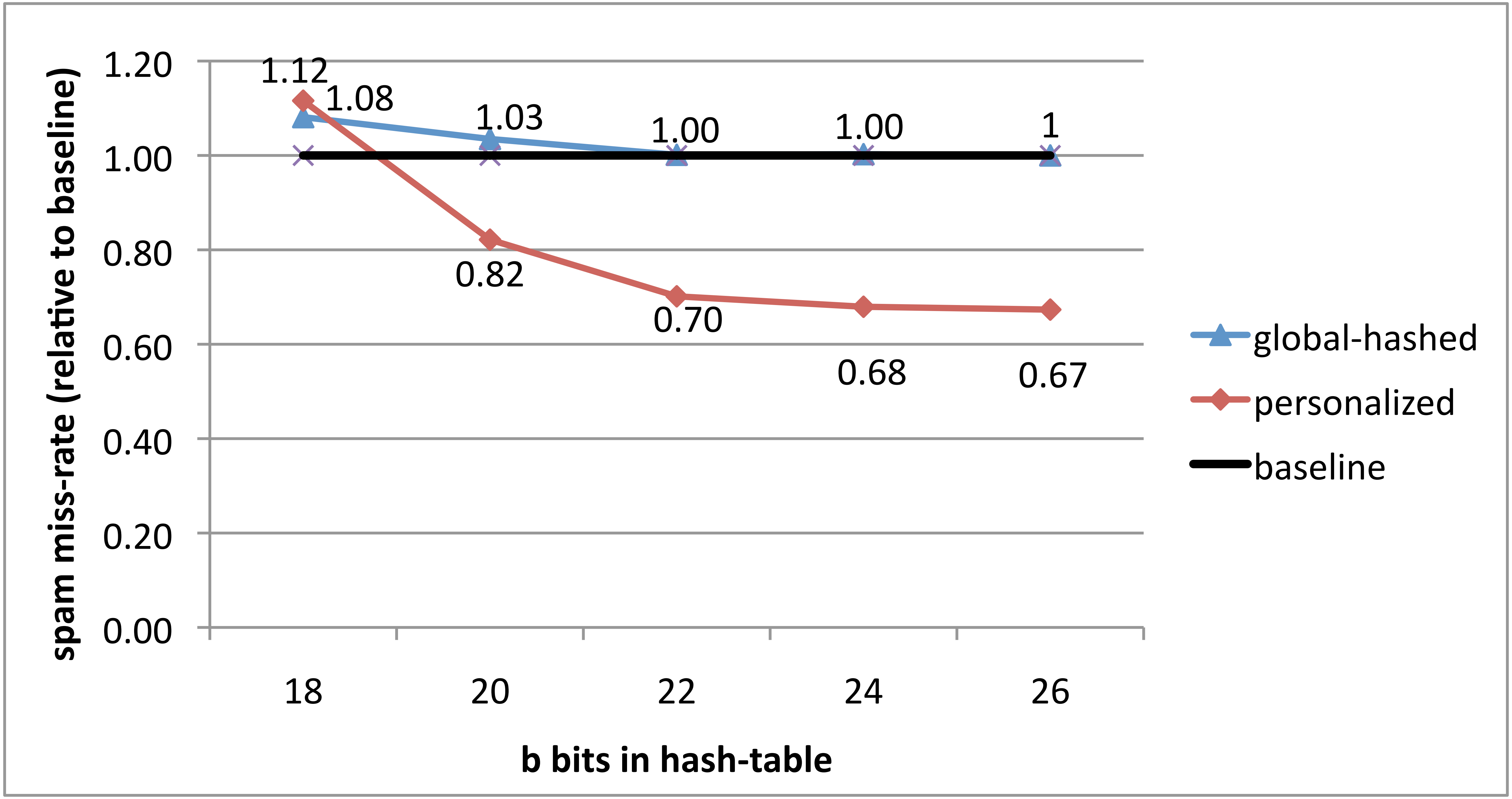
$$\Pr\{|\langle w_v, h_u(x)\rangle| > \epsilon\} \leq 2e^{-C\epsilon^2 m}$$

- We can use this for multitask learning

Direct sum in Hilbert Space → **Sum** in Hash Space

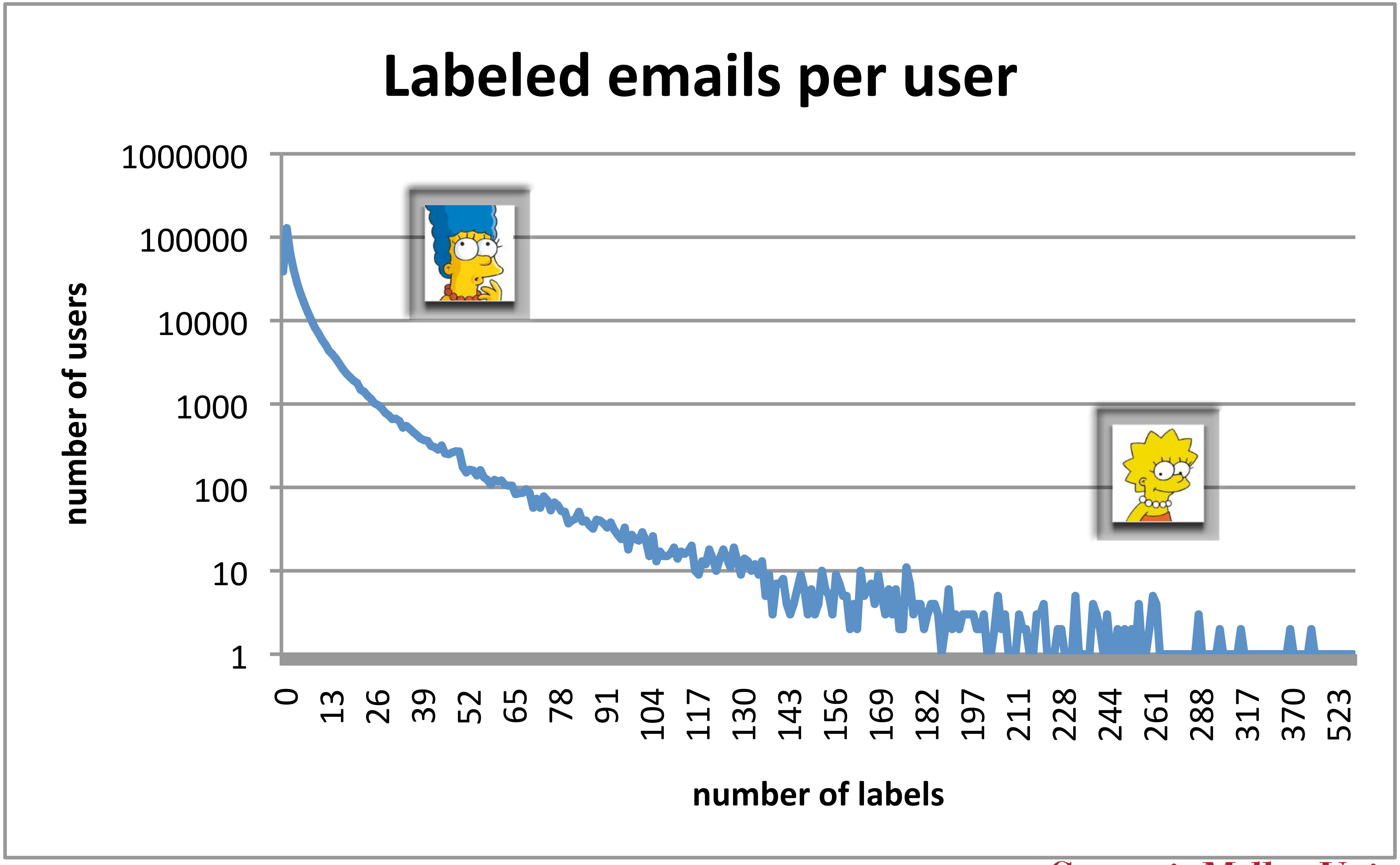
- The hashed inner product is unbiased
  Proof: take expectation over random signs
- The variance is O(1/n)
  Proof: brute force expansion
- Restricted isometry property (Kumar, Sarlos, Dasgupta 2010)

Spam classification results



N=20M, U=400K

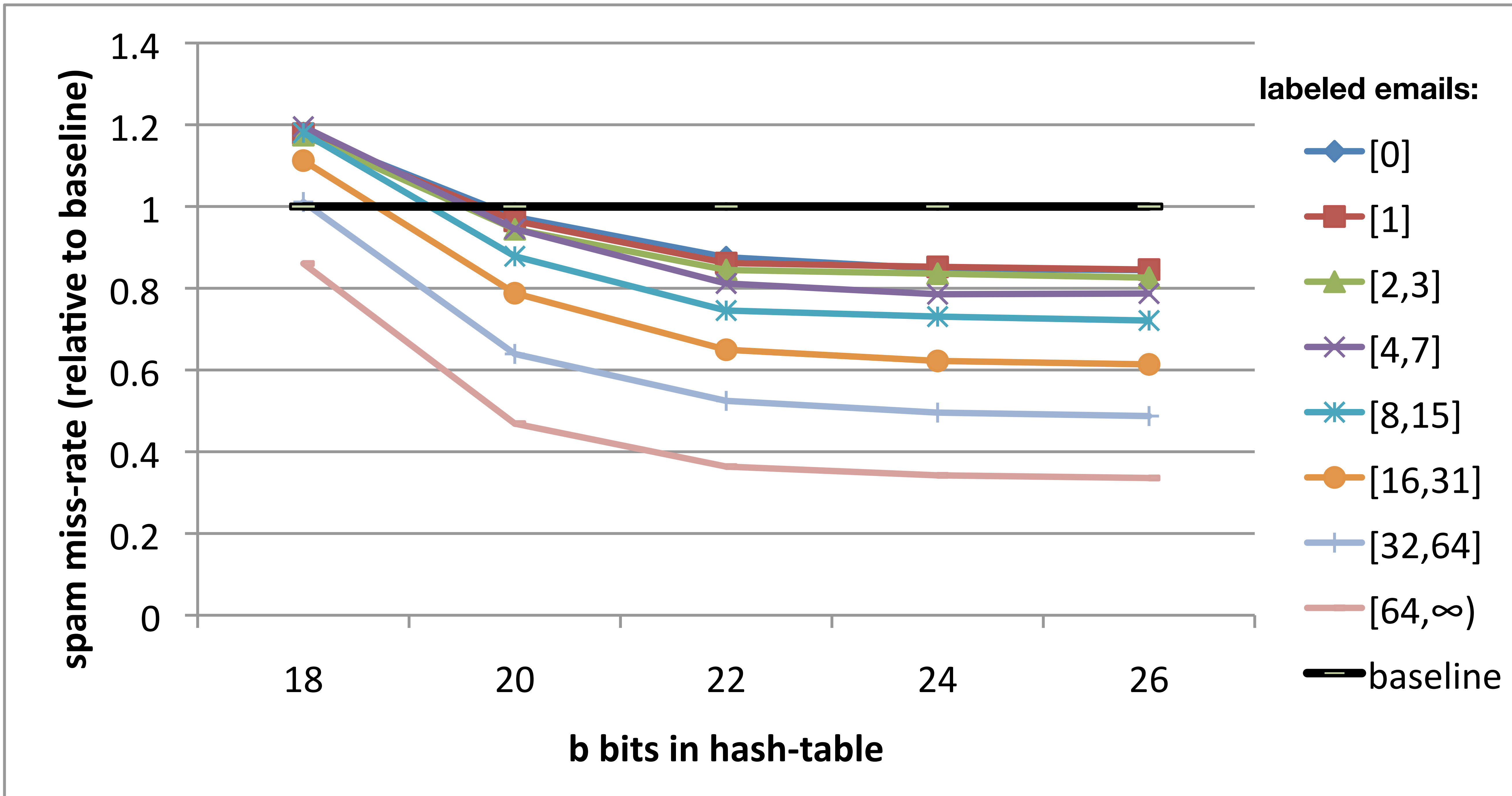
Lazy users …



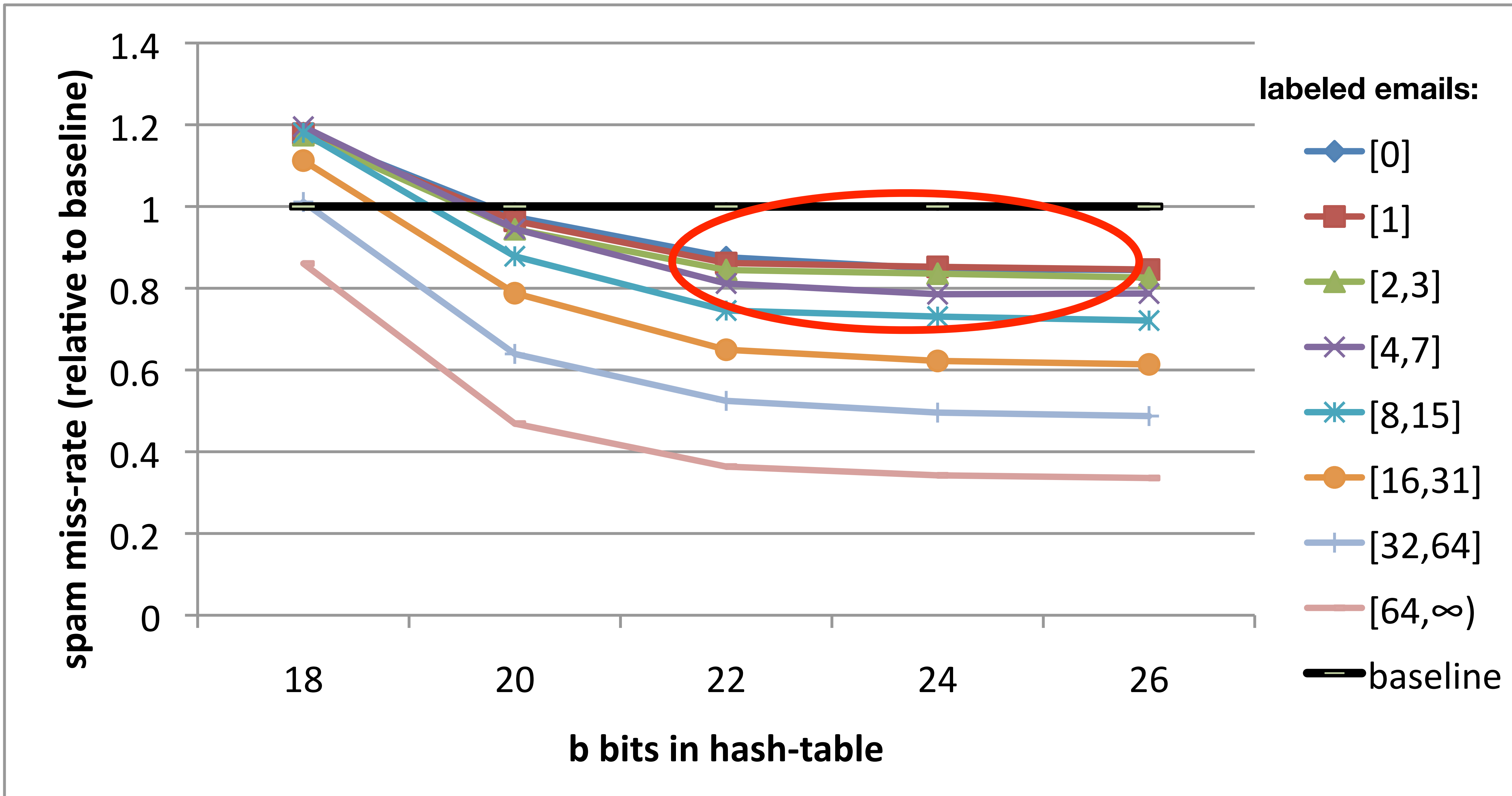
Labeled emails per user
1000000
100000
10000
1000
100
10
1
number of users
0
13
26
39
52
65
78
91
104
117
130
143
156
169
182
197
211
228
244
261
288
317
370
523
number of labels

Results by user group

Results by user group



Results by user group



Details

Estimation details

- Works best with stochastic gradient descent (or any other primal space method)
- Never instantiate hash map explicitly

$$f(x) = \langle w, \phi(x) \rangle = \sum_s w[h(s)] n_s(x)$$

- Random memory access pattern (latency)
- Multiclass classification - joint hash

Approximate Matches

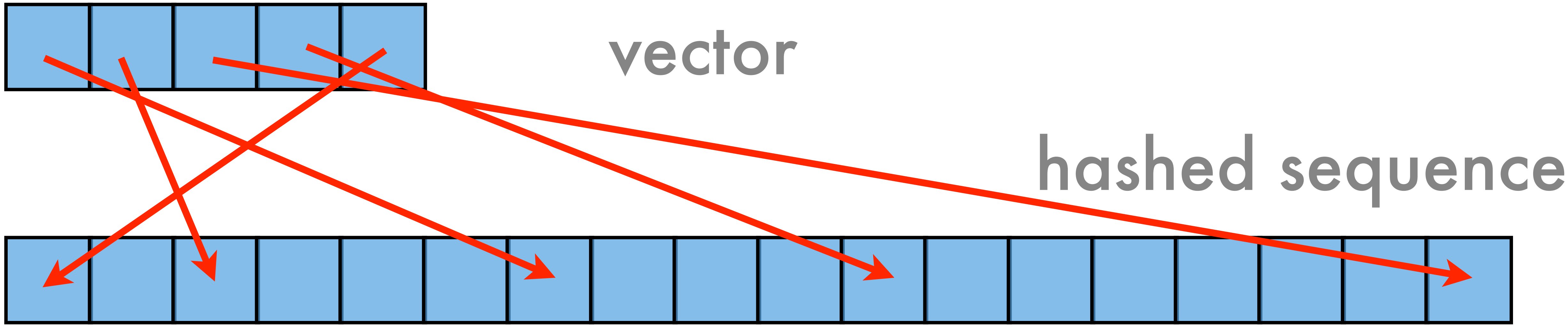
- General idea

$$k(x, x') = \sum_{w \in x} \sum_{w' \in x'} \kappa(w, w') \text{ for } |w - w'| \leq \delta$$

- Simplification
  - Weigh by mismatch amount |w-w'|
  - Map into fragments: `dog -> (*og, d*g, do*)`
  - Hash fragments and weigh them based on mismatch amount
  - Exponential in amount of mismatch
    But not in alphabet size

Memory access patterns

- Cache size is a few MBs
  Very fast random memory access
- RAM (DDR3 or better) is GBs
  - Fast sequential memory access (burst read)
  - CPU caches memory read from RAM
  - Random memory access is very slow
  - CPU caches memory read from RAM



Speeding up access

- Key idea - bound the range of h(i,j)

for j=1 to n access h(i,j)

- Linear offset
  bad collisions in i

$$h(i, j) = h(i) + j$$

- Sum of hash functions
  bad collisions in j

$$h(i, j) = h(i) + h'(j)$$

- Optimal Golomb Ruler (Langford)
  NP hard in general

$$h(i, j) = h(i) + \mathrm{OGR}(j)$$

- Feistel Network / Cryptography (new)

$$h(i, j) = h(i) + \mathrm{crypt}(j|i)$$