Large-Scale Machine Learning
I. Scalability issues

Jean-Philippe Vert

jean-philippe.vert@{mines-paristech,curie,ens}.fr

Outline

Acknowledgement

In the preparation of these slides I got inspiration and copied several slides from several sources:

- Sanjiv Kumar's "Large-scale machine learning" course: http://www.sanjivk.com/EECS6898/lectures.html
- Ala Al-Fuqaha's "Data mining" course: https://cs.wmich.edu/alfuqaha/summer14/cs6530/lectures/SimilarityAnalysis.pdf
- Léon Bottou's "Large-scale machine learning revisited" conference https://bigdata2013.sciencesconf.org/conference/bigdata2013/pages/bottou.pdf

Outline

BUSINESS INSIDER INDIA

TECH

2017 is the year of Machine Learning. Here's why

GAURAV SANGWANI | 0 | JAN 13, 2017, 12.51 PM

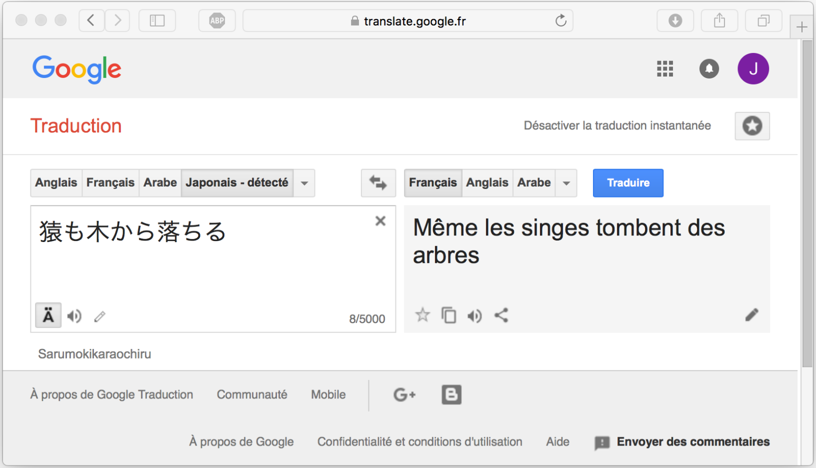
Facebook Linkedin Twitter Google+ Reddit

Machine learning is maybe the most sweltering thing in Silicon Valley at this moment. Particularly deep learning. The reason why it is so hot is on the grounds that it can assume control of numerous repetitive, thoughtless tasks. It'll improve doctors, and make lawyers better lawyers. What's more, it makes cars drive themselves.

Perception

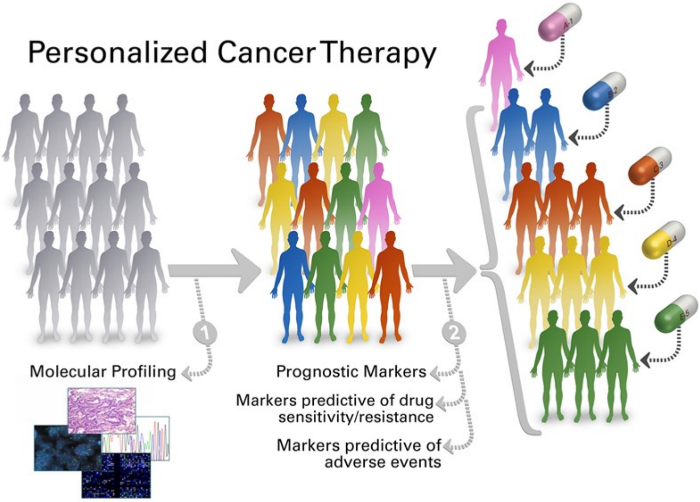

Dr. France Prešeren monument
(1905) by Ivan Zajc and Maks Fabiani... >
History
Audio
Video

Communication



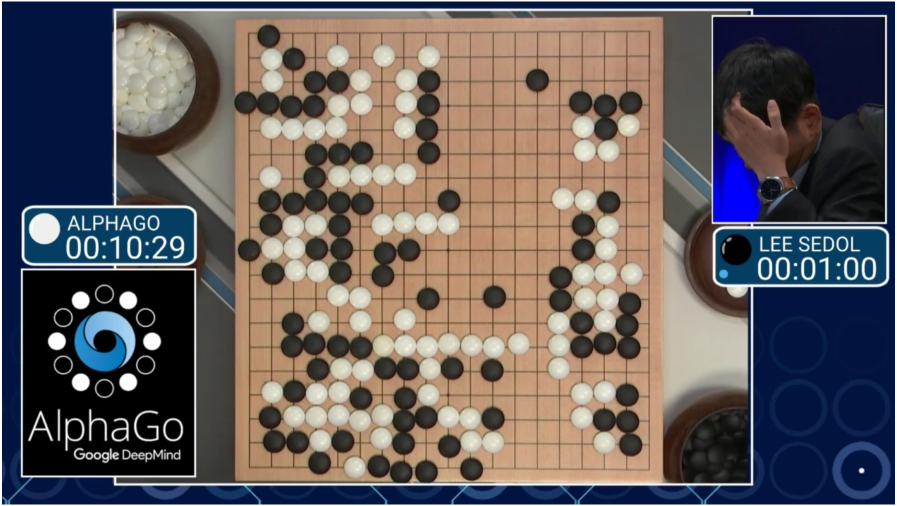
Mobility

Health

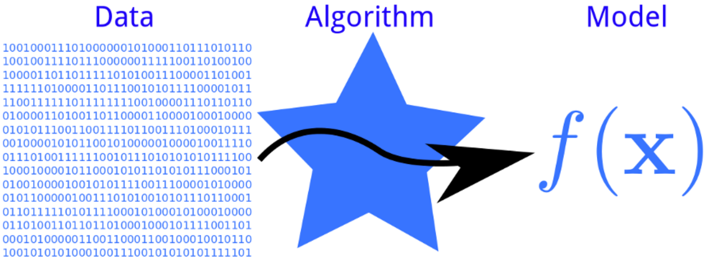


https://pct.mdanderson.org

Reasoning



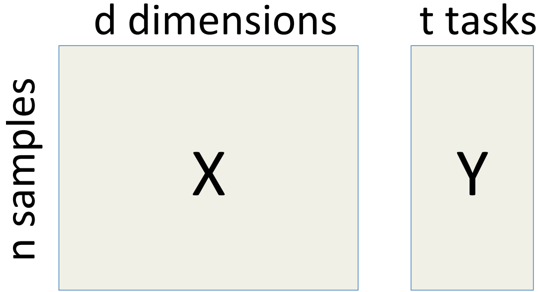
A common process: learning from data



https://www.linkedin.com/pulse/supervised-machine-learning-pega-decisioning-solution-nizam-muhammad

- Given examples (training data), make a machine learn how to predict on new samples, or discover patterns in data
- Statistics + optimization + computer science
- Gets better with more training examples and bigger computers

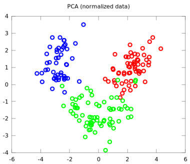
Large-scale ML?

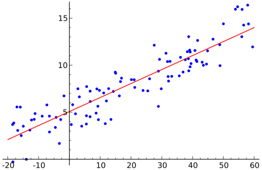


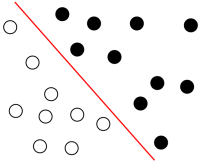
- Iris dataset: $n = 150, d = 4, t = 1$
- Cancer drug sensitivity: $n = 1k, d = 1M, t = 100$
- Imagenet: $n = 14M, d = 60k+, t = 22k$
- Shopping, e-marketing $n = O(M), d = O(B), t = O(100M)$
- Astronomy, GAFA, web... $n = O(B), d = O(B), t = O(B)$

Today's goals

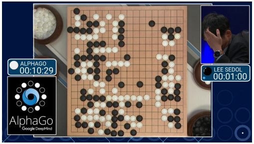
1. Review a few standard ML techniques







2. Introduce a few ideas and techniques to scale them to modern, big datasets



Outline

1. Introduction
2. Standard machine learning
   - Dimension reduction: PCA
   - Clustering: k-means
   - Regression: ridge regression
   - Classification: kNN, logistic regression and SVM
   - Nonlinear models: kernel methods
3. Scalability issues

Main ML paradigms

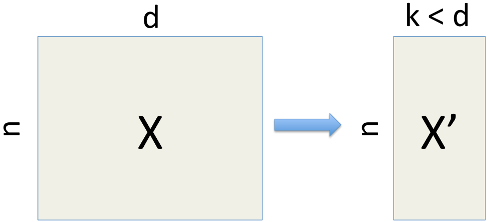
- Unsupervised learning
  - Dimension reduction
  - Clustering
  - Density estimation
  - Feature learning
- Supervised learning
  - Regression
  - Classification
  - Structured output classification
- Semi-supervised learning
- Reinforcement learning

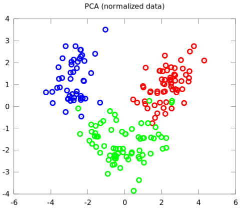
Main ML paradigms

- Unsupervised learning
  - Dimension reduction: PCA
  - Clustering: k-means
  - Density estimation
  - Feature learning
- Supervised learning
  - Regression: OLS, ridge regression
  - Classification: kNN, logistic regression, SVM
  - Structured output classification
- Semi-supervised learning
- Reinforcement learning

Outline

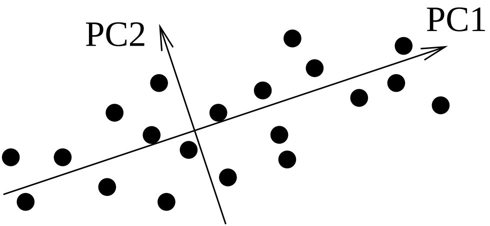
Motivation





- Dimension reduction
- Preprocessing (remove noise, keep signal)
- Visualization ($k = 2, 3$)
- Discover structure

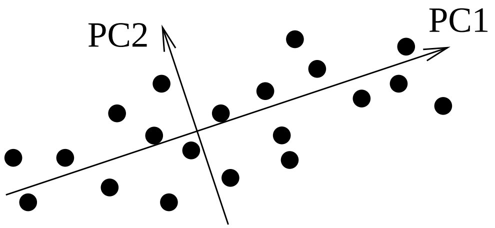
PCA definition



- Training set $\mathcal{S} = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$
- For $i = 1, \ldots, k \leq d$, PC_i is the linear projection onto the direction that captures the largest amount of variance and is orthogonal to the previous ones:

$$u_i \in \underset{\|u\|=1,\, u \perp \{u_1, \ldots, u_{i-1}\}}{\operatorname{argmax}} \sum_{i=1}^{n} \left(x_i^\top u - \frac{1}{n} \sum_{j=1}^{n} x_j^\top u \right)^2$$

PCA solution

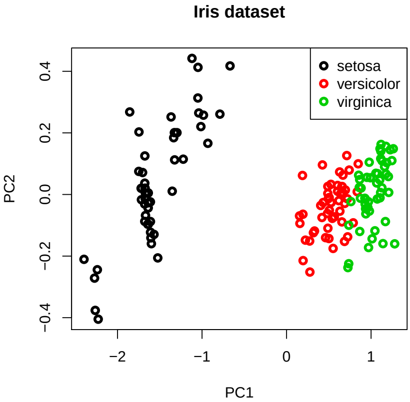


- Let $\tilde{X}$ be the centered $n \times d$ data matrix
- PCA solves, for $i = 1, \ldots, k \leq d$:

$$u_i \in \underset{\| u \|=1,\, u \perp \{u_1, \ldots, u_{i-1}\}}{\operatorname{argmax}} u^\top \tilde{X}^\top \tilde{X} u$$

- Solution: u_i is the i-th eigenvector of $C = \tilde{X}^\top \tilde{X}$, the empirical covariance matrix

PCA example




```
> data(iris)
> head(iris, 3)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
> m <- princomp(log(iris[,1:4]))
```

PCA complexity

- Memory: store X and C: $O(max(nd, d^2))$
- Compute C: $O(nd^2)$
- Compute k eigenvectors of C (power method): $O(kd^2)$

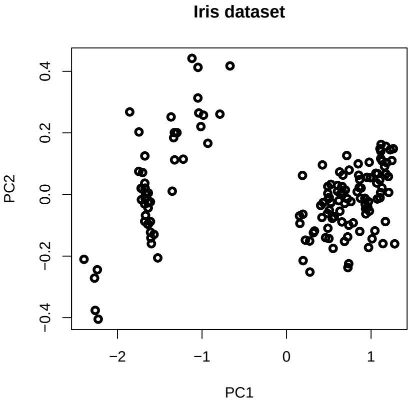
Computing C is more expensive than computing its eigenvectors ($n > k$)!

$n = 1B, d = 100M$
Store C: $40,000\,TB$
Compute C: $2 \times 10^{25} FLOPS = 20 yottaFLOPS$ (about 300 years of the most powerful supercomputer in 2016)

Outline

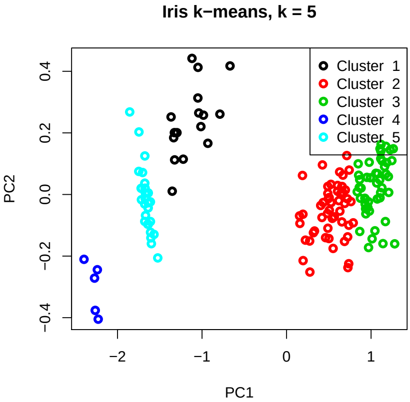
1. Introduction
2. Standard machine learning
   - Dimension reduction: PCA
   - Clustering: k-means
   - Regression: ridge regression
   - Classification: kNN, logistic regression and SVM
   - Nonlinear models: kernel methods
3. Scalability issues

Motivation



- Unsupervised learning
- Discover groups
- Reduce dimension

Motivation



- Unsupervised learning
- Discover groups
- Reduce dimension

k-means definition

- Training set $\mathcal{S} = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$
- Given k, find $C = (C_1, \ldots, C_n) \in \{1, k\}^n$ that solves

$$\min_C \sum_{i=1}^{n} \| x_i - \mu_{C_i} \|^2$$

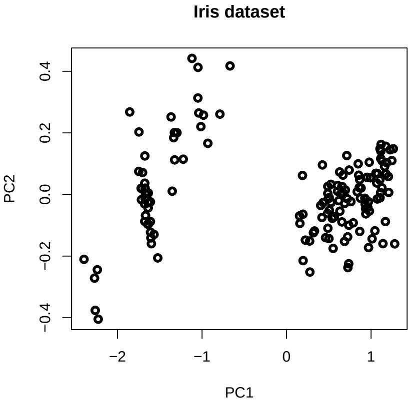
  where is the barycentre of data in class i.
- This is an NP-hard problem. k-means finds an approximate solution by iterating
  1. Assignment step: fix μ, optimize C

$$\forall i = 1, \ldots, n, \quad C_i \leftarrow \arg \min_{c \in \{1,\ldots,k\}} \| x_i - \mu_c \|$$

  2. Update step

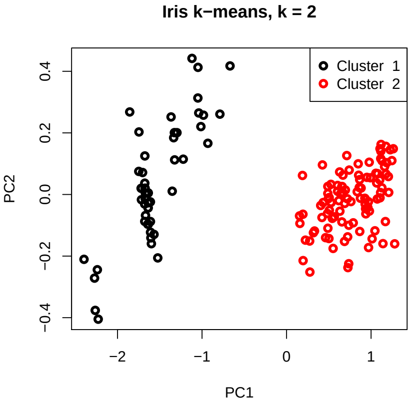
$$\forall i = 1, \ldots, k, \quad \mu_i \leftarrow \frac{1}{|C_i|} \sum_{j : C_j = i} x_j$$

k-means example



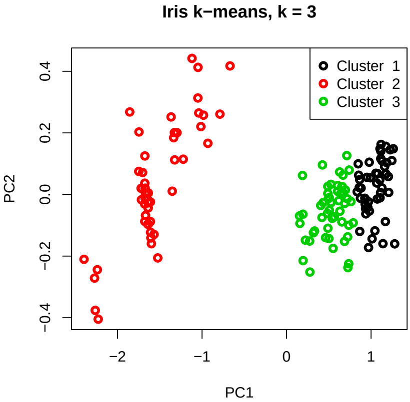

```
> irisCluster <- kmeans(log(iris[, 1:4]), 3, nstart = 20)
> table(irisCluster$cluster, iris$Species)

    setosa versicolor virginica
  1      0         48         4
  2     50          0         0
  3      0          2        46
```

k-means example



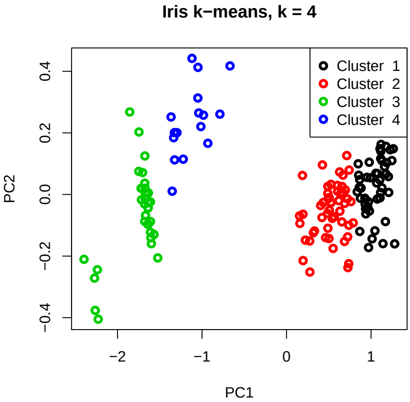

```
> irisCluster <- kmeans(log(iris[, 1:4]), 3, nstart = 20)
> table(irisCluster$cluster, iris$Species)

    setosa versicolor virginica
  1      0         48         4
  2     50          0         0
  3      0          2        46
```

k-means example



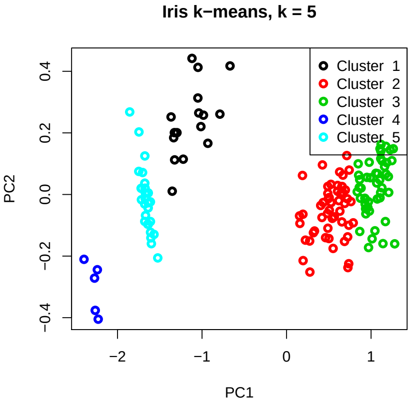

```
> irisCluster <- kmeans(log(iris[, 1:4]), 3, nstart = 20)
> table(irisCluster$cluster, iris$Species)

    setosa versicolor virginica
  1      0         48         4
  2     50          0         0
  3      0          2        46
```

k-means example




```
> irisCluster <- kmeans(log(iris[, 1:4]), 3, nstart = 20)
> table(irisCluster$cluster, iris$Species)

    setosa versicolor virginica
  1      0         48         4
  2     50          0         0
  3      0          2        46
```

k-means example



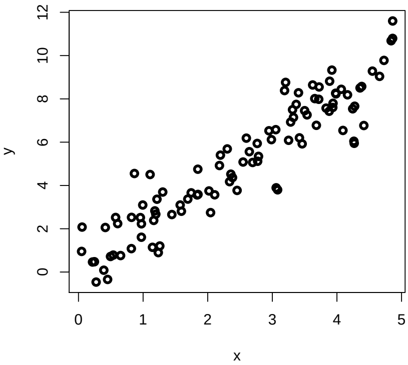

```
> irisCluster <- kmeans(log(iris[, 1:4]), 3, nstart = 20)
> table(irisCluster$cluster, iris$Species)

    setosa versicolor virginica
  1      0         48         4
  2     50          0         0
  3      0          2        46
```

k-means complexity

- Each update step: $O(nd)$
- Each assignment step: $O(ndk)$

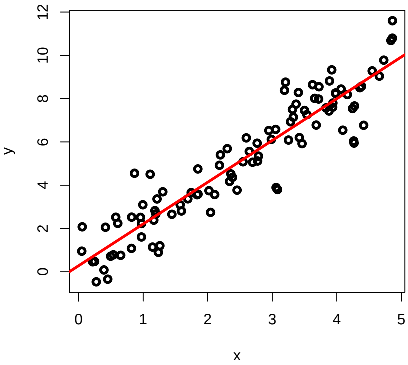
Outline

Motivation



- Predict a continuous output $Y \in \mathbb{R}$ from an input $X \in \mathbb{R}^d$

Motivation



- Predict a continuous output $Y \in \mathbb{R}$ from an input $X \in \mathbb{R}^d$

Ridge regression (Hoerl and Kennard, 1970)

- Training set $\mathcal{S} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subset \mathbb{R}^d \times \mathbb{R}$
- Fit a linear function:
$$f_\beta(x) = \beta^\top x$$
- Goodness of fit measured by residual sum of squares:
$$RSS(\beta) = \sum_{i=1}^{n} (y_i - f_\beta(x_i))^2$$
- Ridge regression minimizes the regularized RSS:
$$\min_\beta RSS(\beta) + \lambda \sum_{i=1}^{d} \beta_i^2$$

Solution

- Let $X = (x_1, \ldots, x_n)$ the $n \times p$ data matrix, and $Y = (y_1, \ldots, y_n)^\top \in \mathbb{R}^p$ the response vector.

Solution

- Let $X = (x_1, \ldots, x_n)$ the $n \times p$ data matrix, and $Y = (y_1, \ldots, y_n)^\top \in \mathbb{R}^p$ the response vector.
- The penalized risk can be written in matrix form:

$$
\begin{aligned}
R(\beta) + \lambda\Omega(\beta) &= \frac{1}{n}\sum_{i=1}^{n}\left(f_\beta\left(x_i\right) - x_i\right)^2 + \lambda\sum_{i=1}^{p}\beta_i^2 \\
&= \frac{1}{n}\left(Y - X\beta\right)^\top\left(Y - X\beta\right) + \lambda\beta^\top\beta\,.
\end{aligned}
$$

Solution

- Let $X = (x_1, \ldots, x_n)$ the $n \times p$ data matrix, and $Y = (y_1, \ldots, y_n)^\top \in \mathbb{R}^p$ the response vector.
- The penalized risk can be written in matrix form:

$$R(\beta) + \lambda\Omega(\beta) = \frac{1}{n}\sum_{i=1}^{n}\left(f_\beta\left(x_i\right) - x_i\right)^2 + \lambda\sum_{i=1}^{p}\beta_i^2$$
$$= \frac{1}{n}\left(Y - X\beta\right)^\top\left(Y - X\beta\right) + \lambda\beta^\top\beta\,.$$

- Explicit minimizer:

$$\hat{\beta}_\lambda^{\text{ridge}} = \arg\min_{\beta\in\mathbb{R}^p}\left\{R(\beta) + \lambda\Omega(\beta)\right\} = \left(X^\top X + \lambda n I\right)^{-1} X^\top Y\,.$$

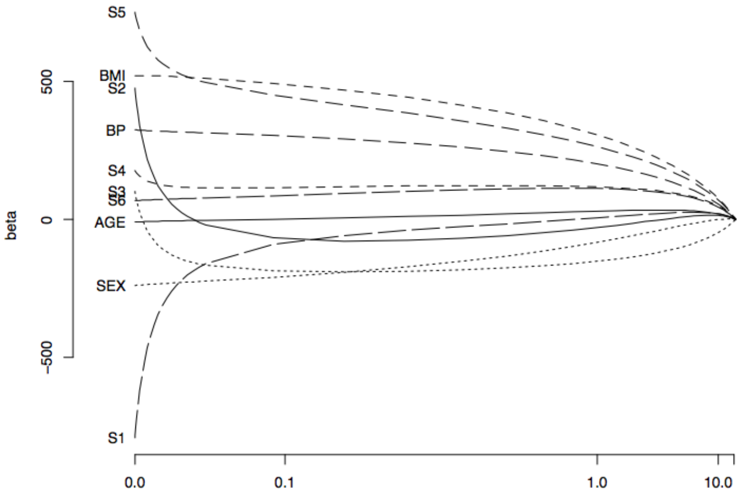
Limit cases

$$\hat{\beta}_\lambda^{\text{ridge}} = \left(X^\top X + \lambda n I\right)^{-1} X^\top Y$$

Corollary

- As $\lambda \to 0$, $\hat{\beta}_\lambda^{\text{ridge}} \to \hat{\beta}^{\text{OLS}}$ (low bias, high variance).
- As $\lambda \to +\infty$, $\hat{\beta}_\lambda^{\text{ridge}} \to 0$ (high bias, low variance).

Ridge regression example



(From Hastie et al., 2001)

Ridge regression with correlated features

Ridge regression is particularly useful in the presence of correlated features:

```
> library(MASS) # for the lm.ridge command
> x1 <- rnorm(20)
> x2 <- rnorm(20,mean=x1,sd=.01)
> y <- rnorm(20,mean=3+x1+x2)
> lm(y~x1+x2)$coef
(Intercept)          x1          x2
   3.070699   25.797872  -23.748019
> lm.ridge(y~x1+x2,lambda=1)
              x1        x2
3.066027 1.015862 0.956560
```

Ridge regression complexity

- Compute $X^\top X$: $O(nd^2)$
- Inverse $\left(X^\top X + \lambda I\right)$: $O(d^3)$

Computing $X^\top X$ is more expensive than inverting it when $n > d$!

Generalization: ℓ_2-regularized learning

- A general ℓ_2-penalized estimator is of the form

$$\min_{\beta} \left\{ R(\beta) + \lambda \|\beta\|_2^2 \right\} , \tag{1}$$

  where

$$R(\beta) = \frac{1}{n} \sum_{i=1}^{n} \ell(f_\beta(x_i), y_i)$$

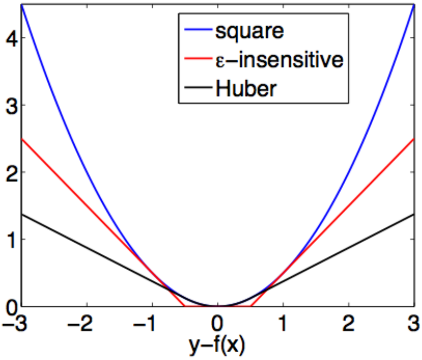
  for some general loss functions ℓ.
- Ridge regression corresponds to the particular loss

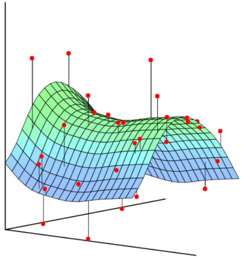
$$\ell(u, y) = (u - y)^2 .$$

- For general, convex losses, the problem (1) is strictly convex and has a unique global minimum, which can usually be found by numerical algorithms for convex optimization.
- Complexity: typically a factor more that ridge regression (e.g., iteratively approximate smooth losses by quadratic functions)

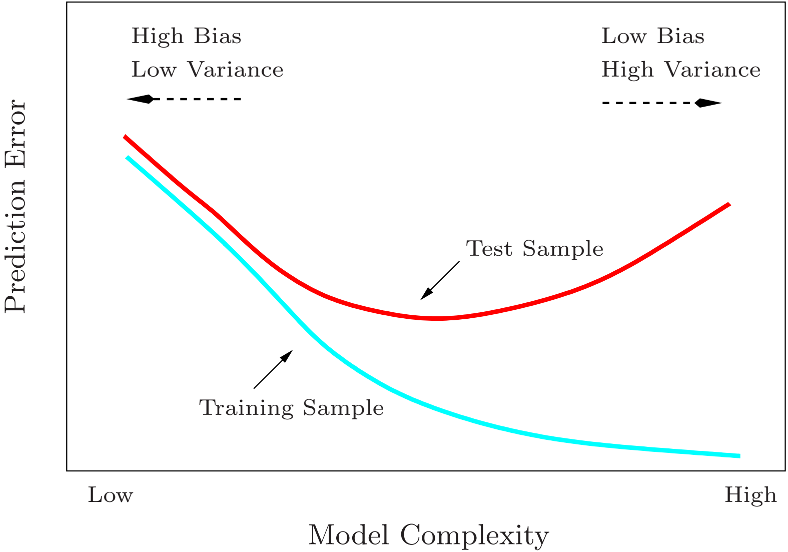
Losses for regression

- Square loss : $\ell(u, y) = (u - y)^2$
- ϵ-insensitive loss : $\ell(u, y) = (| u - y | - \epsilon)_+$
- Huber loss : mixed quadratic/linear





Choice of λ



Cross-validation

A simple and systematic procedure to estimate the risk (and to optimize the model's parameters)

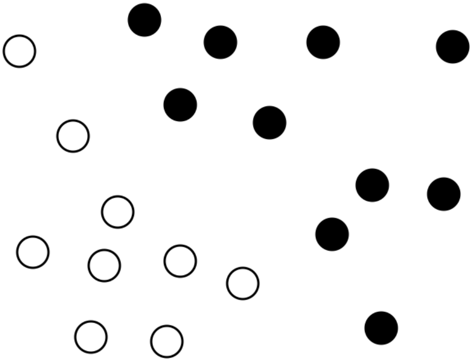
1. Randomly divide the training set (of size n) into K (almost) equal portions, each of size K/n
2. For each portion, fit the model with different parameters on the $K-1$ other groups and test its performance on the left-out group
3. Average performance over the K groups, and take the parameter with the smallest average performance.

Taking $K = 5$ or 10 is recommended as a good default choice.

Complexity: multiply by K

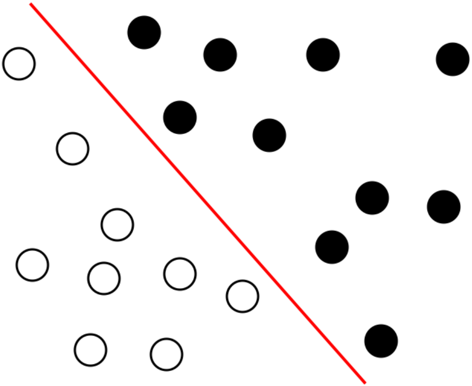
Outline

Motivation



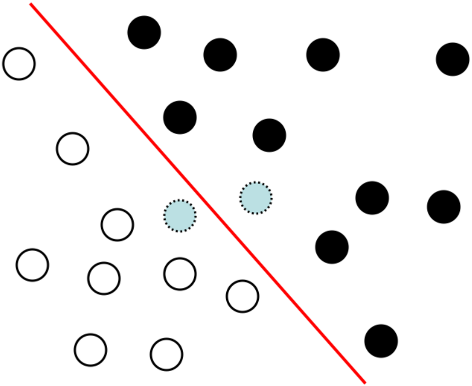
- Predict the category of a data
- 2 or more (sometimes many) categories

Motivation



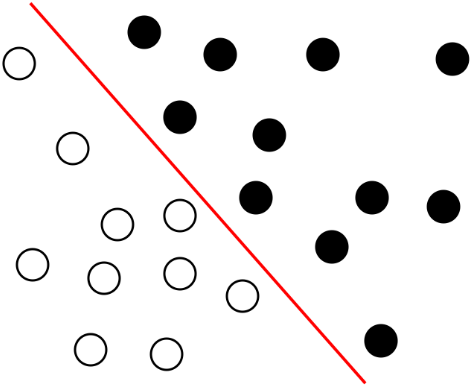
- Predict the category of a data
- 2 or more (sometimes many) categories

Motivation



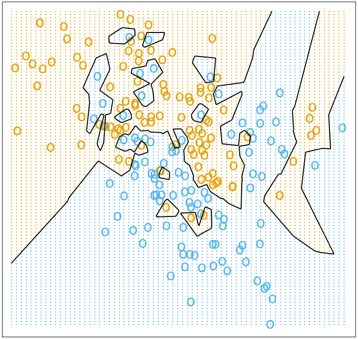
- Predict the category of a data
- 2 or more (sometimes many) categories

Motivation



- Predict the category of a data
- 2 or more (sometimes many) categories

k-nearest neigbors (kNN)



(Hastie et al. The elements of statistical learning. Springer, 2001.)

- Training set $\mathcal{S} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subset \mathbb{R}^d \times \{-1, 1\}$
- No training
- Given a new point $x \in \mathbb{R}^d$, predict the majority class among its k nearest neighbors (take k odd)

kNN properties

Uniform Bayes consistency (Stone, 1977)

- Take $k = \sqrt{n}$ (for example)
- Let P be any distribution over (X, Y) pairs
- Assume training data are random pairs sampled i.i.d. according to P
- Then the k-NN classifier $\hat{f}_n$ satisfies almost surely:

$$\lim_{n \to +\infty} P(\hat{f}(X) \neq Y) = \inf_{\text{f measurable}} P(f(X) \neq Y)$$

But "no free lunch":

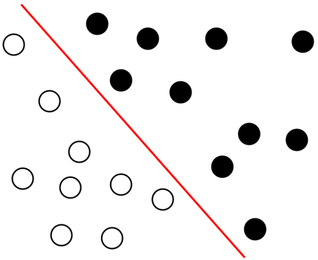
- The speed of convergence to the best classifier can be arbitrarily slow

kNN complexity

Complexity:

- Memory: storing X is $O(nd)$
- Training time: 0 (the best!)
- Prediction: $O(nd)$ for each test point (outch!)

Linear models for classification



- Training set $\mathcal{S} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subset \mathbb{R}^d \times \{-1, 1\}$
- Fit a linear function

$$f_\beta(x) = \beta^\top x$$

- The prediction on a new point $x \in \mathbb{R}^d$ is:

$$\begin{cases} +1 & \text{if } f_\beta(x) > 0\,, \\ -1 & \text{otherwise.} \end{cases}$$

The 0/1 loss

- The 0/1 loss measures if a prediction is correct or not:

$$\ell_{0/1}\left(f(x), y\right) = \mathbf{1}\left(yf(x) < 0\right) = \begin{cases} 0 & \text{if } y = sign\left(f(x)\right) \\ 1 & \text{otherwise.} \end{cases}$$

- It is them tempting to learn $f_\beta(x) = \beta^\top x$ by solving:

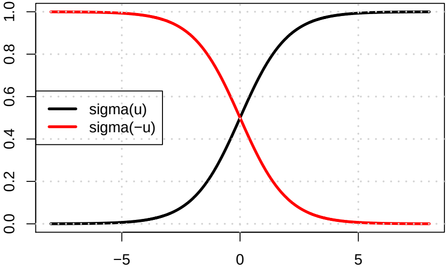
$$\min_{\beta \in \mathbb{R}^p} \underbrace{\frac{1}{n}\sum_{i=1}^{n} \ell_{0/1}\left(f_\beta\left(x_i\right), y_i\right)}_{\text{misclassification rate}} + \underbrace{\lambda \| \beta \|_2^2}_{\text{regularization}}$$

- However:
  - The problem is non-smooth, and typically NP-hard to solve
  - The regularization has no effect since the 0/1 loss is invariant by scaling of β
  - In fact, no function achieves the minimum when $\lambda > 0$ (*why?*)

The logistic loss

- An alternative is to define a probabilistic model of y parametrized by $f(x)$, e.g.:

$$\forall y \in \{-1, 1\}, \quad p(y \mid f(x)) = \frac{1}{1 + e^{-yf(x)}} = \sigma(yf(x))$$



- The logistic loss is the negative conditional likelihood:

$$\ell_{logistic}(f(x), y) = -\ln p(y \mid f(x)) = \ln\left(1 + e^{-yf(x)}\right)$$

Ridge logistic regression (Le Cessie and van Houwelingen, 1992)

$$\min_{\beta \in \mathbb{R}^p} J(\beta) = \frac{1}{n} \sum_{i=1}^{n} \ln\left(1 + e^{-y_i \beta^\top x_i}\right) + \lambda \|\beta\|_2^2$$

- Can be interpreted as a regularized conditional maximum likelihood estimator
- No explicit solution, but smooth convex optimization problem that can be solved numerically

Solving ridge logistic regression

$$\min_{\beta} J(\beta) = \frac{1}{n}\sum_{i=1}^{n} \ln\left(1 + e^{-y_i\beta^\top x_i}\right) + \lambda\|\beta\|_2^2$$

No explicit solution, but convex problem with:

$$\begin{aligned}
\nabla_\beta J(\beta) &= -\frac{1}{n}\sum_{i=1}^{n} \frac{y_i x_i}{1 + e^{y_i\beta^\top x_i}} + 2\lambda\beta \\
&= -\frac{1}{n}\sum_{i=1}^{n} y_i \left[1 - P_\beta(y_i \,|\, x_i)\right] x_i + 2\lambda\beta \\
\nabla_\beta^2 J(\beta) &= \frac{1}{n}\sum_{i=1}^{n} \frac{x_i x_i^\top e^{y_i\beta^\top x_i}}{\left(1 + e^{y_i\beta^\top x_i}\right)^2} + 2\lambda I \\
&= \frac{1}{n}\sum_{i=1}^{n} P_\beta(1 \,|\, x_i)\left(1 - P_\beta(1 \,|\, x_i)\right) x_i x_i^\top + 2\lambda I
\end{aligned}$$

Solving ridge logistic regression (cont.)

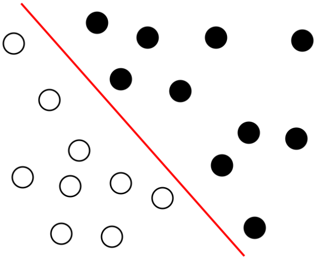
$$\min_{\beta} J(\beta) = \frac{1}{n} \sum_{i=1}^{n} \ln \left(1 + e^{-y_i \beta^\top x_i} \right) + \lambda \|\beta\|_2^2$$

- The solution can then be found by Newton-Raphson iterations:

$$\beta^{new} \leftarrow \beta^{old} - \left[\nabla_\beta^2 J \left(\beta^{old} \right) \right]^{-1} \nabla_\beta J \left(\beta^{old} \right) .$$

- Each step is equivalent to solving a weighted ridge regression problem (*left as exercise*)
- This method is therefore called iteratively reweighted least squares (IRLS).
- Complexity $O(iterations * (nd^2 + d^3))$

Large-margin classifiers



- For any $f : \mathbb{R}^d \to \mathbb{R}$, the margin of f on an (x, y) pair is

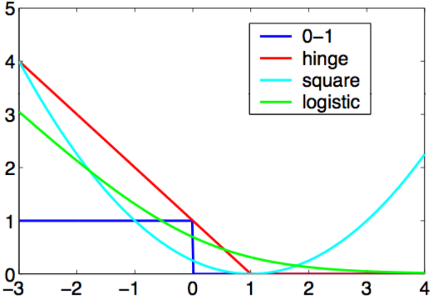
$$yf(x)$$

- Large-margin classifiers fit a classifier by maximizing the margins on the training set:

$$\min_{\beta} \sum_{i=1}^{n} \varphi\left(y_i f_\beta(x_i)\right) + \lambda \beta^\top \beta$$

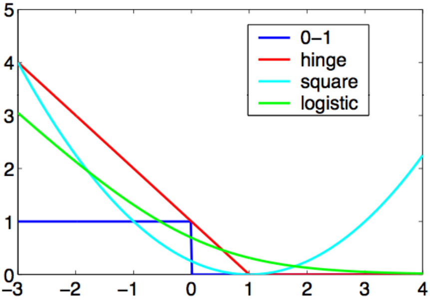
for a convex, non-increasing function $\varphi : \mathbb{R} \to \mathbb{R}+$

Loss function examples



Loss	Method	$\varphi(u)$
0-1	none	$1(u \leq 0)$
Hinge	Support vector machine (SVM)	$\max(1-u, 0)$
Logistic	Logistic regression	$\log(1+e^{-u})$
Square	Ridge regression	$(1-u)^2$
Exponential	Boosting	e^{-u}

Which φ?



- Computation
  - φ convex means we need to solve a convex optimization problem.
  - A "good" φ may be one which allows for fast optimization
- Theory
  - Most φ lead to consistent estimators (see next slides)
  - Some may be more efficient

A tiny bit of learning theory

Assumptions and notations

- Let $\mathbb{P}$ be an (unknown) distribution on $\mathcal{X} \times \mathcal{Y}$, and $\eta(x) = \mathbb{P}(Y = 1 \,|\, X = x)$ a measurable version of the conditional distribution of Y given X
- Assume the training set $\mathcal{S}_n = (X_i, Y_i)_{i=1,\ldots,n}$ are i.i.d. random variables according to $\mathbb{P}$.
- The risk of a classifier $f : \mathcal{X} \to \mathbb{R}$ is $R(f) = \mathbb{P}\left(sign(f(X)) \neq Y\right)$
- The Bayes risk is

$$R^* = \inf_{f \text{ measurable}} R(f)$$

  which is attained for $f^*(x) = \eta(x) - 1/2$
- The empirical risk of a classifier $f : \mathcal{X} \to \mathbb{R}$ is

$$R^n(f) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\left(sign(f(X_i)) \neq Y_i\right)$$

φ-risk

- Let the empirical φ-risk be the empirical risk optimized by a large-margin classifier:

$$R_\varphi^n(f) = \frac{1}{n}\sum_{i=1}^{n} \varphi\left(Y_i f(X_i)\right)$$

- It is the empirical version of the φ-risk

$$R_\varphi(f) = \mathbb{E}[\varphi\left(Yf(X)\right)]$$

- Can we hope to have a small risk $R(f)$ if we focus instead on the φ-risk $R_\varphi(f)$?

A small φ-risk ensures a small 0/1 risk

Theorem (?)

Let $\varphi : \mathbb{R} \to \mathbb{R}_+$ be convex, non-increasing, differentiable at 0 with $\varphi'(0) < 0$. Let $f : \mathcal{X} \to \mathbb{R}$ measurable such that

$$R_\varphi(f) = \min_{g \text{ measurable}} R_\varphi(g) = R_\varphi^* .$$

Then

$$R(f) = \min_{g \text{ measurable}} R(g) = R^* .$$

Remarks:

- This tells us that, if we know $\mathbb{P}$, then minimizing the φ-risk is a good idea even if our focus is on the classification error.
- The assumptions on φ can be relaxed; it works for the broader class of *classification-calibrated* loss functions (**?**).
- More generally, we can show that if $R_\varphi(f) - R_\varphi^*$ is small, then $R(f) - R^*$ is small too (**?**).

A small φ-risk ensures a small 0/1 risk

Proof sketch:
Condition on $X = x$:

$$R_\varphi(f \mid X = x) = \mathbb{E}\left[\varphi\left(Yf\left(X\right)\right) \mid X = x\right] = \eta(x)\varphi\left(f(x)\right) + \left(1 - \eta(x)\right)\varphi\left(-f(x)\right)$$
$$R_\varphi(-f \mid X = x) = \mathbb{E}\left[\varphi\left(-Yf\left(X\right)\right) \mid X = x\right] = \eta(x)\varphi\left(-f(x)\right) + \left(1 - \eta(x)\right)\varphi\left(f(x)\right)$$

Therefore:

$$R_\varphi(f \mid X = x) - R_\varphi(-f \mid X = x) = [2\eta(x) - 1] \times [\varphi\left(f(x)\right) - \varphi\left(-f(x)\right)]$$

This must be a.s. ≤ 0 because $R_\varphi(f) \leq R_\varphi(-f)$, which implies:

- if $\eta(x) > \frac{1}{2}$, $\varphi\left(f(x)\right) \leq \varphi\left(-f(x)\right) \implies f(x) \geq 0$
- if $\eta(x) < \frac{1}{2}$, $\varphi\left(f(x)\right) \geq \varphi\left(-f(x)\right) \implies f(x) \leq 0$

These inequalities are in fact strict thanks to the assumptions we made on φ (*left as exercice*). $\square$

SVM (Boser et al., 1992)

$$\min_{\beta\in\mathbb{R}^p} \quad \sum_{i=1}^{n} \max\left(0, 1 - y_i\beta^\top x_i\right) + \lambda\beta^\top\beta$$

- A non-smooth convex optimization problem (convex quadratic program)
- Equivalent to the dual problem

$$\max_{\alpha\in\mathbb{R}^n} 2\alpha^\top Y - \alpha^\top XX^\top\alpha \quad \text{s.t.} \quad 0 \le \mathbf{y}_i\alpha_i \le \frac{1}{2\lambda} \text{ for } i = 1, \ldots, n$$

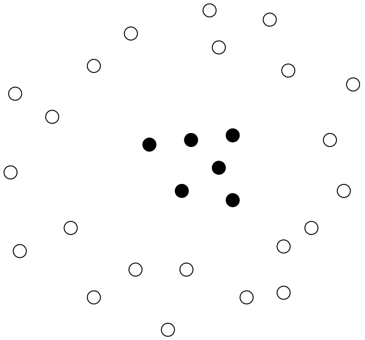
- The solution β^* of the primal is obtained from the solution α^* of the dual:

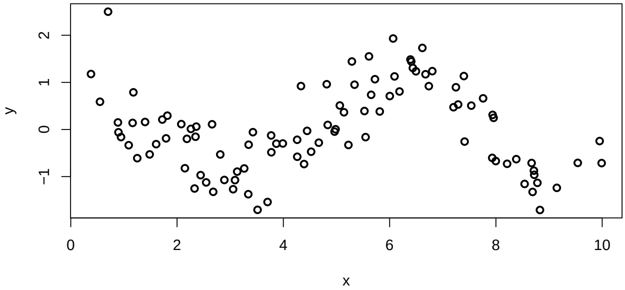
$$\beta^* = X^\top\alpha^* \qquad f_{\beta^*}(x) = (\beta^*)^\top x = (\alpha^*)^\top Xx$$

- Training complexity: $O(n^2)$ to store $XX^\top$, $O(n^3)$ to find α^*
- Prediction: $O(d)$ for $(\beta^*)^\top x$, $O(nd)$ for $(\alpha^*)^\top Xx$

Outline

Motivation





Model

- Learn a function $f : \mathbb{R}^d \to \mathbb{R}$ of the form

$$f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x)$$

- For a positive definite (p.d.) kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, such as

$$\begin{aligned}
\text{Linear} \quad & K(x, x') = x^\top x' \\
\text{Polynomial} \quad & K(x, x') = \left(x^\top x' + c\right)^p \\
\text{Gaussian} \quad & K(x, x') = \exp\left(-\frac{\| x - x' \|^2}{2\sigma^2}\right) \\
\text{Min/max} \quad & K(x, x') = \sum_{i=1}^{d} \frac{\min(|x_i|, |x_i'|)}{\max(|x_i|, |x_i'|)}
\end{aligned}$$

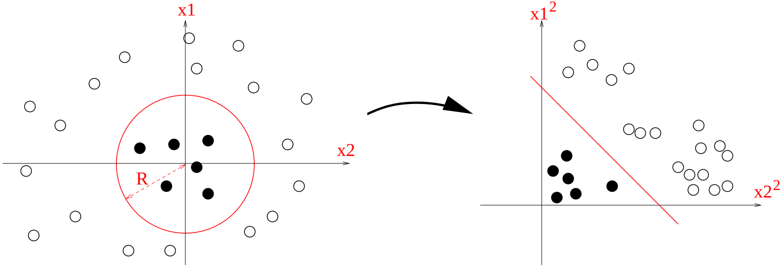
Feature space

- A function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a p.d. kernel if and only if there existe a mapping $\Phi : \mathbb{R}^d \to \mathbb{R}^D$, for some $D \in \mathbb{N} \cup \{+\infty\}$, such that

$$\forall x, x' \in \mathbb{R}^d, \quad K(x, x') = \Phi(x)^\top \Phi(x')$$

- Surprise: all functions in the previous slide are kernels! (sometime with $D = +\infty$)
- *Exercice: can you prove it?*

Example: polynomial kernel



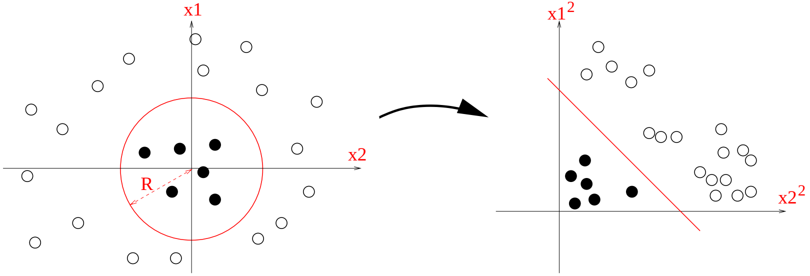
For $\vec{x} = (x_1, x_2)^\top \in \mathbb{R}^2$, let $\vec{\Phi}(\vec{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \in \mathbb{R}^3$:

$$
\begin{aligned}
K(\vec{x}, \vec{x}') &= x_1^2 x_1'^2 + 2x_1x_2x_1'x_2' + x_2^2 x_2'^2 \\
&= \left(x_1x_1' + x_2x_2'\right)^2 \\
&= \left(\vec{x}^\top \vec{x}'\right)^2 .
\end{aligned}
$$

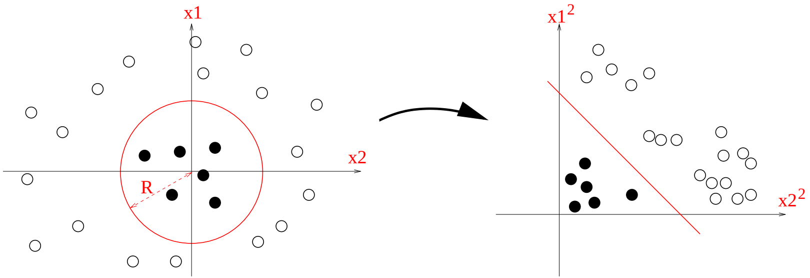
From $\alpha \in \mathbb{R}^n$ to $\beta \in \mathbb{R}^D$

$$\sum_{i=1}^{n} \alpha_i K(x_i, x) = \sum_{i=1}^{n} \alpha_i \Phi(x_i)^\top \Phi(x) = \beta^\top \Phi(x)$$

for $\beta = \sum_{i=1}^{n} \alpha_i \Phi(x_i)$.



Learning



- We can learn $f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x)$ by fitting a linear model $\beta^\top \Phi(x)$ in the feature space
- Example: ridge regression / logistic regression / SVM

$$\min_{\beta \in \mathbb{R}^D} \sum_{i=1}^{n} \ell(y_i, \beta^\top \Phi(x_i)) + \lambda \beta^\top \beta$$

- But D can be very large, even infinite...

Kernel tricks

- $K(x, x') = \Phi(x)^\top \Phi(x')$ can be quick to compute even if D is large (even infinite)
- For a set of training samples $\{x_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^d$ let K_n the $n \times n$ Gram matrix:

$$[K_n]_{ij} = K(x_i, x_j)$$

- For $\beta = \sum_{i=1}^{n} \alpha_i \Phi(x_i)$ we have

$$\beta^\top \Phi(x_i) = [K\alpha]_i \qquad \text{and} \qquad \beta^\top \beta = \alpha^\top K \alpha$$

- We can therefore solve the equivalent problem in $\alpha \in \mathbb{R}^n$

$$\min_{\alpha \in \mathbb{R}^n} \sum_{i=1}^{n} \ell(y_i, [K\alpha]_i) + \lambda \alpha^\top K \alpha$$

Example: kernel ridge regression (KRR)

$$\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} \left(y_i - \beta^\top \Phi(x_i) \right)^2 + \lambda \beta^\top \beta$$

- Solve in $\mathbb{R}^D$:

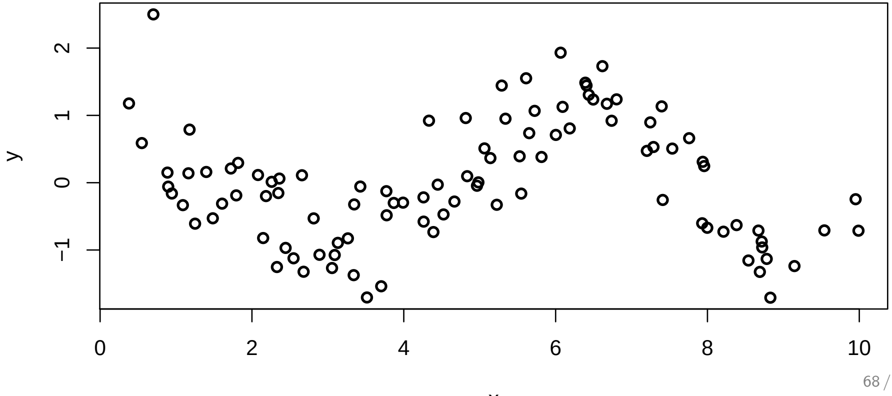
$$\hat{\beta} = \underbrace{\left(\Phi(X)^\top \Phi(X) + \lambda I \right)^{-1}}_{D \times D} \Phi(X)^\top Y$$

- Solve in $\mathbb{R}^n$:

$$\hat{\alpha} = \underbrace{(K + \lambda I)^{-1}}_{n \times n} Y$$

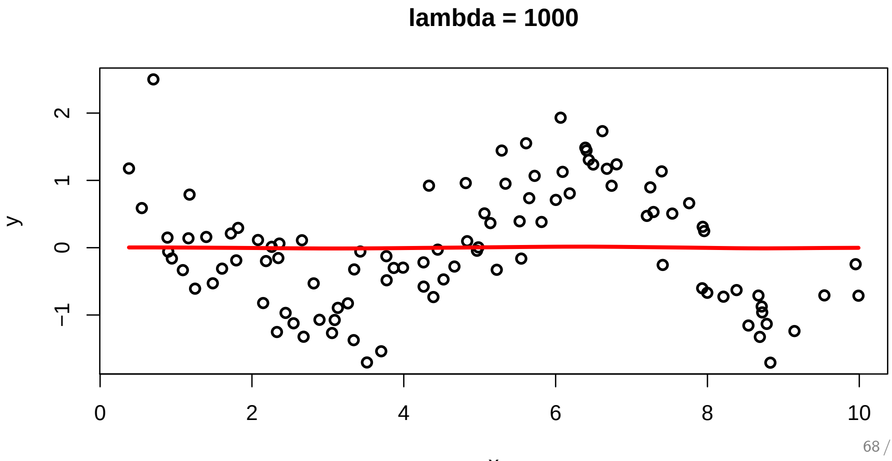
KRR with Gaussian RBF kernel

$$\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} \left(y_i - \beta^\top \Phi(x_i) \right)^2 + \lambda \beta^\top \beta \qquad K(x, x') = \exp\left(\frac{\| x - x' \|^2}{2\sigma^2} \right)$$



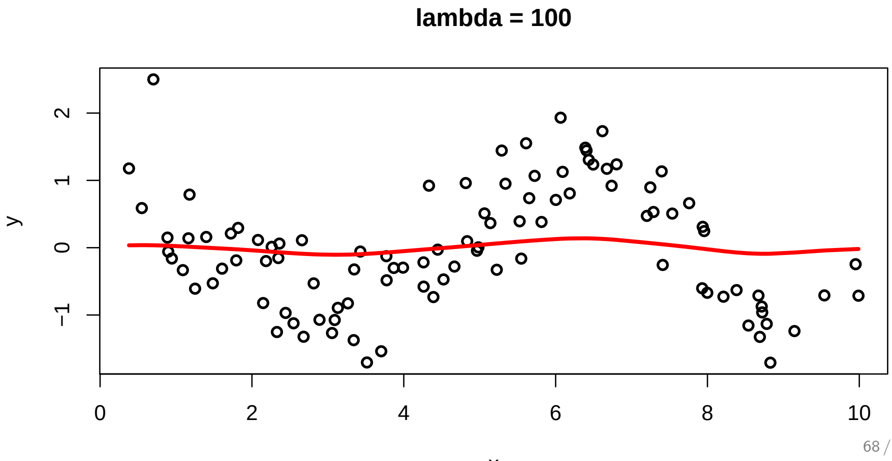
KRR with Gaussian RBF kernel

$$\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} \left(y_i - \beta^\top \Phi(x_i) \right)^2 + \lambda \beta^\top \beta \qquad K(x, x') = \exp\left(\frac{\| x - x' \|^2}{2\sigma^2} \right)$$



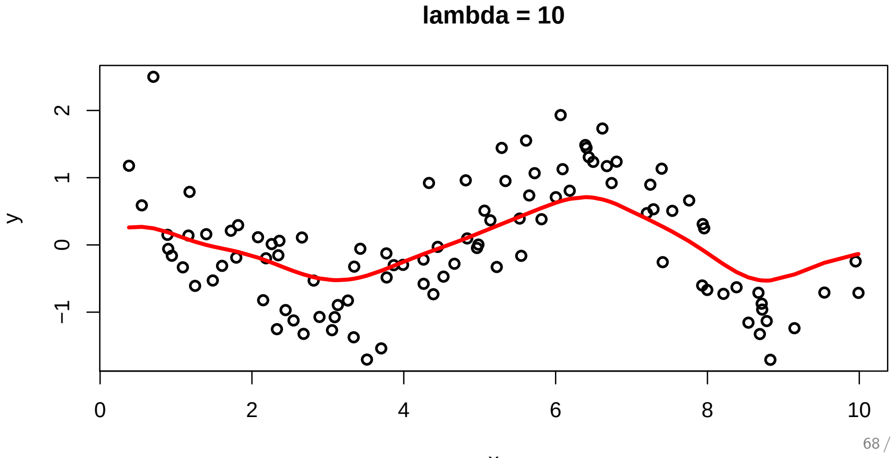
KRR with Gaussian RBF kernel

$$\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} \left(y_i - \beta^\top \Phi(x_i) \right)^2 + \lambda \beta^\top \beta \qquad K(x, x') = \exp\left(\frac{\| x - x' \|^2}{2\sigma^2} \right)$$



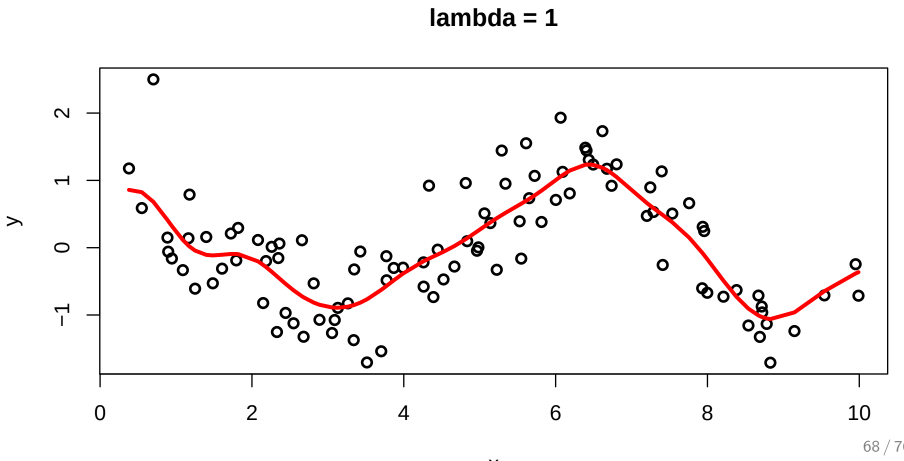
KRR with Gaussian RBF kernel

$$\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} \left(y_i - \beta^\top \Phi(x_i) \right)^2 + \lambda \beta^\top \beta \qquad K(x, x') = \exp\left(\frac{\| x - x' \|^2}{2\sigma^2} \right)$$



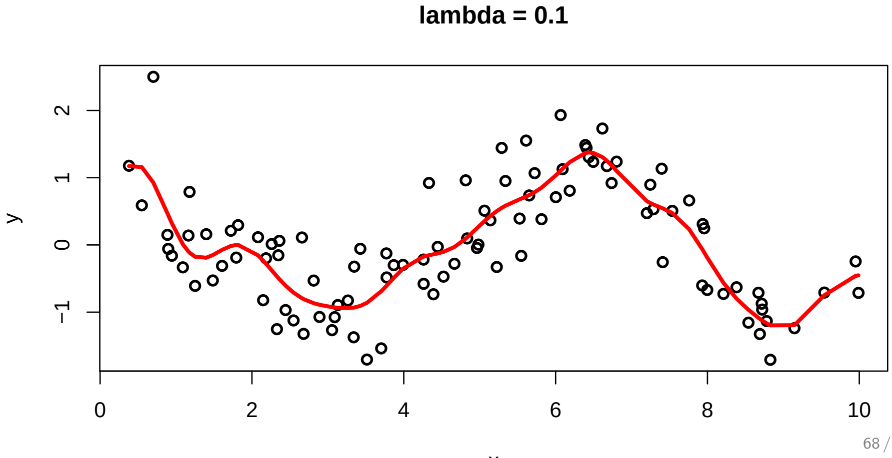
KRR with Gaussian RBF kernel

$$\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} \left(y_i - \beta^\top \Phi(x_i) \right)^2 + \lambda \beta^\top \beta \qquad K(x, x') = \exp\left(\frac{\| x - x' \|^2}{2\sigma^2} \right)$$



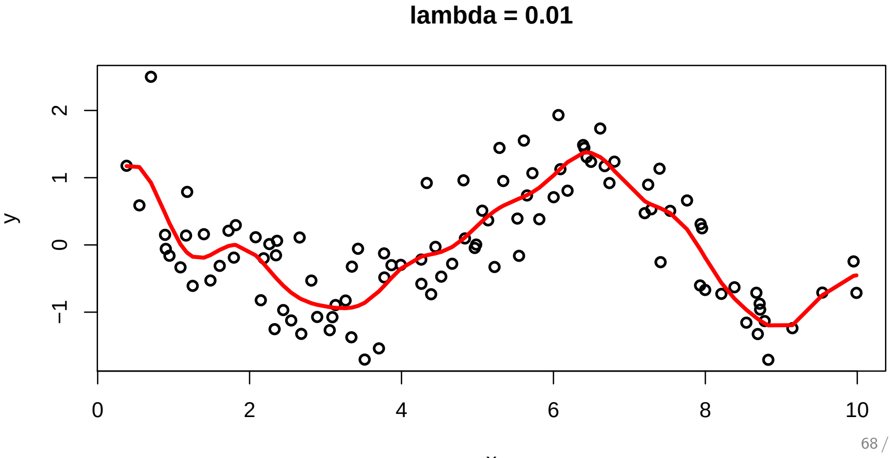
KRR with Gaussian RBF kernel

$$\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} \left(y_i - \beta^\top \Phi(x_i)\right)^2 + \lambda \beta^\top \beta \qquad K(x, x') = \exp\left(\frac{\| x - x' \|^2}{2\sigma^2}\right)$$



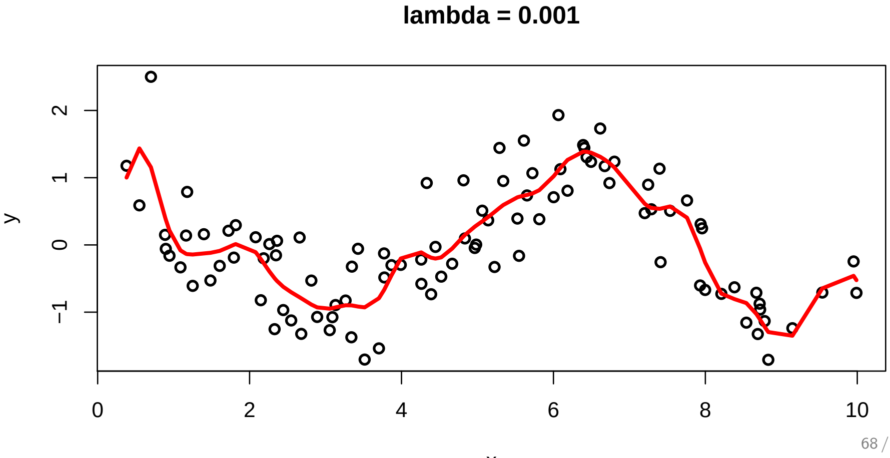
KRR with Gaussian RBF kernel

$$\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} \left(y_i - \beta^\top \Phi(x_i) \right)^2 + \lambda \beta^\top \beta \qquad K(x, x') = \exp\left(\frac{\| x - x' \|^2}{2\sigma^2} \right)$$



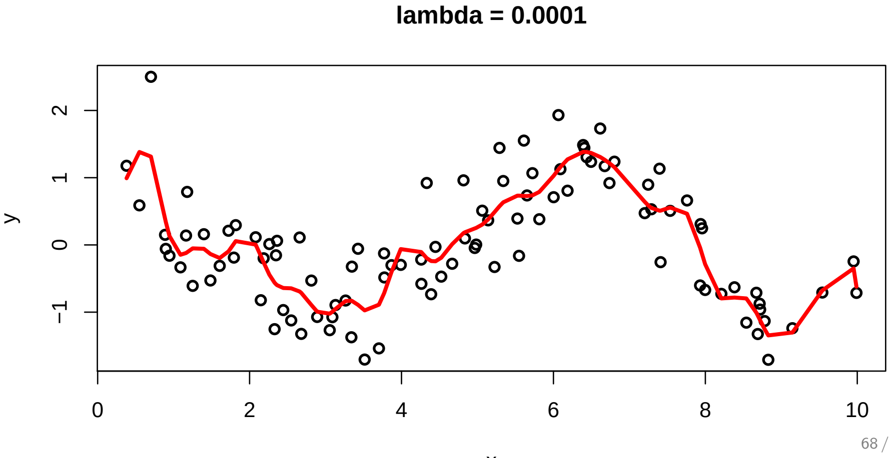
KRR with Gaussian RBF kernel

$$\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} \left(y_i - \beta^\top \Phi(x_i) \right)^2 + \lambda \beta^\top \beta \qquad K(x, x') = \exp\left(\frac{\| x - x' \|^2}{2\sigma^2} \right)$$



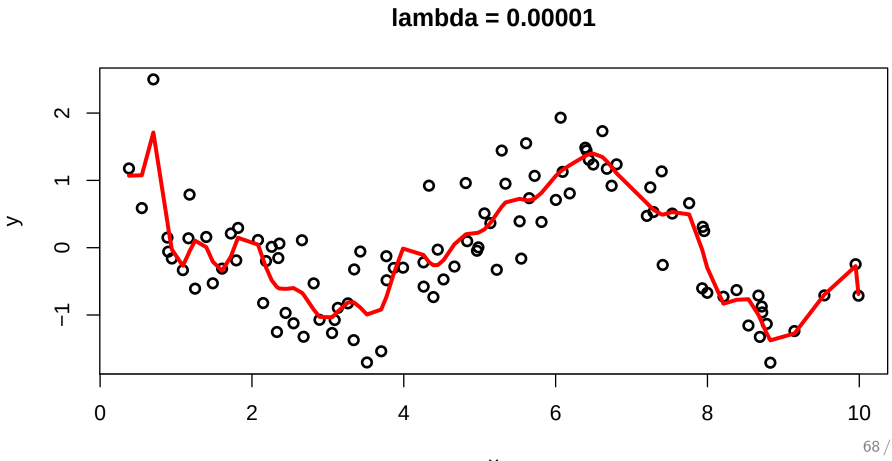
KRR with Gaussian RBF kernel

$$\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} \left(y_i - \beta^\top \Phi(x_i) \right)^2 + \lambda \beta^\top \beta \qquad K(x, x') = \exp\left(\frac{\| x - x' \|^2}{2\sigma^2} \right)$$



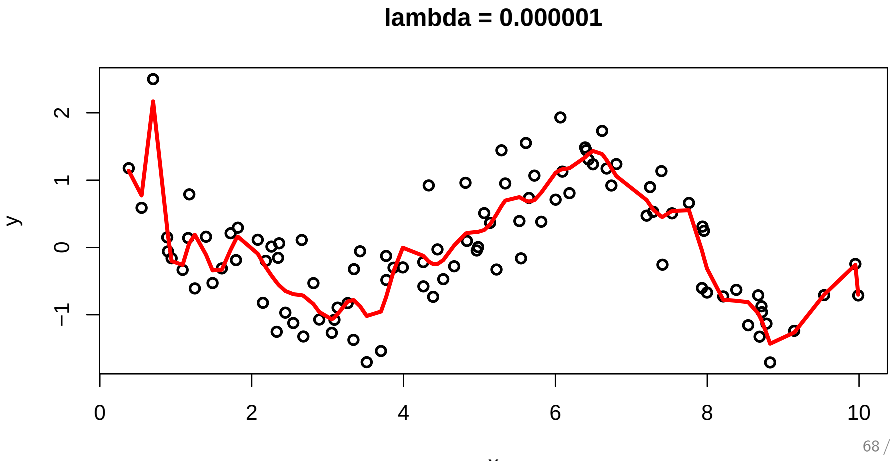
KRR with Gaussian RBF kernel

$$\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} \left(y_i - \beta^\top \Phi(x_i) \right)^2 + \lambda \beta^\top \beta \qquad K(x, x') = \exp\left(\frac{\| x - x' \|^2}{2\sigma^2} \right)$$



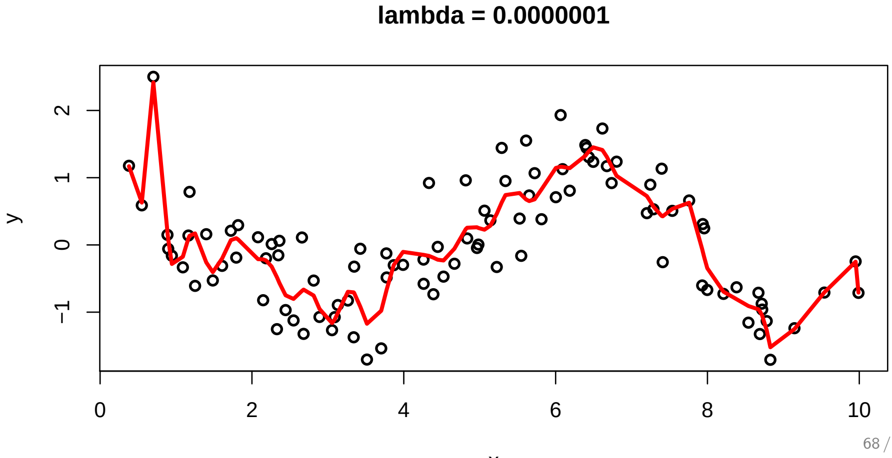
KRR with Gaussian RBF kernel

$$\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} \left(y_i - \beta^\top \Phi(x_i) \right)^2 + \lambda \beta^\top \beta \qquad K(x, x') = \exp\left(\frac{\| x - x' \|^2}{2\sigma^2} \right)$$

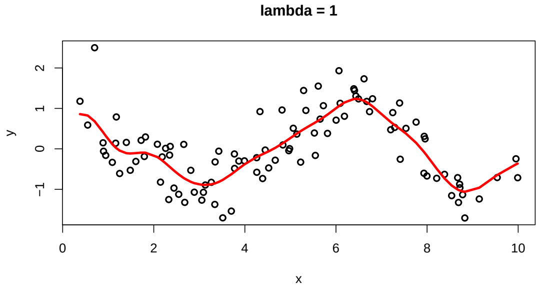


KRR with Gaussian RBF kernel

$$\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} \left(y_i - \beta^\top \Phi(x_i)\right)^2 + \lambda \beta^\top \beta \qquad K(x, x') = \exp\left(\frac{\| x - x' \|^2}{2\sigma^2}\right)$$



Complexity



- Compute K: $O(dn^2)$
- Store K: $O(n^2)$
- Solve α: $O(n^{2\sim 3})$
- Compute $f(x)$ for one x: $O(nd)$
- Unpractical for $n > 10 \sim 100k$

Outline

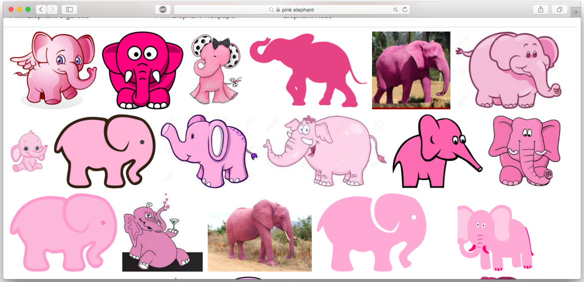
1. Introduction

2. Standard machine learning
   - Dimension reduction: PCA
   - Clustering: k-means
   - Regression: ridge regression
   - Classification: kNN, logistic regression and SVM
   - Nonlinear models: kernel methods

3. Scalability issues

What is "large-scale"?

- Data cannot fit in RAM
- Algorithm cannot run on a single machine in reasonable time (algorithm-dependent)
- Sometimes even $O(n)$ is too large! (e.g., nearest neighbor in a database of $O(B+)$ items)
- Many tasks / parameters (e.g., image categorization in $O(10M)$ classes)
- Streams of data



Things to worry about

- Training time (usually offline)
- Memory requirements
- Test time
- Complexities so far

Method	Memory	Training time	Test time
PCA	$O(d^2)$	$O(nd^2)$	$O(d)$
k-means	$O(nd)$	$O(ndk)$	$O(kd)$
Ridge regression	$O(d^2)$	$O(nd^2)$	$O(d)$
kNN	$O(nd)$	0	$O(nd)$
Logistic regression	$O(nd)$	$O(nd^2)$	$O(d)$
SVM, kernel methods	$O(n^2)$	$O(n^3)$	$O(nd)$

Techniques for large-scale ML

- Understand modern architecture, and how to distribute data / computation (cf C. Azencott)
- Trade optimization accuracy for speed (cf F. Bach)
- Know the tricks, eg, for deep learning (cf F. Moutarde)
- Randomization helps (cf friday)

References I

D. Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66(4):671–687, 2003. doi: 10.1016/S0022-0000(03)00025-4. URL http://dx.doi.org/10.1016/S0022-0000(03)00025-4.

N. Ailon and B. Chazelle. The fast Johnson-Lindenstrauss transform and approximate nearest neighbors. *SIAM J. Comput.*, 39(1):302–322, 2009. doi: 10.1137/060673096. URL http://dx.doi.org/10.1137/060673096.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th annual ACM workshop on Computational Learning Theory*, pages 144–152, New York, NY, USA, 1992. ACM Press. URL http://www.clopinet.com/isabelle/Papers/colt92.ps.Z.

L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Adv. Neural. Inform. Process Syst.*, volume 20, pages 161–168. Curran Associates, Inc., 2008. URL http://papers.nips.cc/paper/3323-the-tradeoffs-of-large-scale-learning.pdf.

A. Z. Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences*, pages 21–29, 1997. doi: 10.1109/SEQUEN.1997.666900. URL http://dx.doi.org/10.1109/SEQUEN.1997.666900.

M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, apr 1995. doi: 10.1137/S0097539793242618. URL http://dx.doi.org/10.1137/S0097539793242618.

References II

T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2001.

A. E. Hoerl and R. W. Kennard. Ridge regression : biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemp. Math.*, 26:189–206, 1984. doi: 10.1090/conm/026/737400. URL http://dx.doi.org/10.1090/conm/026/737400.

S. Le Cessie and J. C. van Houwelingen. Ridge estimators in logistic regression. *Appl. Statist.*, 41(1):191–201, 1992. URL http://www.jstor.org/stable/2347628.

P. Li and A. C. König. *b*-bit minwise hashing. In *WWW*, pages 671–680, Raleigh, NC, 2010.

P. Li, A. O., and C. hui Z. One permutation hashing. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 3113–3121. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4778-one-permutation-hashing.pdf.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Adv. Neural. Inform. Process Syst.*, volume 20, pages 1177–1184. Curran Associates, Inc., 2008. URL http://papers.nips.cc/paper/3182-random-features-for-large-scale-kernel-machines.pdf.

Q. Shi, J. Petterson, G. Dror, J. Langford, A. Smola, and S. Vishwanathan. Hash kernels for structured data. *Journal of Machine Learning Research*, 10:2615–2637, 2009.

References III

C. Stone. Consistent nonparametric regression. *Ann. Stat.*, 8:1348–1360, 1977. URL http://links.jstor.org/sici?sici=0090-5364%28197707%295%3A4%3C595%3ACNR%3E2.0.CO%3B2-O.