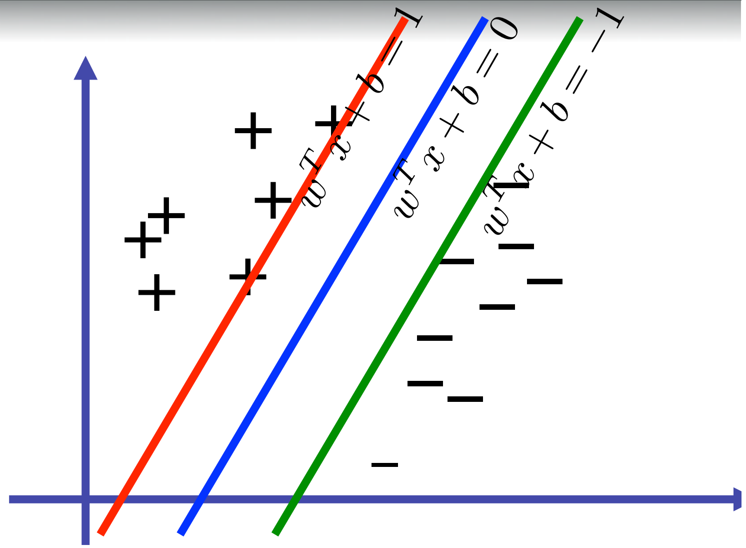
Course organization

- **Retrieval**
  - Given a query, find "most similar" item in a large data set
  - *Applications*: GoogleGoggles, Shazam, …
- **Supervised learning** (Classification, Regression)
  - Learn a concept (function mapping queries to labels)
  - *Applications*: Spam filtering, predicting price changes, …
- **Unsupervised learning** (Clustering, dimension reduction)
  - Identify clusters, "common patterns"; anomaly detection
  - *Applications*: Recommender systems, fraud detection, …
- **Learning with limited feedback**
  - Learn to optimize a function that's expensive to evaluate
  - *Applications*: Online advertising, opt. UI, learning rankings, …

Support Vector Machine

$$\min_{w,b} w^T w$$

$$\text{s.t.} y_i(w^T x_i + b) \geq 1$$



- How can we solve this optimization?
- What about local minima?

- This is a convex (quadratic) program

Generally: Online convex programming

- Input:
  - Feasible set $S \subseteq R^d$
  - Starting point $w_0 \in S$
- Each round t do
  - Receive convex function $f_t : S \to \mathbb{R}$
  - Incur loss $\ell_t = f_t(w_t)$
  - Update: $w_{t+1} = \mathrm{Proj}_S\big(w_t - \eta_t \nabla f_t(w_t)\big)$

Solve: $\min \sum_{t=1}^{N} f_t(x)$
s.t. $x \in S$

E.g., SVM

- Regret:

$$R_T = \Big(\sum_{t=1}^{T} \ell_t\Big) - \min_{w \in S} \sum_{t=1}^{T} f_t(w)$$

Regret for online convex programming

Theorem [Zinkevich '03]

Let $f_1, \ldots, f_T$ be an arbitrary sequence of convex functions with feasible set S

Set $\eta_t = 1/\sqrt{t}$

Then, the regret of online convex programming is bounded by

$$R_T \leq \frac{||S||^2\sqrt{T}}{2} + \left(\sqrt{T} - \frac{1}{2}\right)||\nabla f||^2$$

additional loss in accuracy due to online setting

$$\frac{R_T}{T} = O\left(\frac{\sqrt{T}}{T}\right) = O\left(\frac{1}{\sqrt{T}}\right) \to 0$$

More results on supervised learning

- Feature selection
- Dealing with multiple classes
- **Regression**
- Nonlinear methods

Regression

- So far, our goal was to predict a discrete label
- In many problems, we need to predict a real-valued output

$$y = f(x; w) + noise$$

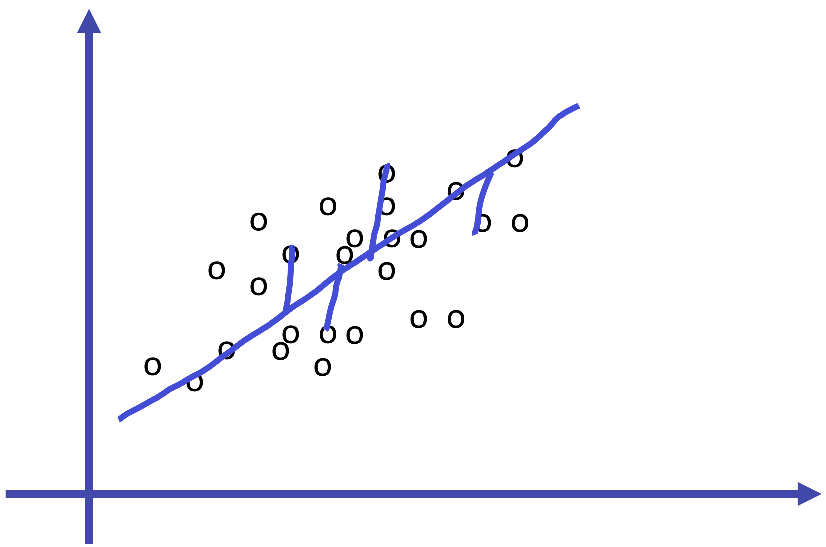
- E.g.:
  - Predict grade based on #homeworks solved
  - Predict flight delay at one airport given delays at other airports
  - …

Linear regression

- Given $(x_1, y_1), \ldots, (x_n, y_n)$

- Assume: $y_i = w^T x_i + noise$

- To optimize *w* need to quantify goodness of fit



Square loss

- Want to solve

$x_i \in \mathbb{R}^d, \quad X = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{pmatrix} \in \mathbb{R}^{N \times d}$

$$w^* = \arg\min_w \sum_{i=1}^{n} (y_i - w^T x_i)^2$$

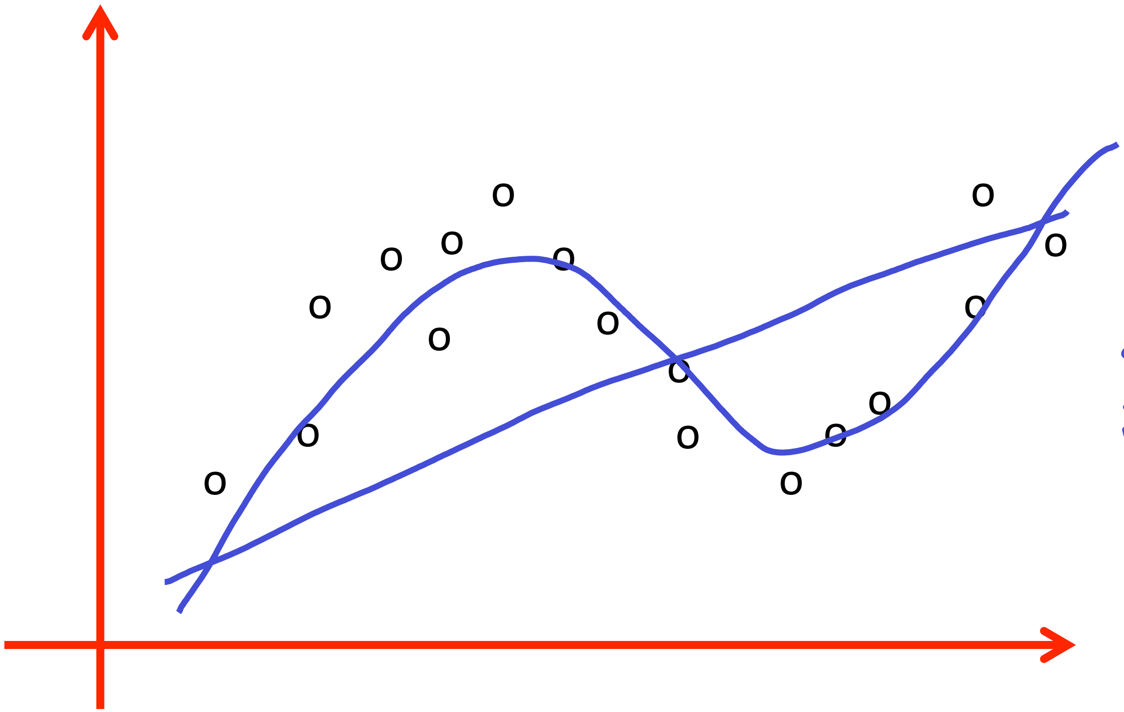
- Closed form solution: $$w^* = (X^T X)^{-1} X^T y$$

$\Omega(d^3)$, Also, maybe no unique solution

- Complexity?

- Intractable for large # of dimensions!
- Will see how we can efficiently compute with OCP!

Learning non-linear functions

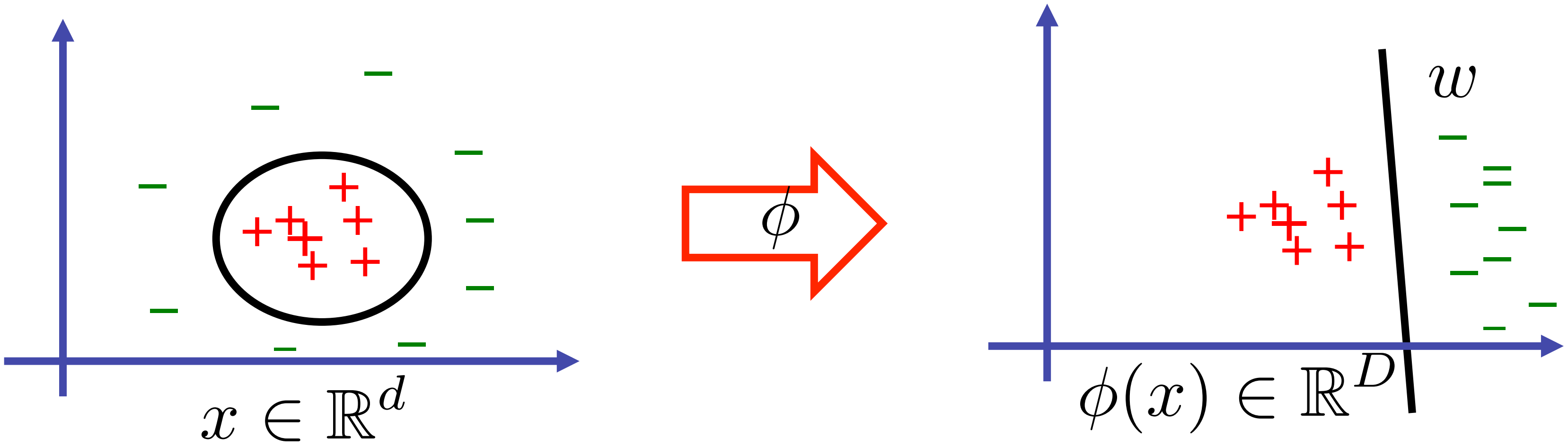


$\phi(x) = (1, x, x^2, x^3)$

still linear in w

Key insight: Can learn nonlinear functions using linear methods! Works for classification too!

Solving nonlinear problems



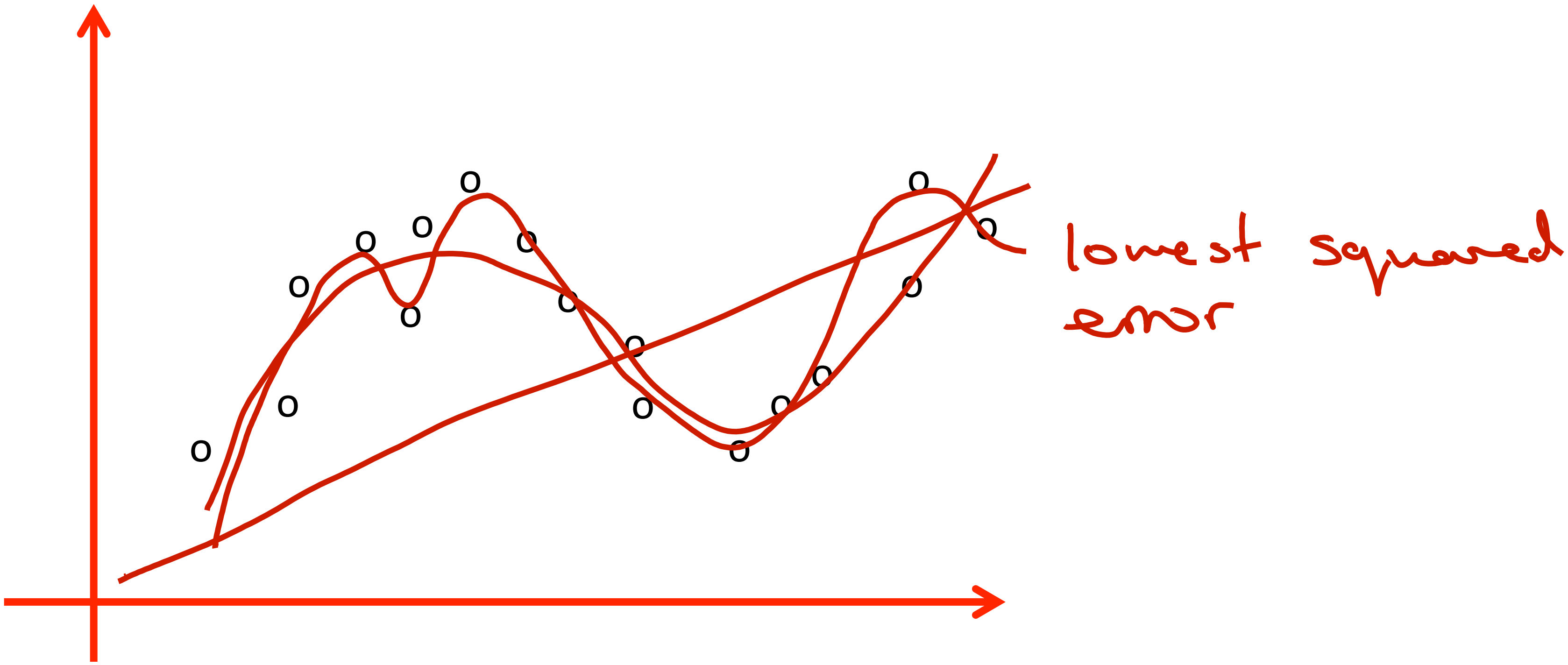
$$\text{sign}(w^T \phi(x))$$

$x \in \mathbb{R}^2$

$\phi(x) = (x_1^2, x_1 x_2, x_2^2) \in \mathbb{R}^3$

linearly separable in higher-dimensional feature space

Model selection in regression



Suppose we consider polynomials.
Which degree should we choose?

Regularization

- When learning complex / high dimensional functions, need to control the complexity of the model
  - In practice, this means ensuring that weights *w* are small
- This process is called regularization

Regularized regression

- Ridge regression:

$$w^* = \arg\min_w \lambda ||w||_2^2 + \sum_{i=1}^{n} (y_i - w^T x_i)^2$$

- Closed form solution:

ensures a unique solution

$$w^* = (X^T X + \lambda I)^{-1} X^T y$$

- Shrinks weights of ‘unimportant’ variables.

Regularized regression

- L1-regularized regression – "Lasso":

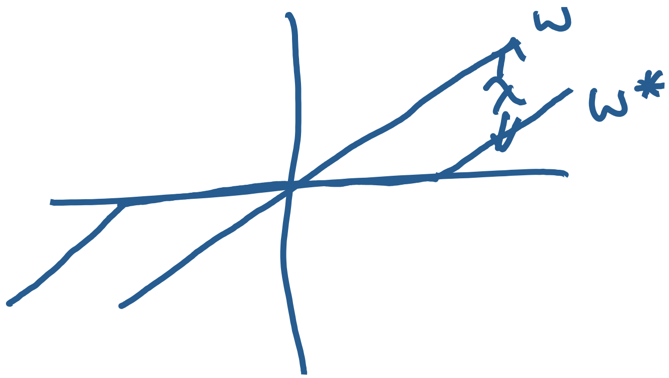
$$w^* = \arg\min_w \lambda ||w||_1 + \sum_{i=1}^{n} (y_i - w^T x_i)^2$$

- In general, no closed form solution.

least squares solution

If $X^T X = I$

$$w^* = \text{sign}(w)(|w| - \lambda)_+$$



More general loss functions

- A large fraction of methods in supervised learning can be reduced to optimization problems of the form

$$w^* = \arg\min_w \lambda||w|| + \sum_{i=1}^{n} \ell(y_i; x_i, w_i)$$

- Example loss functions
  - Hinge loss (SVM)
  - Multi-class hinge loss
  - Log loss (next homework!)
  - Square loss
  - ε-sensitive loss
  - …

Solving regularized learning problems

- Reduce to online convex programming:

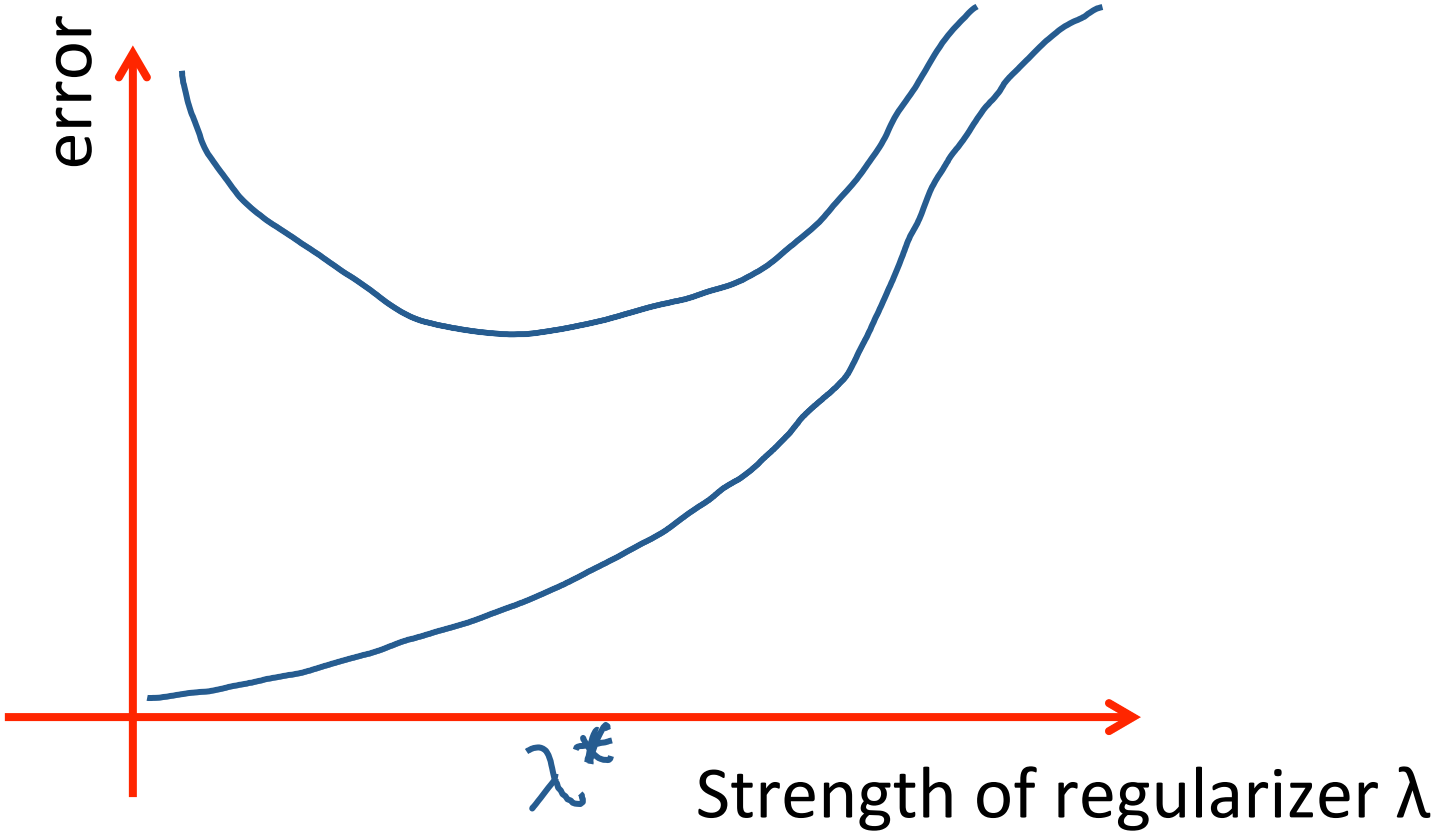
$$w_{t+1} = \mathrm{Proj}_S\big(x_t - \eta_t \nabla \ell(y_t; x_t, w_t)\big)$$

  - Gradient computation specific to loss function
  - Reprojection: Need to solve

$$\arg\min_{w' \in S} \|w' - w_t\|_2$$

Choosing the right regularizer

- How should we choose the regularization parameter?



Cross-validation

- May overfit if we optimize for fixed training set!
- Remedy: Cross-validation

D_1	D_2	…	D_k

- Split data set into k "folds"
- For each possible regularization parameter setting λ:
  - For i = 1:k
    - Train on all but i-th fold; calculate error E_i
  - Estimate generalization error for param. λ as $$\frac{1}{k}\sum_i E_i$$

Aside: Cross-validation

- How to choose k? 5, 10…

$$K = n(1 - \frac{1}{\log n - 1})$$

- Then cross-validation is equivalent to the Bayesian Information Criterion

$$BIC = -2\log \ell + m \log n$$

- CV penalises the *degrees of freedom*.
- These results only apply for *linear models* with *squared error loss.*

Aside: Dual formulation of SVM

- Primal form: $$\min_{w,b,\xi \geq 0} w^T w + C \sum_i \xi_i$$
$$\text{s.t.} y_i(w^T x_i + b) \geq 1 - \xi_i$$

Using Lagrange multipliers:

$$\min_{w,b,\xi \geq 0} \max_{\alpha \geq 0} w^T w + C \sum_i \xi_i - \sum_i \alpha_i [y_i(w^T x_i + b) - 1 + \xi_i] - \sum_i \lambda_i \xi_i$$

- Dual form:

$$\frac{\partial L}{\partial b} = 0 \rightarrow \sum_i \alpha_i y_i = 0$$

$$\frac{\partial L}{\partial w} = 0 \rightarrow w = \sum_i \alpha_i y_i x_i$$

$$\frac{\partial L}{\partial \xi_i} = 0 \rightarrow C - \alpha_i - \lambda_i = 0$$
$$0 \leq \alpha_i \leq C$$

$$\max_{0 \leq \alpha \leq C} \sum_i \alpha_i - \sum_{ij} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

Aside: The „Kernel Trick“

- Standard lesson in Machine Learning:
  - Can solve linear problem in feature space implicitly using inner products only

$x \in \mathbb{R}^2 \quad \phi(x) = (x_1^2, 2x_1x_2, x_2^2)$

- Example: Dual formulation of SVM

$\phi(x)^T \phi(x') = (x^T x')^2$

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$

$$\text{s.t. } 0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^{N} \alpha_i y_i = 0$$

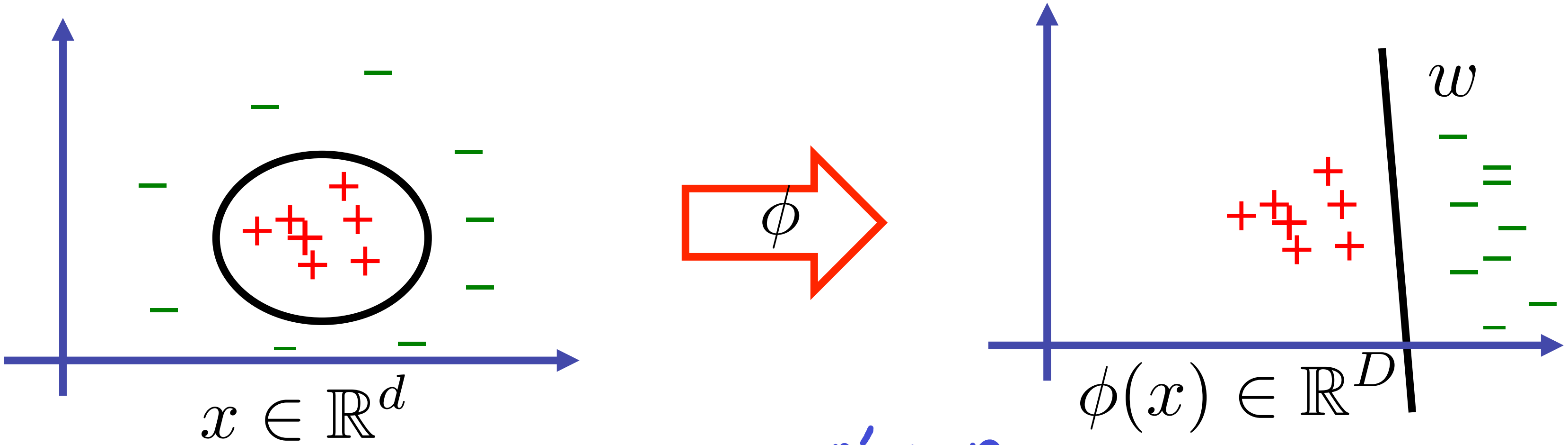
$$w = \sum_{i} \alpha_i y_i \phi(x_i)$$

have to maintain N α_i's

- Can we use this for large data??

„Inverse Kernel Trick“ [Rahimi, Recht, NIPS ‘07]

- Idea: **Explicitly** generate low-dim (nonlinear!) features



Construct $z(x) \in \mathbb{R}^{D'}$, $D' \ll D$

s.t. $z(x)^T z(x') \approx k(x, x')$

do linear classification / regression explicitly with $z(x)$

Random Fourier Features [RR NIPS'07]

Algorithm 1 Random Fourier Features.

Require: A positive definite shift-invariant kernel $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{x} - \mathbf{y})$.
Ensure: A randomized feature map $\mathbf{z}(\mathbf{x}) : \mathcal{R}^d \to \mathcal{R}^{2D}$ so that $\mathbf{z}(\mathbf{x})'\mathbf{z}(\mathbf{y}) \approx k(\mathbf{x} - \mathbf{y})$.

Compute the Fourier transform p of the kernel k: $p(\omega) = \frac{1}{2\pi} \int e^{-j\omega'\Delta} k(\Delta)\, d\Delta$.

Draw D iid samples $\omega_1, \cdots, \omega_D \in \mathcal{R}^d$ from p.

Let $\mathbf{z}(\mathbf{x}) \equiv \sqrt{\frac{1}{D}} \left[\cos(\omega_1'\mathbf{x}) \cdots \cos(\omega_D'\mathbf{x})\ \sin(\omega_1'\mathbf{x}) \cdots \sin(\omega_D'\mathbf{x}) \right]'$.

Kernel Name	$k(\Delta)$	$p(\omega)$
Gaussian	$e^{-\frac{\\|\Delta\\|_2^2}{2}}$	$(2\pi)^{-\frac{D}{2}} e^{-\frac{\\|\omega\\|_2^2}{2}}$
Laplacian	$e^{-\\|\Delta\\|_1}$	$\prod_d \frac{1}{\pi(1+\omega_d^2)}$
Cauchy	$\prod_d \frac{2}{1+\Delta_d^2}$	$e^{-\\|\Delta\\|_1}$

Performance of random features

Claim 1 (Uniform convergence of Fourier features). *Let $\mathcal{M}$ be a compact subset of $\mathcal{R}^d$ with diameter* $\mathrm{diam}(\mathcal{M})$. *Then, for the mapping* $\mathbf{z}$ *defined in Algorithm 1, we have*

$$\Pr\left[\sup_{x,y\in\mathcal{M}} |\mathbf{z}(\mathbf{x})'\mathbf{z}(\mathbf{y}) - k(\mathbf{x},\mathbf{y})| \geq \epsilon\right] \leq 2^8 \left(\frac{\sigma_p \,\mathrm{diam}(\mathcal{M})}{\epsilon}\right)^2 \exp\left(-\frac{D\epsilon^2}{4(d+2)}\right),$$

where $\sigma_p^2 \equiv E_p[\omega'\omega]$ *is the second moment of the Fourier transform of* k. *Further,* $\sup_{x,y\in\mathcal{M}} |\mathbf{z}(\mathbf{x})'\mathbf{z}(\mathbf{y}) - k(\mathbf{y},\mathbf{x})| \leq \epsilon$ *with any constant probability when* $D = \Omega\left(\frac{d}{\epsilon^2}\log\frac{\sigma_p \,\mathrm{diam}(\mathcal{M})}{\epsilon}\right)$.

- Solving linear SVM on explicit (random) features provably „almost the same“ as solving non-linear SVM

Performance of random features [RR '07]

Dataset	Fourier+LS	Binning+LS	CVM	Exact SVM
CPU regression 6500 instances 21 dims	3.6% 20 secs $D = 300$	5.3% 3 mins $P = 350$	5.5% 51 secs	11% 31 secs ASVM
Census regression 18,000 instances 119 dims	5% 36 secs $D = 500$	7.5% 19 mins $P = 30$	8.8% 7.5 mins	9% 13 mins SVMTorch
Adult classification 32,000 instances 123 dims	14.9% 9 secs $D = 500$	15.3% 1.5 mins $P = 30$	14.8% 73 mins	15.1% 7 mins $\text{SVM}^{\text{light}}$
Forest Cover classification 522,000 instances 54 dims	11.6% 71 mins $D = 5000$	2.2% 25 mins $P = 50$	2.3% 7.5 hrs	2.2% 44 hrs libSVM
KDDCUP99 (see footnote) classification 4,900,000 instances 127 dims	7.3% 1.5 min $D = 50$	7.3% 35 mins $P = 10$	6.2% (18%) 1.4 secs (20 secs)	8.3% < 1 s SVM+sampling

- Linear SVM/Regression on random features outperforms nonlinear methods

Summary

- Online convex programming is a natural approach to solve regularized learning problems
- Can be parallelized (to some extent)
- Flexible choice of loss function and regularizer gives rise to many useful methods
  - SVM
  - L1-SVM
  - Ridge regression
  - L1-regularized regression
  - Logistic regression (homework)
  - …
- Can even learn nonlinear functions!