Introduction to Machine Learning (IML)

LECTURE #4: CLASSIFICATION – SVM IN DEPTH

236756 – 2024 SPRING – TECHNION

LECTURER: NIR ROSENFELD

Today

- Last lecture of first part of course (!)
- More SVM (!!)
- (Mostly optimization; some modeling)
- Hard SVM (separable)
  - short recap
  - finish up
- Soft SVM (non-separable)
- Dual SVM and kernels (non-linearity via linearity)

Recap

- Def: margin of hyperplane w:

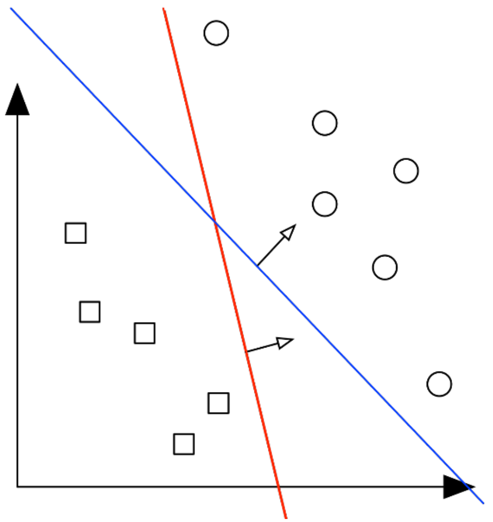
$$\text{margin}(w;S) = \min_{i\in[m]} \frac{|w^\top x_i|}{\|w\|} := \gamma(w;S)$$

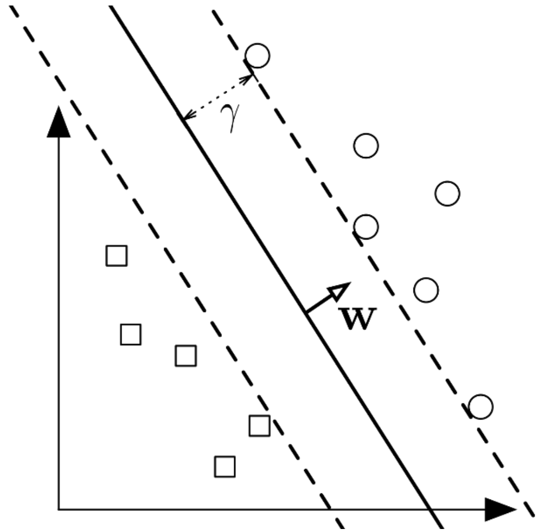
- **SVM looks for max margin classifier**

- Hard SVM works under **linear separability**:
  $\exists w$ s.t. $y_i \cdot w^\top x_i \geq 0 \; \forall i \in [m]$
  (homogeneous case)

- Hyperplanes are scale-invariant, and so are margins:

$$\gamma(w;S) = \gamma(\alpha w;S) \quad \forall \alpha \in \mathbb{R}$$





Discussion

- **Original max-margin motivation:** $\text{argmax}_w \, \gamma(w;S) \;\; \text{s.t.} \;\; y_i \cdot w^\top x_i \geq 0 \quad \forall i \in [m]$
- **Final Hard SVM objective:** $\boxed{w_{\text{H-SVM}} = \text{argmin}_w \|w\|_2^2 \;\; \text{s.t.} \;\; y_i \cdot w^\top x_i \geq 1 \;\; \forall i \in [m]}$
- **Main insight**: increasing margin ≡ reducing norm (for fixed margin of size=1)

what we *want*

constrain as =1

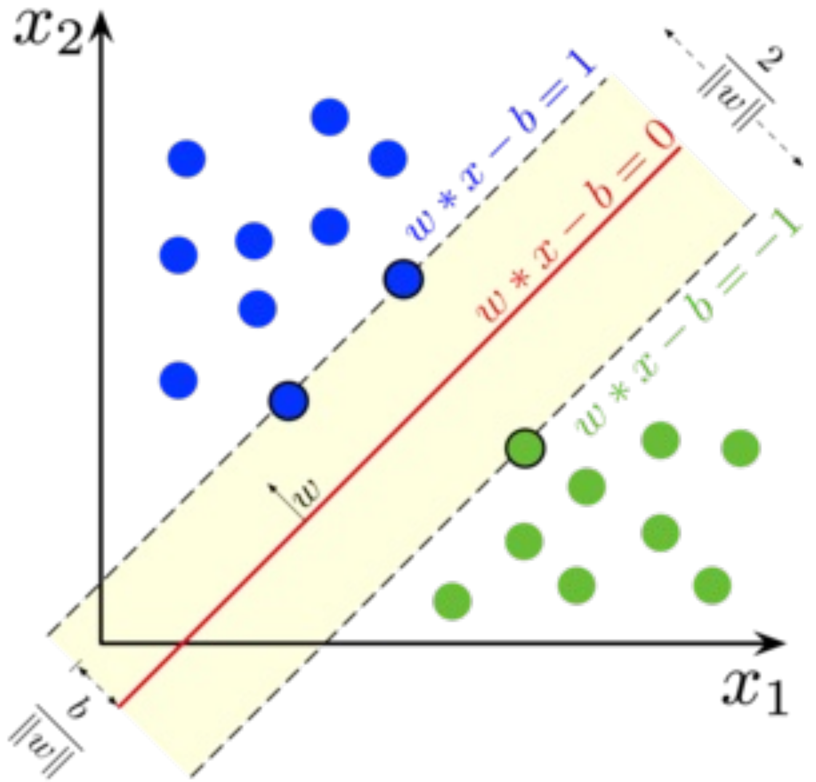
$$\gamma(w;S) = \min_{i \in [m]} \frac{|w^\top x_i|}{\|w\|_2} \equiv \frac{1}{\|w\|_2}$$

what we *do*

- Results in simple, convex objective with linear constraints (easy to optimize + unique solution)

Discussion

- **Final Hard SVM objective**: $\boxed{w_{\text{H-SVM}} = \text{argmin}_w \|w\|_2^2 \ \text{ s.t. } \ y_i \cdot w^\top x_i \geq 1 \ \ \forall i \in [m]}$
- **Claim**: at least one example (but possibly more) "touches" margin (=constraint is tight)
- Margin-touching examples are called "**support vectors**":
  - (hence the name - support vector *machines*)
  - removing "support" examples changes learned model
  - removing other examples does not
- These will pop up again later



Soft SVM

Hard SVM derivation

- Last week we derived a sequence of three optimization problems:

$$(1)\ \ w_1 = \mathrm{argmax}_w \frac{1}{\|w\|_2} \min_{i\in[m]} |w^\top x_i| \ \text{ s.t. }\ y_i \cdot w^\top x_i \geq 0 \qquad \forall i \in [m]$$

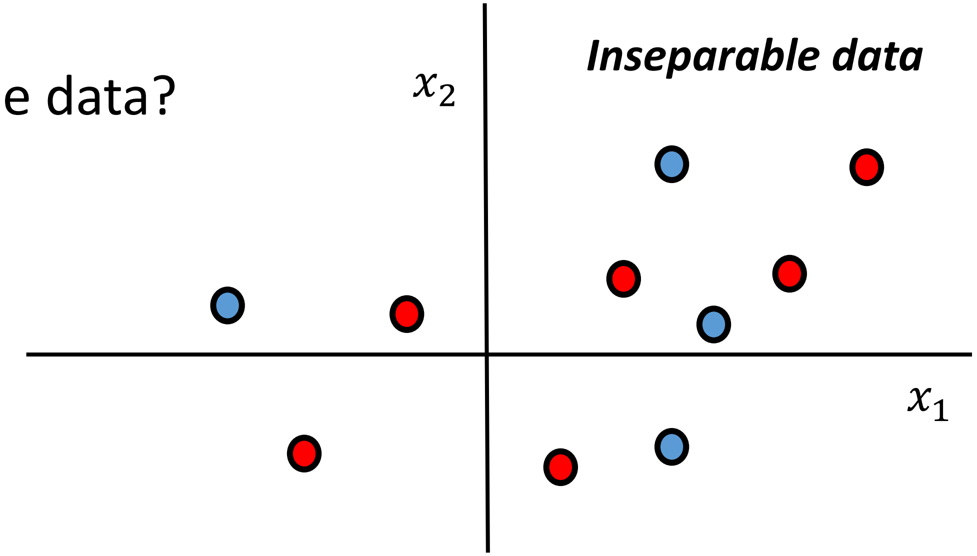
$$(2)\ \ w_2 = \mathrm{argmax}_w \frac{1}{\|w\|_2} \ \text{ s.t. }\ \min_{i\in[m]} |w^\top x_i| = 1,\ \ y_i \cdot w^\top x_i \geq 0\ \ \forall i \in [m]$$

$$(3)\ \ w_3 = \mathrm{argmin}_w \|w\|_2^2 \ \text{ s.t. }\ y_i \cdot w^\top x_i \geq 1\ \ \forall i \in [m]$$

- Now it’s time for #4 (and then #5!)

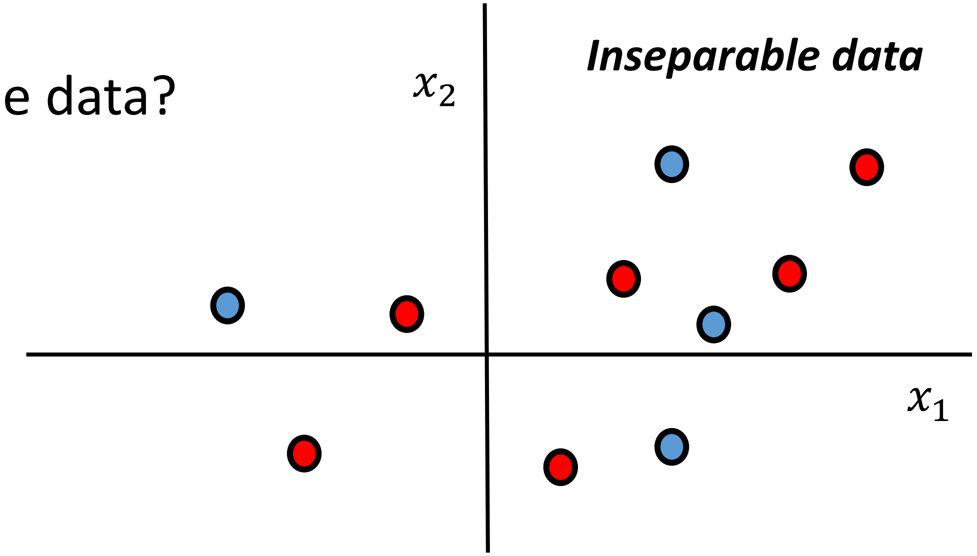
Removing the separability assumption

- **Q**: What happens if we use Hard SVM on non-separable data?



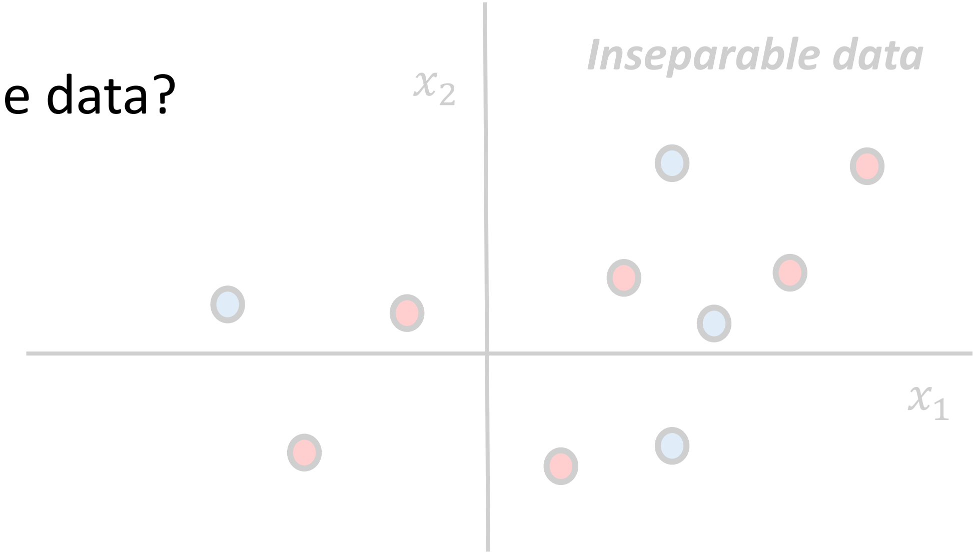
Removing the separability assumption

- **Q**: What happens if we use Hard SVM on non-separable data?
- **A**: Optimization problem has no feasible solutions.
  (meaning constraints cannot be satisfied)
- **Solution** – use "soft" margin constraints:
  penalize w by how much constraints are violated
- **Soft SVM**: penalize violations linearly (on average)



Removing the separability assumption

- **Q**: What happens if we use Hard SVM on non-separable data?
- **A**: Optimization problem has no feasible solutions.
  (meaning constraints cannot be satisfied)
- **Solution** – use "soft" margin constraints:
  penalize w by how much constraints are violated
- **Soft SVM**: penalize violations linearly (on average)



$$\begin{aligned} \underset{w \in \mathbb{R}^d, \xi \in \mathbb{R}^m}{\operatorname{argmin}} \quad & \lambda \|w\|_2^2 + \frac{1}{m} \sum_{i=1}^{m} \xi_i \\ \text{s.t.} \quad & y_i \cdot w^\top x_i \geq 1 - \xi_i \quad \forall i \in [m] \\ & \xi_i \geq 0 \end{aligned}$$

Soft SVM

- Penalties $\xi_i \geq 0$ are also called *slack variables*
- Separable data $\Rightarrow$ optimal solution has $\xi_i = 0$
- Penalization allows points to be inside margin, or even on "wrong" side!
- λ – will get back to this

$$\underset{w\in\mathbb{R}^d,\xi\in\mathbb{R}^m}{\operatorname{argmin}} \ \lambda\|w\|_2^2 + \frac{1}{m}\sum_{i=1}^{m}\xi_i$$

$$\text{s.t.} \quad y_i \cdot w^\top x_i \geq 1 - \xi_i \quad \forall i \in [m]$$

$$\xi_i \geq 0$$

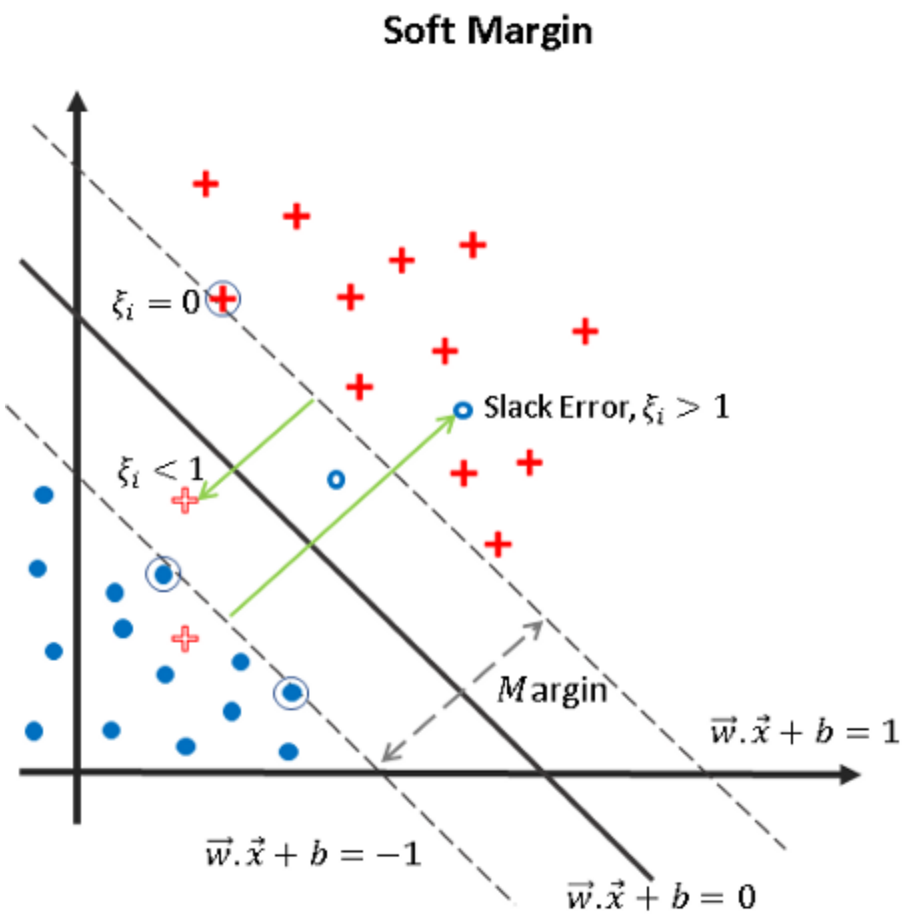


image: https://towardsdatascience.com/support-vector-machines-dual-formulation-quadratic-programming-sequential-minimal-optimization-57f4387ce4dd

Soft SVM

$$\underset{w\in\mathbb{R}^d,\xi\in\mathbb{R}^m}{\operatorname{argmin}} \ \lambda\|w\|_2^2 + \frac{1}{m}\sum_{i=1}^{m}\xi_i$$

$$\text{s.t.} \quad y_i \cdot w^\top x_i \geq 1 - \xi_i \ \ \forall i \in [m]$$

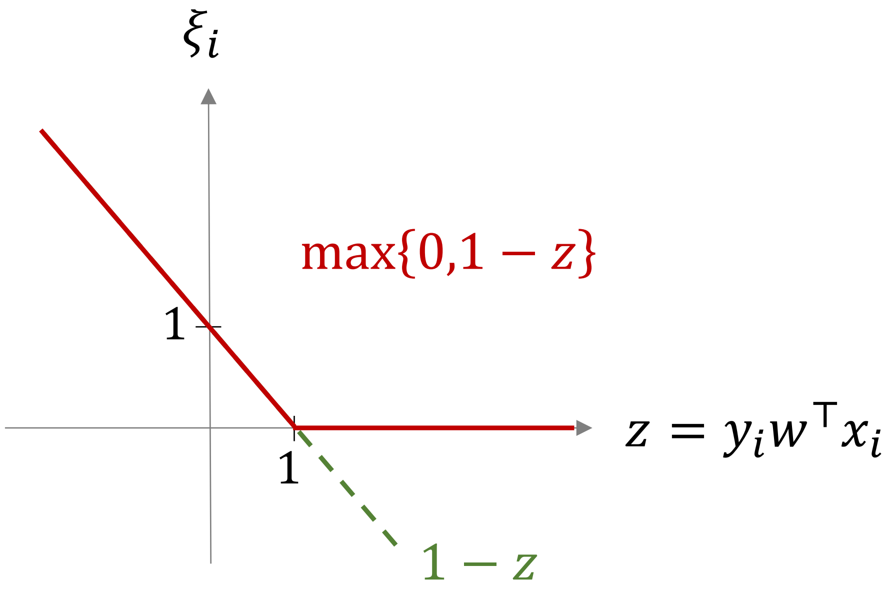
$$\xi_i \geq 0$$

- Constraints are ugly! (and not fun to optimize)
- Because we minimize over $\xi_i \geq 0$:

$$\xi_i = \begin{cases} 0 & \text{if } y_i \cdot w^\top x_i \geq 1 \\ 1 - y_i \cdot w^\top x_i & \text{if } y_i \cdot w^\top x_i < 1 \end{cases}$$

- Rewrite: $\xi_i = \max\{0, 1 - y_i \cdot w^\top x_i\}$

Soft SVM



$$\underset{w\in\mathbb{R}^d,\xi\in\mathbb{R}^m}{\operatorname{argmin}} \; \lambda\|w\|_2^2 + \frac{1}{m}\sum_{i=1}^{m}\xi_i$$

$$\text{s.t.} \quad y_i \cdot w^\top x_i \geq 1 - \xi_i \;\; \forall i \in [m]$$

$$\xi_i \geq 0$$

- Constraints are ugly! (and not fun to optimize)
- Because we minimize over $\xi_i \geq 0$:

$$\xi_i = \begin{cases} 0 & \text{if } y_i \cdot w^\top x_i \geq 1 \\ 1 - y_i \cdot w^\top x_i & \text{if } y_i \cdot w^\top x_i < 1 \end{cases}$$

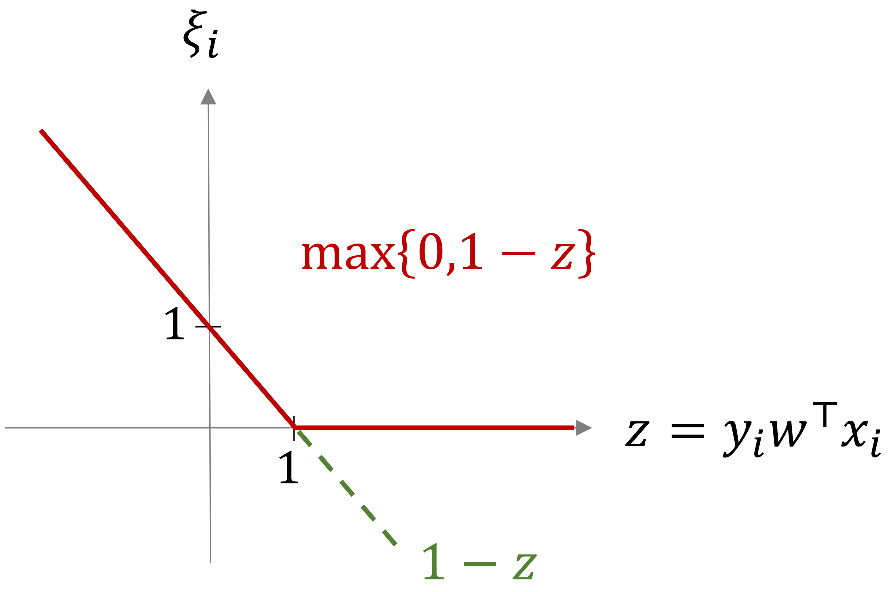
- Rewrite: $\xi_i = \max\{0, 1 - \underbrace{y_i \cdot w^\top x_i}_{= z}\}$

Soft SVM

- Plug in to get final **Soft SVM** formulation:

$$\underset{w\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{m}\sum_{i=1}^{m} \max\{0,1-y_i\cdot w^\top x_i\} + \lambda\|w\|_2^2$$

no more ξ!



$$\underset{w\in\mathbb{R}^d,\xi\in\mathbb{R}^m}{\operatorname{argmin}} \ \lambda\|w\|_2^2 + \frac{1}{m}\sum_{i=1}^{m}\xi_i$$

$$\text{s.t.} \quad y_i\cdot w^\top x_i \geq 1-\xi_i \ \ \forall i\in[m]$$

$$\xi_i \geq 0$$

- Constraints are ugly! (and not fun to optimize)
- Because we minimize over $\xi_i \geq 0$:

$$\xi_i = \begin{cases} 0 & \text{if } y_i\cdot w^\top x_i \geq 1 \\ 1-y_i\cdot w^\top x_i & \text{if } y_i\cdot w^\top x_i < 1 \end{cases}$$

- Rewrite: $\xi_i = \max\{0,1-\underbrace{y_i\cdot w^\top x_i}_{=z}\}$

Hinge formulation

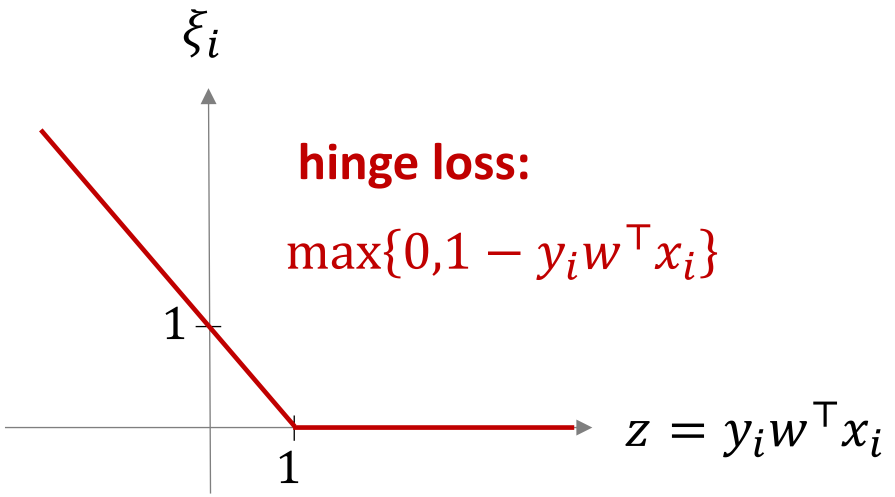
- We now see a template emerging:

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \underbrace{\boxed{\max\{0, 1 - y_i \cdot w^\top x_i\}}}_{\text{loss}} + \underbrace{\boxed{\lambda \|w\|_2^2}}_{\text{regularization}}$$

Hinge formulation

- We now see a template emerging:

$$\underset{w\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{m}\sum_{i=1}^{m} \underbrace{\boxed{\max\{0, 1 - y_i \cdot w^\top x_i\}}}_{\text{loss}} + \underbrace{\lambda \|w\|_2^2}_{\text{regularization}}$$



<u>**Loss:**</u>

- Penalizes model for being wrong
- SVM loss called *hinge loss* (=ציר, מפרק)

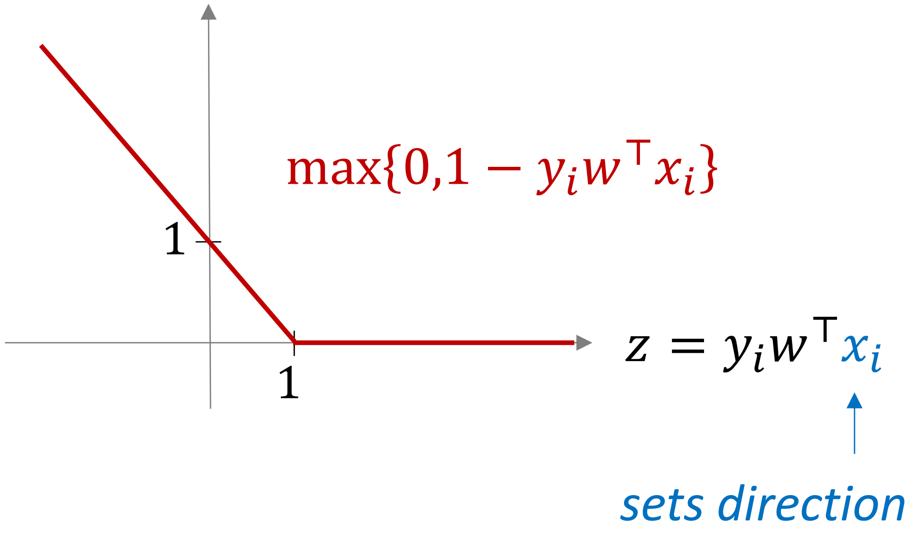
Hinge formulation

- We now see a template emerging:

$$\underset{w\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{m}\sum_{i=1}^{m} \boxed{\max\{0,1-y_i\cdot w^\top x_i\}} + \lambda\|w\|_2^2$$

loss regularization

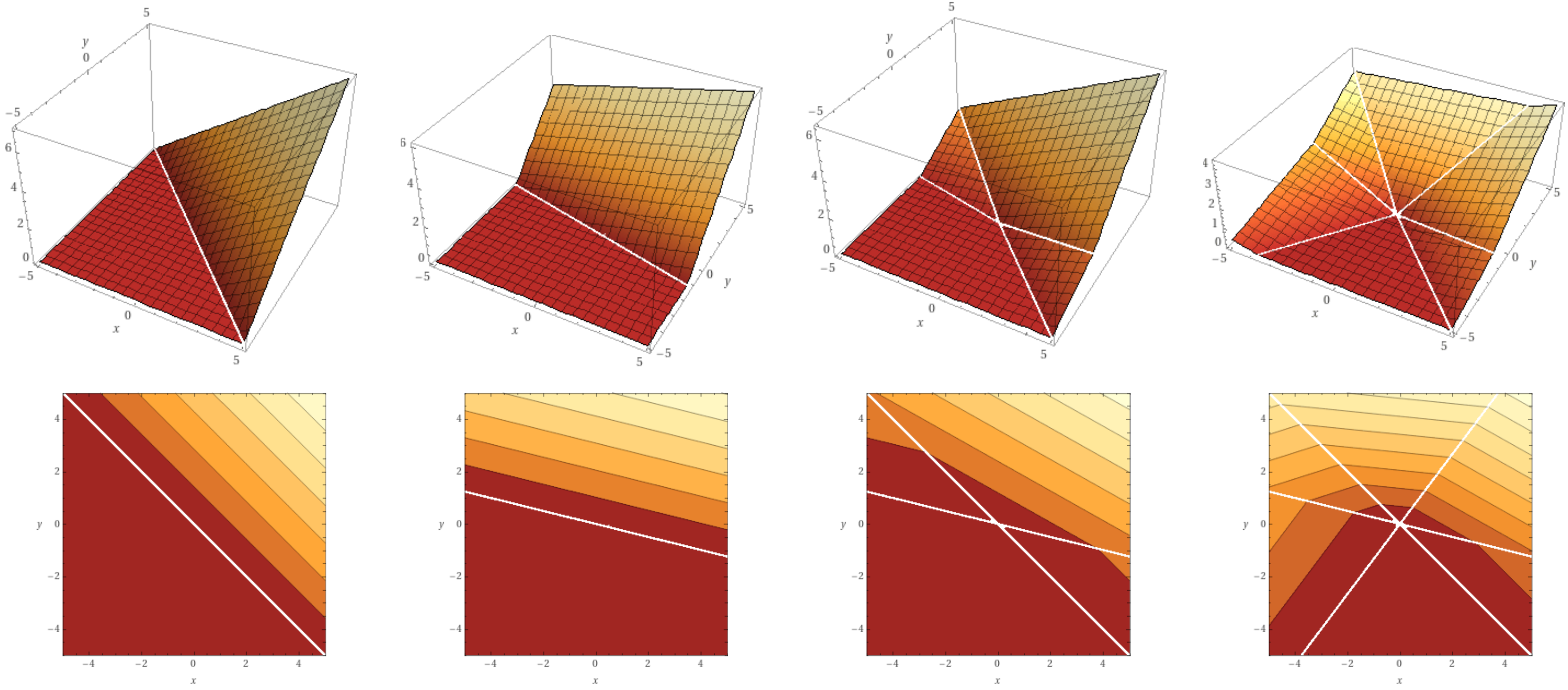
single example:



Loss:

- Penalizes model for being wrong
- SVM loss called *hinge loss* (=ציר, מפרק)
- Illustration is helpful, but can be misleading!
  - ➢ loss is plotted for one example – whereas real loss is average over many examples
  - ➢ loss appears to vary in a single "dimension" – but w can change in any direction in $\mathbb{R}^d$

multiple multidimensional examples: hinge loss as function of w



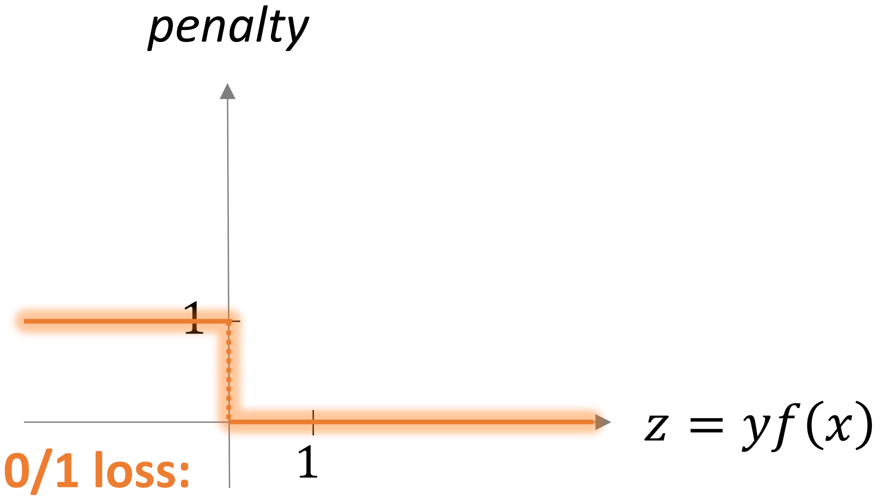
$x_1 = (0.7, 0.7)$ $\quad$ $x_2 = (0.24, 0.97)$ $\quad$ x_1, x_2 $\quad$ $x_1, x_2, x_3 = (-0.83, 0.55)$

https://www.wolframalpha.com/input/?i=plot3d+0.33*max%280%2C0.24x%2B0.97y%29%2B0.33*max%280%2C0.7x%2B0.7y%29%2B0.33*max%280%2C-0.83x%2B0.55y%29%2C+x%3D-5..5%2C+y%3D-5..5

Hinge formulation

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \underbrace{\boxed{\max\{0, 1 - y_i \cdot w^\top x_i\}}}_{\text{loss}} + \underbrace{\lambda \|w\|_2^2}_{\text{regularization}}$$

- Recall our real goal was to minimize the 0/1 loss, which is difficult to optimize



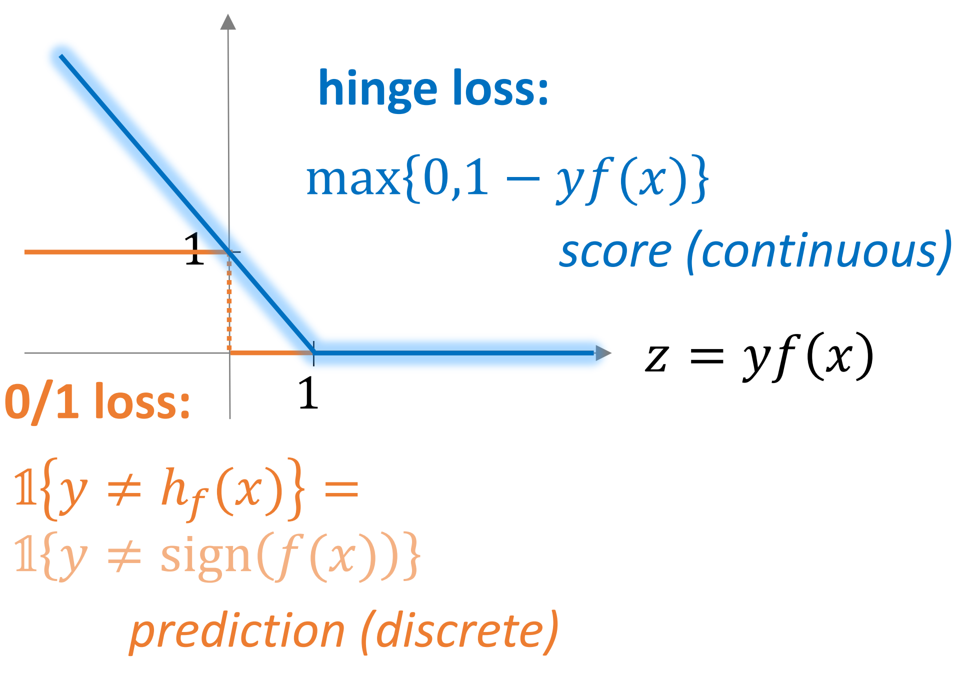
0/1 loss:

$\mathbb{1}\{y \neq h_f(x)\}$

Hinge formulation

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \boxed{\max\{0, 1 - y_i \cdot w^\top x_i\}} + \lambda \|w\|_2^2$$

loss regularization



- Recall our real goal was to minimize the 0/1 loss, which is difficult to optimize
- Hinge loss is a **tractable** alternative:
  it is continuous and convex, and upper bounds 0/1 loss
  note: sum of convex = convex
- Replaces optimization over (discrete) classifiers h with optimization over (continuous) score functions f
- **But tractability has its price:**
  - may suffer loss even if correct (i.e., when correct but by less than the margin)
  - if wrong, suffer linear loss (can be unbounded! vs. fixed loss of 1, as in 0-1 loss)
  - **well-known weakness**: outliers (think – why?)

Retelling our story

- Once upon a time we wanted to minimize expected 0/1 loss:
$$\min_{h \in H} \mathbb{E}_{(x,y)\sim D}[\mathbb{1}\{y \neq h(x)\}]$$
- But this is <u>statistically</u> hard (no access to D),
so instead we tried to minimize the empirical 0/1 loss (ERM):
$$\operatorname*{argmin}_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\{y_i \neq h(x_i)\}$$

Retelling our story

- Once upon a time we wanted to minimize expected 0/1 loss:

$$\min_{h \in H} \mathbb{E}_{(x,y)\sim D}[\mathbb{1}\{y \neq h(x)\}]$$

- But this is <u>statistically</u> hard (no access to D),
  so instead we tried to minimize the empirical 0/1 loss (ERM):

$$\operatorname*{argmin}_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\{y_i \neq h(x_i)\}$$

- But this is <u>computationally</u> hard (NP-hard discrete optimization),
  so instead we ended up minimizing a proxy loss – the hinge loss:

$$\operatorname*{argmin}_{w \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} \underbrace{\boxed{\max\{0, 1 - y_i \cdot w^\top x_i\}}}_{\text{loss}} + \lambda \|w\|_2^2$$

Retelling our story

- Once upon a time we wanted to minimize expected 0/1 loss:
$$\min_{h \in H} \mathbb{E}_{(x,y)\sim D}[\mathbb{1}\{y \neq h(x)\}]$$
- But this is <u>statistically</u> hard (no access to D),
so instead we tried to minimize the empirical 0/1 loss (ERM):
$$\operatorname*{argmin}_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\{y_i \neq h(x_i)\}$$
- But this is <u>computationally</u> hard (NP-hard discrete optimization),
so instead we ended up minimizing a proxy loss – the hinge loss:
$$\operatorname*{argmin}_{w \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i \cdot w^\top x_i\} + \boxed{\lambda \|w\|_2^2} \quad \leftarrow \textit{not ERM}$$
- **But oh no!** Turns out we're not minimizing the empirical error! What <u>are</u> we minimizing?

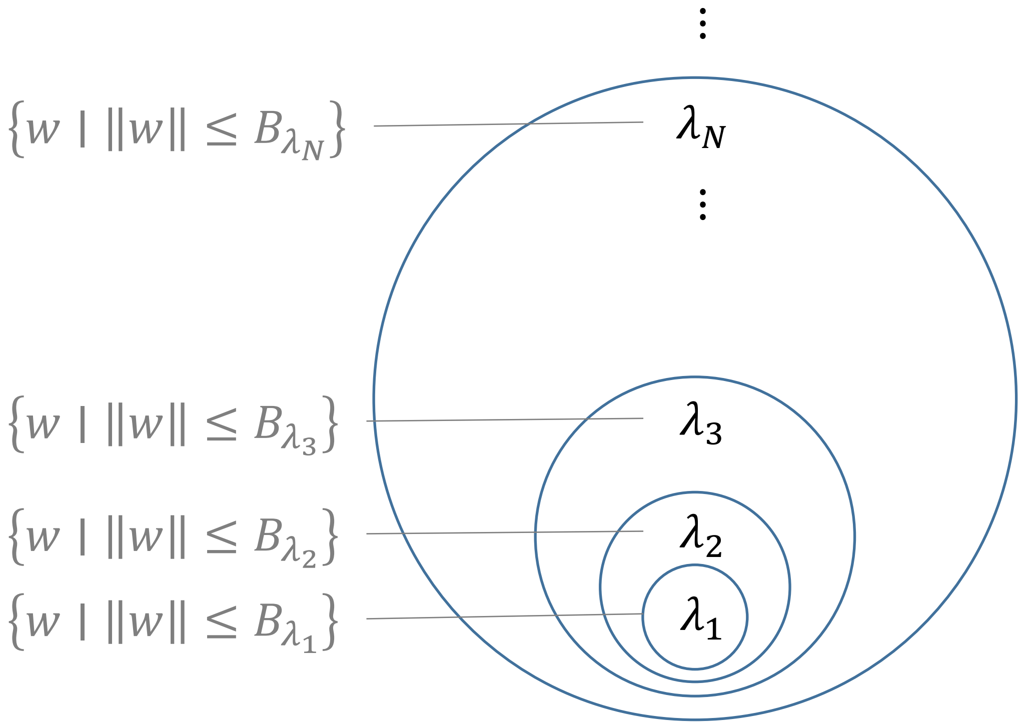
Regularization

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \underbrace{\max\{0, 1 - y_i \cdot w^\top x_i\}}_{\text{loss}} + \underbrace{\boxed{\lambda \|w\|_2^2}}_{\text{regularization}}$$

- The additive term $+\lambda\|w\|_2^2$ is called regularization – it *regularizes* solutions to have small norms
- Can be thought of as penalty on w-s with a large norm
- This is a way to inject prior knowledge, expressing a believe that low-norm w-s are "better"
- **Recall**: got $\|w\|$ by wanting max margin
- So in what sense are low-norm solutions better?

Hinge formulation

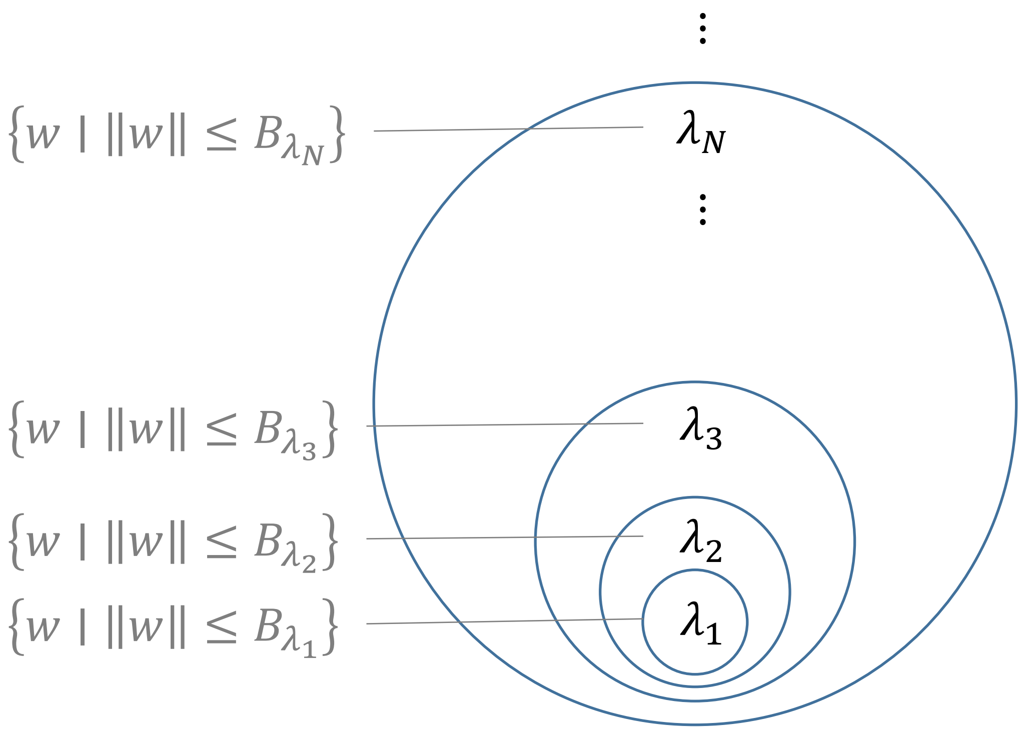
$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \underbrace{\frac{1}{m}\sum_{i=1}^{m} \max\{0, 1 - y_i \cdot w^\top x_i\}}_{\text{loss}} + \underbrace{\boxed{\lambda \|w\|_2^2}}_{\text{regularization}}$$



- **Claim:** low-norm w-s provide <u>better generalization</u>
- (formal proof next week; for now – some intuition)
- λ controls trade-off between loss and regularization:
    - Small $\lambda \Rightarrow$ large $\|w\|$ (and lower loss)
    - Large $\lambda \Rightarrow$ small $\|w\|$ (and possibly worse loss)
- Can think of each λ as inducing hard constraint $\|w\| \le B_\lambda$ for some B_λ (which we don't know!)
- A series $\lambda_1 > \lambda_2 > \cdots > \lambda_N > \cdots$ induces a nested hierarchy of models
- Can think of these as having **increased model complexity**, so choosing λ = choosing model class!

Hinge formulation

$$\underset{w\in\mathbb{R}^d}{\operatorname{argmin}} \underbrace{\frac{1}{m}\sum_{i=1}^{m} \max\{0,1-y_i\cdot w^\top x_i\}}_{\text{loss}} + \underbrace{\boxed{\lambda\|w\|_2^2}}_{\text{regularization}}$$



- **Conclusion:** tuning λ can control overfitting

- **Q1**: in a non-homogeneous model ($b \neq 0$), should we also regularize b?
- (A1: no – b just shifts the data, and has no effect on overfitting)
- **Q2**: we saw (and used!) the fact that H is scale invariant; don't norms just change the scale?
- (A2: no – even though for a <u>given</u> w "changing the scale" is similar to "changing the norm", this is not the same as considering a <u>set</u> of w-s with (at most) given norm)

Hinge formulation

$$\operatorname*{argmin}_{w \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i \cdot w^\top x_i\} + \boxed{\lambda} \|w\|_2^2$$

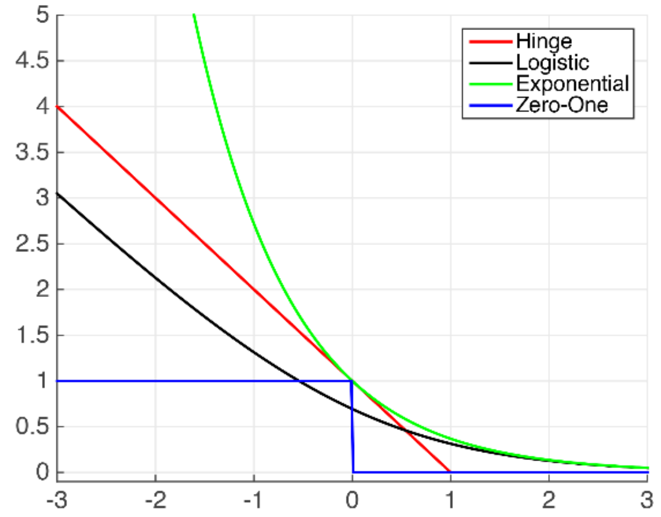
loss regularization

Regularization coefficient (λ):

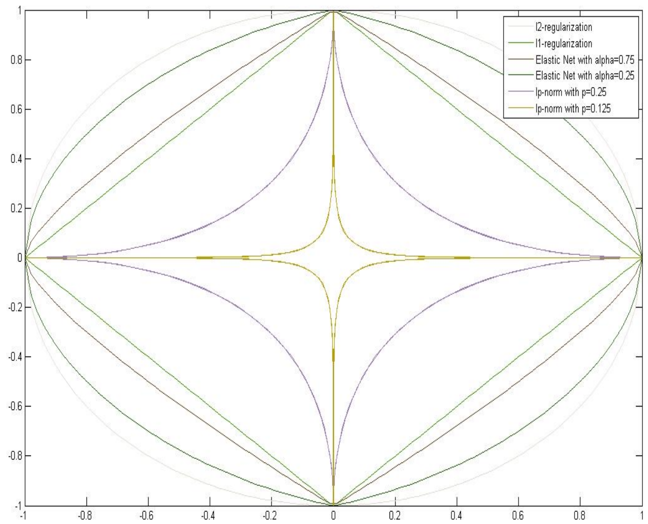
- Soft SVM penalizes violations of margin constraints
- But now, must decide – what is more important:
  large margin (min norm) –or– *small penalty* (constraint violations)?
- Determine **tradeoff** using λ
- As $\lambda \to 0$, constraints become "hard" constraints (recall ξ formulation)
- How to choose λ? Later in course!

Beyond SVM

$$\operatorname*{argmin}_{w \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} \underbrace{\boxed{\max\{0, 1 - y_i \cdot w^\top x_i\}}}_{\text{loss}} + \underbrace{\boxed{\lambda \|w\|_2^2}}_{\text{regularization}}$$



- Think of current objective as "template"
- Can use:
  - Other losses (logistic, exponential, ramp, …)
  - Other regularization (L_1, L_p, mixed, structured, …)
- Each has its pros, cons, use cases, and derivations
- We will see some later in course

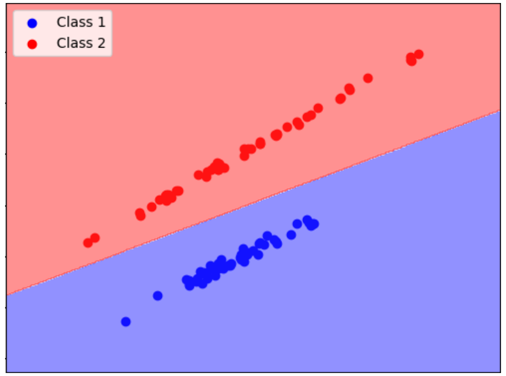


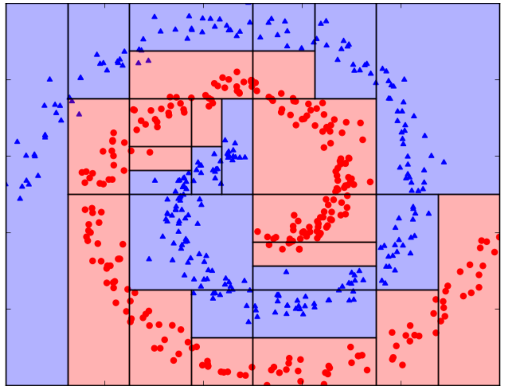
Kernels

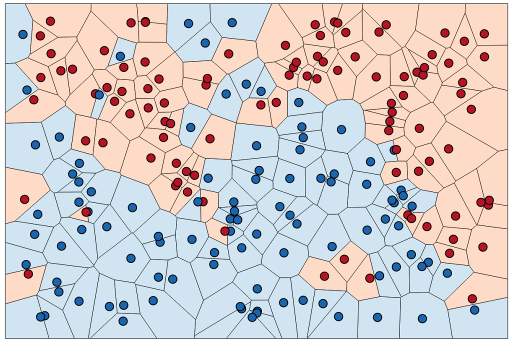
Linear non-linearity

- SVMs are great!
- But a clear limitation is experssivity:
  work only for **linear tresholds**
- Compare this to k-NN and decisions trees

- What can we do to make SVMs **more experssive**?



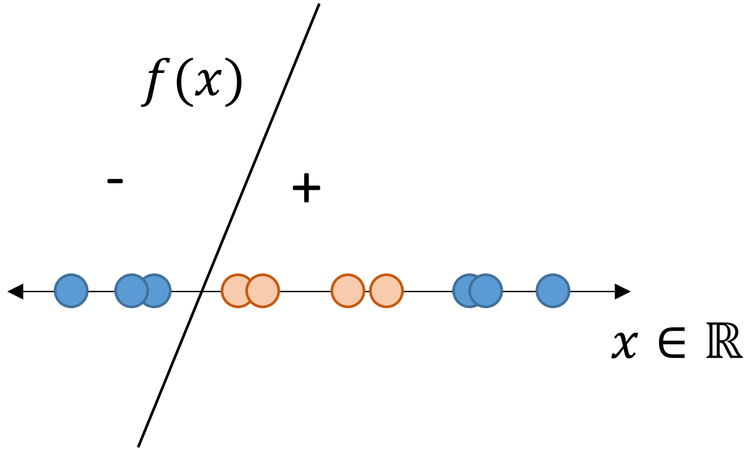




Linear non-linearity

- Recall 1D example from lecture #2:
- Can't get perfect accuracy with a linear threshold classifier!

$$h(x) = \text{sign}\big(f(x)\big), \qquad f(x) = w_0 + w_1 x$$



Linear non-linearity

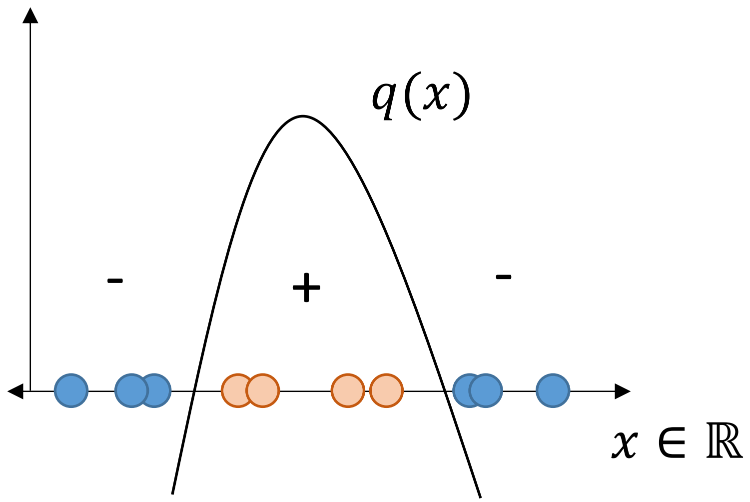
- Recall 1D example from lecture #2:
- Can't get perfect accuracy with a linear threshold classifier!
- But, can do this with <u>polynomial</u> threshold: (here, of degree=2)

$$h(x) = \text{sign}\big(q(x)\big), \qquad q(x) = w_0 + w_1 x + w_2 x^2$$

- But a non-linear polynomial is also… linear! (?!
- Can rewrite as *linear* function of polynomial *representation*:

$$q(x) = f_w\big(\phi(x)\big) = w^\top \phi(x), \qquad \phi(x) = (1, x, x^2)$$

- $\phi: \mathbb{R}^d \to \mathbb{R}^{d'}$ is called a **feature mapping**
  (these will pup-up again and again…)

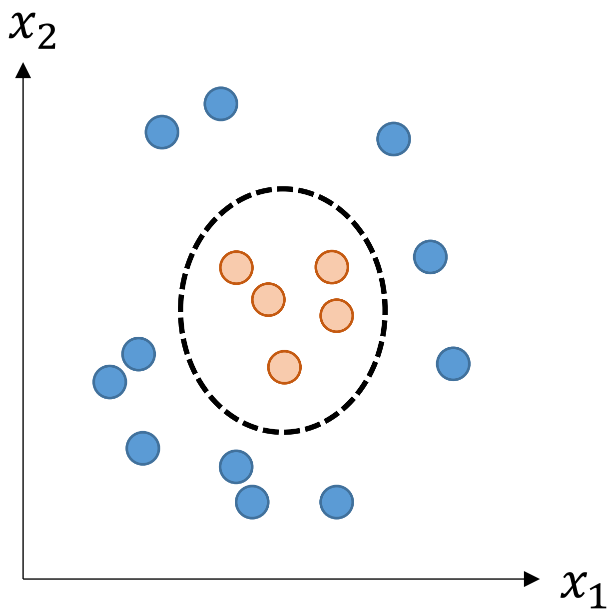


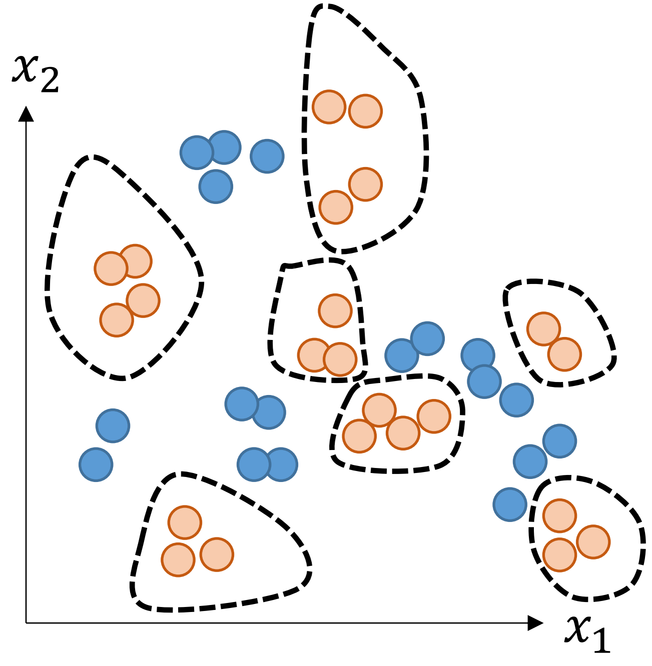
Linear non-linearity

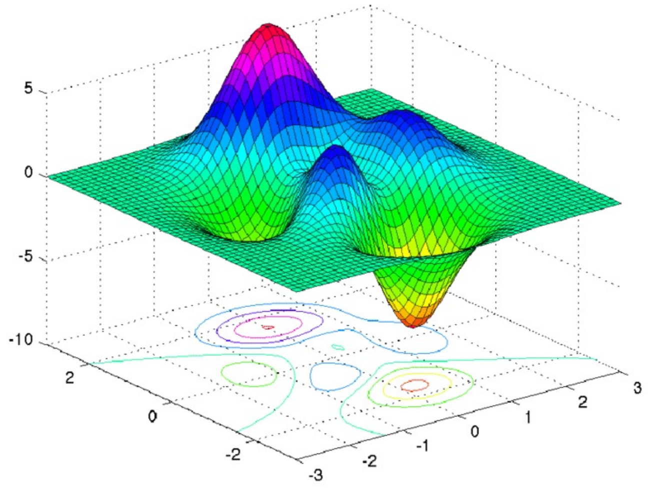
- Also works for higher dimensional *inputs*
- E.g., 2D example: (cannot be linearly separated in x)
- Possible useful feature mappings:

1. **Polynomial**: $\phi(x) = (1, x_1, x_2, x_1 x_2, x_1^2, x_2^2)$
2. **Radial**: $\phi(x) = (r, \theta)$ (if data is centered)

- Sometimes we'll need higher-dimensional *outputs* (=polynomials)







Linear non-linearity

- **General Feature mappings:** $\phi: \mathbb{R}^d \to \mathbb{R}^{d'}$
- Can make linear SVMs be non-linear!
- (any multivariate mapping ϕ works, not restricted to polynomials – more on this later!)

- **Recipe:**
  - choose ϕ (e.g., as some (multi-variate) polynomial)
  - apply $x_i' = \phi(x_i)$ to all x_i in the dataset
  - use the SVM algorithm to learn on modified data (x_i', y_i)
- **Result**: a linear classifier $w^\top \phi(x)$ that behaves like a non-linear classifier on x!

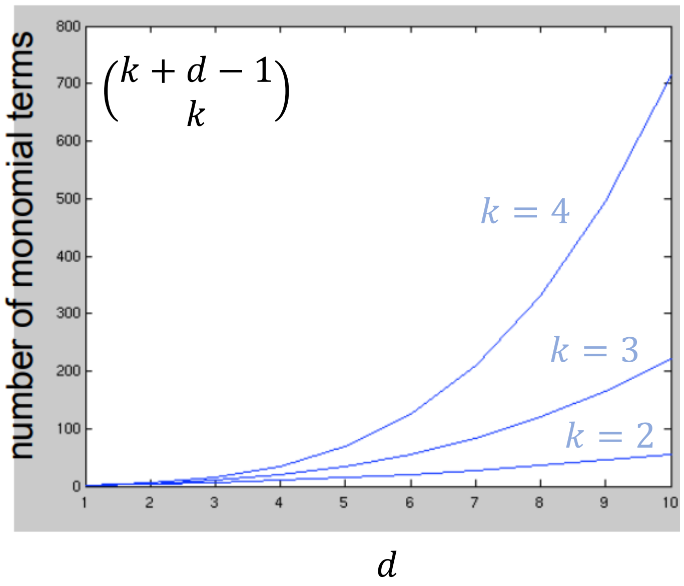
Linear non-linearity

- Using a non-linear representation ϕ is a *modeling choice*
- Will it work well? How can we tell?

- As always, consult our **three pillars**:
  - **Modeling**: choose ϕ if you believe it (linearly) separates the data well (approx.)
  - **Statistics:** (will see next week)
  - **Optimization:** up next!

Linear non-linearity

- Consider ϕ to be a polynomial of degree k and dimension d
- **Q**: What is the dimension d' of the induced linear classifier?
- **A**: $d' = |\phi(x)|$ = number of monomials = $\binom{k+d-1}{k}$
- **Examples**:
  - $d = 3, \quad k = 3 \quad \rightarrow \quad d' = 10$
  - $d = 30, \quad k = 3 \quad \rightarrow \quad d' = 4960$
  - $d = 30, \quad k = 30 \quad \rightarrow \quad d' \approx 6 \times 10^{17}$
- **Problem**: d' grows *fast*, making SVM computationally demanding
- (even just storing $\phi(x)$ can be challenging!)
- **Goal for this section:** make SVM work for arbitrary ϕ – **even huge!**
- **Solution**: transform into *another* optimization problem (yes, again)



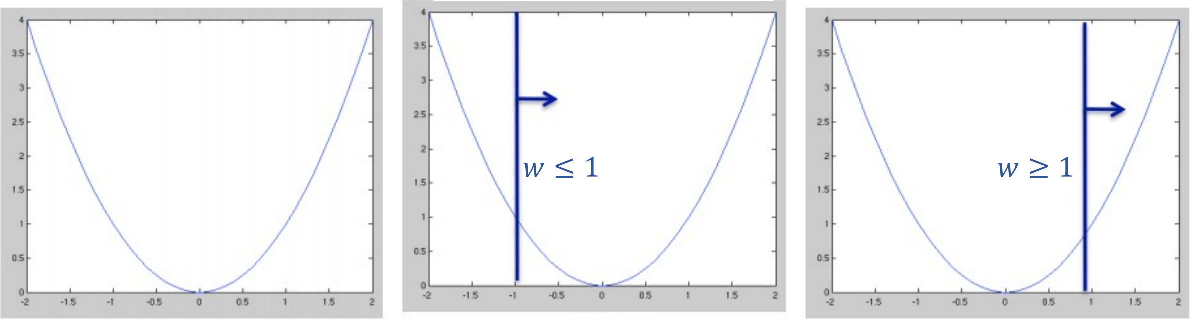
http://www.cs.cmu.edu/~guestrin/Class/10701-S07/Slides/kernels.pdf

Quadratic constrained optimization

- **Gentle start**: Hard-SVM with one example (so single constraint):

$$\min_{w\in\mathbb{R}} w^2 \quad \text{s.t.} \quad xyw \geq 1$$

unconstrained | $x = 1, y = -1$ | $x = 1, y = 1$



$w^* = 0$ | $w^* = 0$ | $w^* = 1$

- **Quadratic + linear constraints**: global optimum ⟺ gradient = 0 (or at boundary)
- **Hand-wavy**: local improvement => global optimum => exact and efficient optimization
- (more on this – week 7)

http://www.cs.cmu.edu/~guestrin/Class/10701-S07/Slides/kernels.pdf

Duality – simple case

- 1D Hard-SVM: $\min_{w \in \mathbb{R}} w^2$ s.t. $w \geq \frac{1}{xy} = b$
- **Lagrange method**: express constraint within objective

Lagrangian: $L(w, \alpha) = w^2 - \alpha(w - b), \ \alpha \in \mathbb{R}_+$

Duality – simple case

- 1D Hard-SVM: $\min_{w\in\mathbb{R}} w^2$ s.t. $w \geq \frac{1}{xy} = b$
- **Lagrange method**: express constraint within objective

$$\boxed{\textbf{Lagrangian: } L(w,\alpha) = w^2 - \alpha(w-b),\ \ \alpha \in \mathbb{R}_+}$$

- **“New“ objective**: solve $\boxed{\min_{w\in\mathbb{R}} \max_{\alpha\in\mathbb{R}_+} L(w,\alpha)}$
- **Claim**: problems are equivalent
- **Intuition**:
  - min and max “compete”
  - $\alpha \geq 0$ ensures w satisfies constraints
  - “Pushes” violating w towards edge of constraints

- **Why equivalence?**
  - Case I: $w < b$ → $w - b < 0$
    → $\max_{\alpha} -\alpha(w-b) = \infty$
    $\min_w$ will never choose such w
  - Case II: $w > b$ → $w - b > 0$
    → $\max_{\alpha} -\alpha(w-b) = 0$
    min doesn’t mind,
    objective recovered: $L(w,\alpha) = w^2$
  - Case III: $w = b$ → any α works
    objective recovered: $L(w,\alpha) = w^2$

Duality – general case

- Hard-SVM: $\operatorname{argmin}_{w \in \mathbb{R}^d} \|w\|_2^2$ s.t. $y_i w^\top x_i \geq 1 \ \forall i \in [m]$
- **Lagrangian**: $L(w, \alpha) = \|w\|_2^2 - \sum_{i=1}^m \alpha_i (y_i w^\top x_i - 1), \alpha \in \mathbb{R}_+^m$ (multiple constraints => sum)

Duality – general case

- Hard-SVM: $\operatorname{argmin}_{w \in \mathbb{R}^d} \|w\|_2^2$ s.t. $y_i w^\top x_i \geq 1 \ \forall i \in [m]$
- **Lagrangian**: $L(w, \alpha) = \|w\|_2^2 - \sum_{i=1}^m \alpha_i (y_i w^\top x_i - 1), \alpha \in \mathbb{R}_+^m$ (multiple constraints => sum)
- Primal objective: $\boxed{\min_{w \in \mathbb{R}^d} \max_{\alpha \in \mathbb{R}_+^m} L(w, \alpha)}$ **?** $\boxed{\max_{\alpha \in \mathbb{R}_+^m} \min_{w \in \mathbb{R}^d} L(w, \alpha)}$ (*dual objective*)
- **Dubious move**: swap min $\leftrightarrow$ max

Duality – general case

- Hard-SVM: $\mathrm{argmin}_{w\in\mathbb{R}^d}\|w\|_2^2$ s.t. $y_i w^\top x_i \geq 1\ \forall i \in [m]$
- **Lagrangian**: $L(w,\alpha) = \|w\|_2^2 - \sum_{i=1}^m \alpha_i(y_i w^\top x_i - 1),\ \alpha \in \mathbb{R}_+^m$ (multiple constraints => sum)
- Primal objective: $\boxed{\min_{w\in\mathbb{R}^d}\ \max_{\alpha\in\mathbb{R}_+^m} L(w,\alpha) \quad \geq \quad \max_{\alpha\in\mathbb{R}_+^m}\ \min_{w\in\mathbb{R}^d} L(w,\alpha)}$ (*dual objective*)

- **Dubious move**: swap min ↔ max
- In general, minmax ≥ maxmin (“max min inequality”; see wiki)

Duality – general case

- Hard-SVM: $\operatorname{argmin}_{w \in \mathbb{R}^d} \|w\|_2^2$ s.t. $y_i w^\top x_i \geq 1 \ \forall i \in [m]$
- **Lagrangian**: $L(w, \alpha) = \|w\|_2^2 - \sum_{i=1}^m \alpha_i (y_i w^\top x_i - 1), \alpha \in \mathbb{R}_+^m$ (multiple constraints => sum)
- Primal objective: $\boxed{\min_{w \in \mathbb{R}^d} \max_{\alpha \in \mathbb{R}_+^m} L(w, \alpha) = \max_{\alpha \in \mathbb{R}_+^m} \min_{w \in \mathbb{R}^d} L(w, \alpha)}$ (*dual objective*)

- **Dubious move**: swap min ↔ max
- In general, minmax ≥ maxmin (“max min inequality”; see wiki)
- **But**: convex in w (for fixed α) + concave in α (for fixed w) ⇒ **equality**!
- (aka minimax theorem; won’t prove)
- **Bonus**: lies at core of game theory (zero-sum games); adversarial learning, GANs.

Dual SVM

- **Dual objective**: $\max_{\alpha \in \mathbb{R}_+^m} \min_{w \in \mathbb{R}^d} \frac{1}{2}\|w\|_2^2 - \sum_{i=1}^m \alpha_i (y_i w^\top x_i - 1),\ \alpha \in \mathbb{R}_+^m$
- **Next step**: solve for optimal w (as function of α) by setting derivative $= 0$

$$\frac{\partial L}{\partial w} = w - \sum_i \alpha_i y_i x_i = 0 \;\rightarrow\; w = \sum_i \alpha_i y_i x_i \quad (w, x \in \mathbb{R}^d)$$

- Plugging into objective (and simplifying) gives:

Dual (Hard) SVM: $\underset{\alpha \in \mathbb{R}_+^m}{\operatorname{argmax}} \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y_i y_j \alpha_i \alpha_j x_i^\top x_j$

- Time to reap the fruits!

Dual SVM

- **Dual (Hard) SVM:** $$\underset{\alpha \in \mathbb{R}_+^m}{\operatorname{argmax}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j x_i^\top x_j$$

- **Q**: Where did w go?

Dual SVM

- **Dual (Hard) SVM:** $\boxed{\operatorname*{argmax}_{\alpha \in \mathbb{R}_+^m} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j x_i^\top x_j}$
- **Q**: Where did w go?
- **A**: Derivation reveals optimality constraint:

$$w = \sum_{i=1}^{m} \alpha_i y_i x_i \quad \Rightarrow \quad f_w(x) = w^\top x = \boxed{\sum_{i=1}^{m} \alpha_i y_i x_i^\top x = f_\alpha(x)}$$

- Classifier as *linear combination of data points* (special case of "representer theorem")
- Dual SVM optimizes *directly over α*
- In practice, α is almost always sparse
  (We know this! Think: only support vectors have $\alpha_i > 0$)

The kernel trick

- **Recall**: we want to work with a *feature mapping* $x \to \phi(x)$

learning objective: *(train time)*

$$\max_{\alpha \in \mathbb{R}^m_+} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j x_i^\top x_j$$

classifier: *(test time)*

$$f_\alpha(x) = \sum_{i=1}^{m} \alpha_i y_i x_i^\top x$$

- **Observation**: in dual, features appear only as *inner products* $x^\top x'$
- **Kernel trick:** given representation ϕ, replace inner product: $x^\top x' \to \phi(x)^\top \phi(x')$
- New *kernelized* objective and classifier:

$$\max_{\alpha \in \mathbb{R}^m_+} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \phi(x_i)^\top \phi(x_j)$$

$$f_\alpha(x) = \sum_{i=1}^{m} \alpha_i y_i \phi(x_i)^\top \phi(x)$$

The kernel trick

- **Definition**: *kernel function* $K(x, x') = \phi(x)^\top \phi(x')$

- Can rewrite:

learning objective: *(train time)*

$$\max_{\alpha \in \mathbb{R}_+^m} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j K(x_i, x_j)$$

classifier: *(test time)*

$$f_\alpha(x) = \sum_{i=1}^{m} \alpha_i y_i K(x_i, x)$$

- To optimize (=train), need only *entries* of *kernel matrix* $K_{ij} = K(x_i, x_j) = \phi(x_i)^\top \phi(x_j)$
  (kernel matrix can be *precomputed* from original features before training)
- To classify, need only kernel *queries* $K(x, x_i) = \phi(x)^\top \phi(x_i)$
  (must be computed on the fly for all x_i for which $\alpha_i > 0$)
- **But:** kernels are <u>useful</u> if $\phi(x)^\top \phi(x')$ can be computed **efficiently** – let's see why!

Polynomial kernel

- Recall that a polynomial ϕ has d' that is exponential in k, d
- This means that computing $\phi(x)^\top \phi(x')$ directly is intractable

- Luckily – a polynomial kernel admits a simple closed-form solution:

$$K_{\text{poly}}(x, x'; k) = \phi(x)^\top \phi(x') = (x^\top x')^k$$

- (To see why, recall that $(a+b)^2 = a^2 + ab + b^2$; now expand to power of k)

- **Compute time**: $O(d)$

Kernels - examples

- There are many, many types of kernels
- Examples of some known kernels with compact formulation:

- Polynomials of degree $= k$: $K(x, x'; k) = (x^\top x')^k$
- Polynomials of degree $\leq k$: $K(x, x'; k) = (x^\top x' + 1)^k$ (*more in tirgul*)
- RBF/Guassian (scale sigma): $K(x, x'; \sigma) = \exp\left\{-\frac{1}{2\sigma^2}\|x - x'\|_2^2\right\}$ (*next up*)
- Sigmoid: $K(x, x'; \eta, \nu) = \tanh(\eta x^\top x' + \nu)$

- Compute time for each of the above is $O(d)$
- Notice kernels have *parameters*

$\sigma \to \pm\infty : e^0 = 1$

$\sigma \to 0 : e^{-\infty} = 0$

אין השפעה של המרחקים מהנק' ⇐

Well-defined kernels

- How do we know a given kernel is “valid”?
- For example, is this a kernel?

RBF kernel: $K(x, x'; \sigma) = \exp\left\{-\frac{1}{\sigma^2}\|x - x'\|_2^2\right\}$

- Two options to validate:
  1. Figure out what the underlying ϕ is; show $K(x, x') = \phi(x)^\top \phi(x')$
  2. Use **kernel algebra**
  3. Show kernel matrix is (semi)-positive-definite (PSD) – in week 7!

Well-defined kernels

- **Kernel composition rules:**
  any of the following operations give valid kernels
  1. $K(x, x') = x^\top x'$
  2. $K(x, x') = x^\top A x',\ A \succcurlyeq 0$ (PSD)
  3. $K(x, x') = cK_1(x, x'),\ c \geq 0$
  4. $K(x, x') = K_1(x, x') + K_2(x, x')$
  5. $K(x, x') = K_1(x, x') \cdot K_2(x, x')$
  6. $K(x, x') = g\big(K_1(x, x')\big),\ g$ polynomial (with ≥ 0 coeffs.)
  7. $K(x, x') = f(x) K(x, x') f(x'),\ f$ is any function
  8. $K(x, x') = \exp\{K_1(x, x')\}$

- Can use above composition rules to check validity
- And to create new kernels!

Claim: RBF kernel

$$K(x, x'; \sigma) = \exp\left\{-\frac{1}{\sigma^2} \|x - x'\|_2^2\right\}$$

is a valid kernel.

Proof: using composition rules

1. $K_1(x, x') = x^\top x'$ (1)
2. $K_2(x, x') = \frac{2}{\sigma^2} K_1(x, x')$ (2)
3. $K_3(x, x') = e^{K_2(x, x')} = e^{\frac{2x^\top x'}{\sigma^2}}$ (8)
4. $K_4(x, x') = e^{\frac{-x^\top x}{\sigma^2}} K_3(x, x') e^{\frac{-x'^\top x'}{\sigma^2}}$ (7)

$$= e^{\frac{-x^\top x}{\sigma^2}} e^{\frac{2x^\top x'}{\sigma^2}} e^{\frac{-x'^\top x'}{\sigma^2}}$$

$$= e^{\frac{-x^\top x + 2x^\top x' - x'^\top x'}{\sigma^2}}$$

$$= e^{\frac{-\|x - x'\|_2^2}{\sigma^2}} = K_{\mathrm{RBF}}(x, x')$$

https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote14.html

More kernels!

- Kernels for *structured inputs*:
  - graphs
  - trees
  - sets
  - strings
  - text
  - molecules
  - …
- See lists, surveys, packages, …

Contents

http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications/#kernel_functions

https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote14.html

RBF primal representation

- **Q**: What is the feature representation $\phi(x)$ underlying RBF?

https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote14.html

RBF primal representation

- **Q**: What is the feature representation $\phi(x)$ underlying RBF?
- **A**:

$$\begin{aligned}
\exp\left(-\frac{1}{2}\|\mathbf{x}-\mathbf{x}'\|^2\right) &= \exp\left(\frac{2}{2}\mathbf{x}^\top\mathbf{x}' - \frac{1}{2}\|\mathbf{x}\|^2 - \frac{1}{2}\|\mathbf{x}'\|^2\right) \\
&= \exp(\mathbf{x}^\top\mathbf{x}')\exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)\exp\left(-\frac{1}{2}\|\mathbf{x}'\|^2\right) \\
&= \sum_{j=0}^{\infty}\frac{(\mathbf{x}^\top\mathbf{x}')^j}{j!}\exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)\exp\left(-\frac{1}{2}\|\mathbf{x}'\|^2\right) \\
&= \sum_{j=0}^{\infty}\sum_{\sum n_i=j}\underbrace{\exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)\frac{x_1^{n_1}\cdots x_k^{n_k}}{\sqrt{n_1!\cdots n_k!}}}_{\text{single entry in }\phi(x)}\underbrace{\exp\left(-\frac{1}{2}\|\mathbf{x}'\|^2\right)\frac{{x'}_1^{n_1}\cdots {x'}_k^{n_k}}{\sqrt{n_1!\cdots n_k!}}}_{\text{single entry in }\phi(x')}
\end{aligned}$$

(source: wikipedia)

- Basis expansion of *infinite* support (that is, $d' = |\phi(x)| = \infty$)
- Nonetheless, can still be learned using Dual SVM (aka *Kernel SVM*)
- Unrestricted complexity - can fit *any function* (*universal* kernel)
- Computationally, no worries – but **beware overfitting!**

Modeling

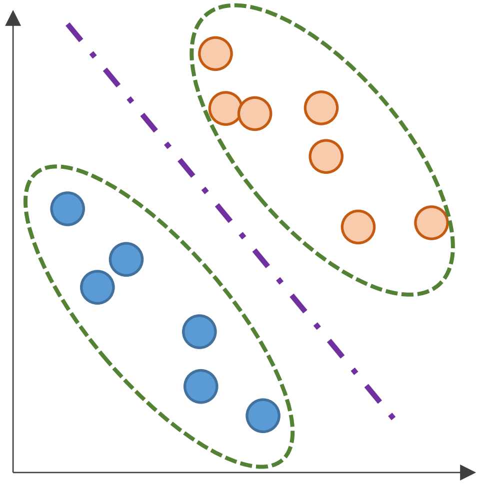
- Kernels define a *similarity measure* via inner products $\phi(x)^\top \phi(x')$
- Compare:

linear kernel:

$$K(x, x') = x^\top x'$$

RBF kernel:

$$K(x, x') = \exp\left\{-\frac{1}{\sigma^2}\|x - x'\|_2^2\right\}$$



- Dual – similarity; Primal – distance (from hyperplane)
- (well separated (*distance*) ≈ clustered together (*similarity*))

- This provides flexible and expressive modeling power:
  instead of thinking in terms of distances, can think in terms of similarities
- (kNN anyone?)
- Relation between **distances** and **similarities** is fundamental in learning!

Discussion

- SVM as major milestone in ML history
- Had all you could ask for:
  1. Computationally tractable
     (fast convergence to optimum)
  2. Highly expressive (with kernels)
  3. Statistical guarantees (next week!)
- Prime example of "complete" learning approach, great to learn from.
- At time, state of the art performance
  (today, still competitive baseline)
- "Just need the right kernel"
  (vs: "just need the right neural architecture")

> "Give me a place to stand, and a lever long enough, and I shall move the earth"
>
> *- Archimedes*

Next week(s)

- **Finished part I**: *supervised binary classification*
- **Next up – part II**: *the different aspects of learning*
  1. Statistics: generalization and PAC theory
  2. Modeling: model selection and evaluation
  3. Optimization: convexity, gradient descent
  4. Practical aspects and potential pitfalls
- (will mostly use SVM as use case)