CPSC 340:
Machine Learning and Data Mining

Linear Classifiers fit & multi-class

Bonus slides

‘λ’ vs ‘C’ as SVM Hyper-Parameter

- We’ve written SVM in terms of regularization parameter ‘λ’:

$$f(w) = \sum_{i=1}^{n} \max\{0, 1 - y_i w^T x_i\} + \frac{\lambda}{2}\|w\|^2$$

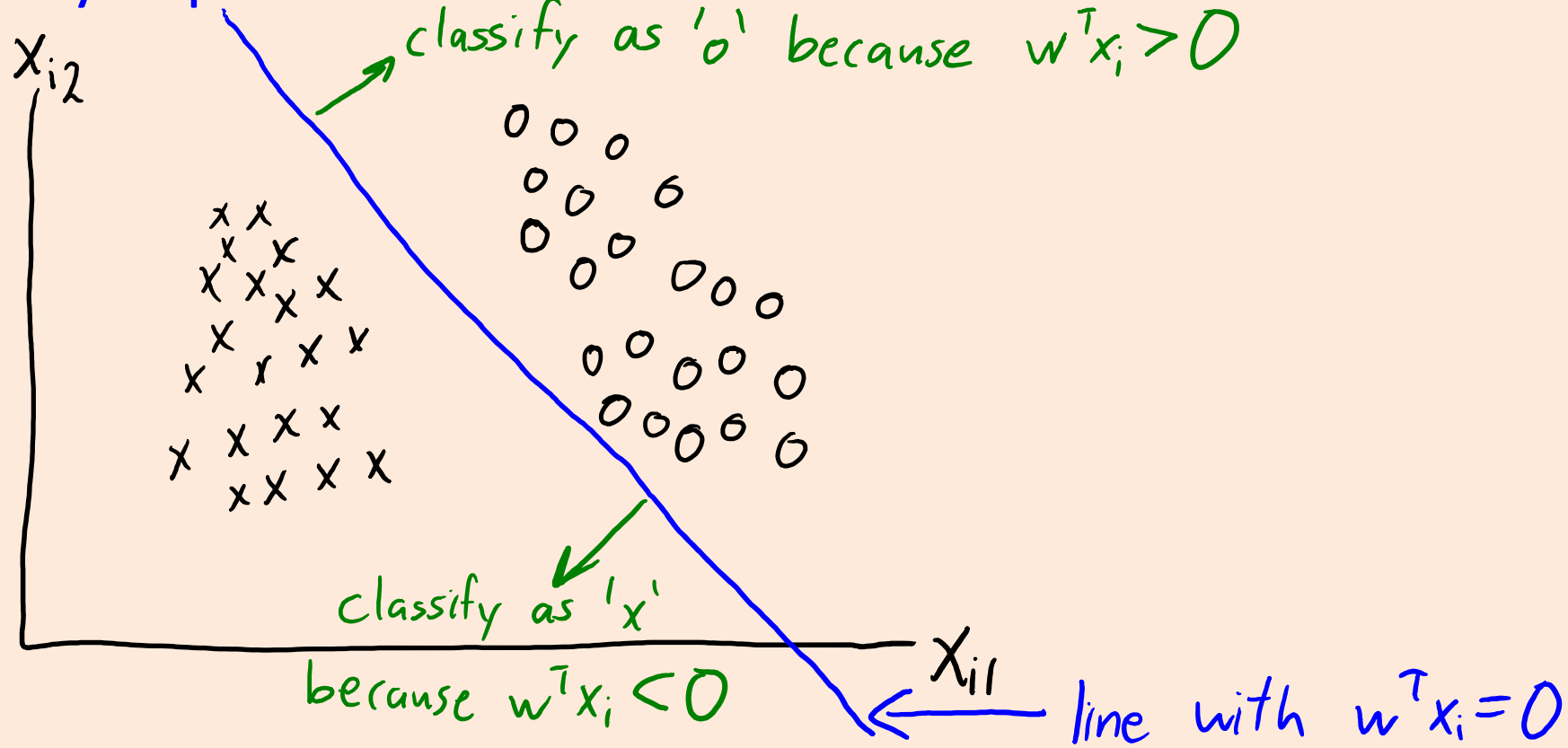
- Some software packages instead have regularization parameter ‘C’:

$$f(w) = C\sum_{i=1}^{n} \max\{0, 1 - y_i w^T x_i\} + \frac{1}{2}\|w\|^2$$

- In our notation, this corresponds to using $\lambda = 1/C$.
  - Equivalent to just multiplying f(w) by constant.
  - Note interpretation of ‘C’ is different: high regularization for small ‘C’.
    - You can think of ‘C’ as “how much to focus on the classification error”.

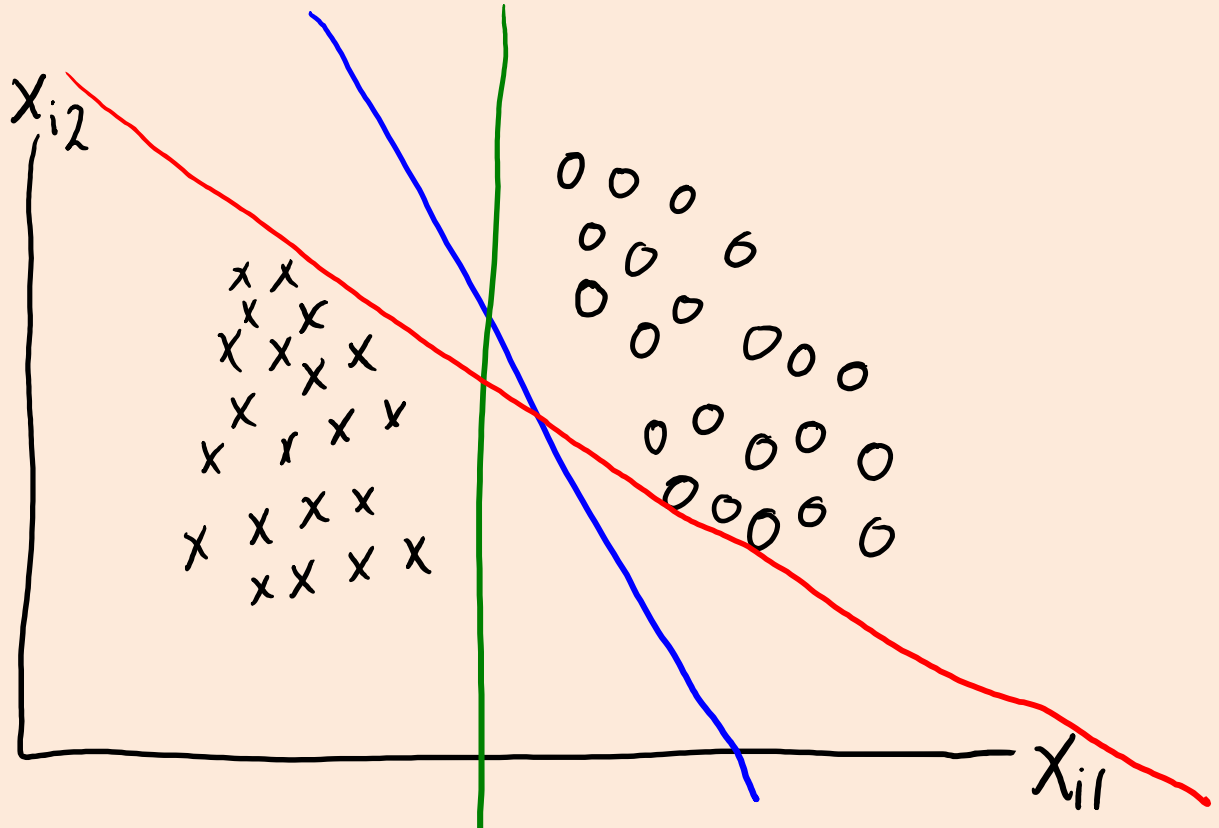
Maximum-Margin Perspective

- Consider a linearly-separable dataset.



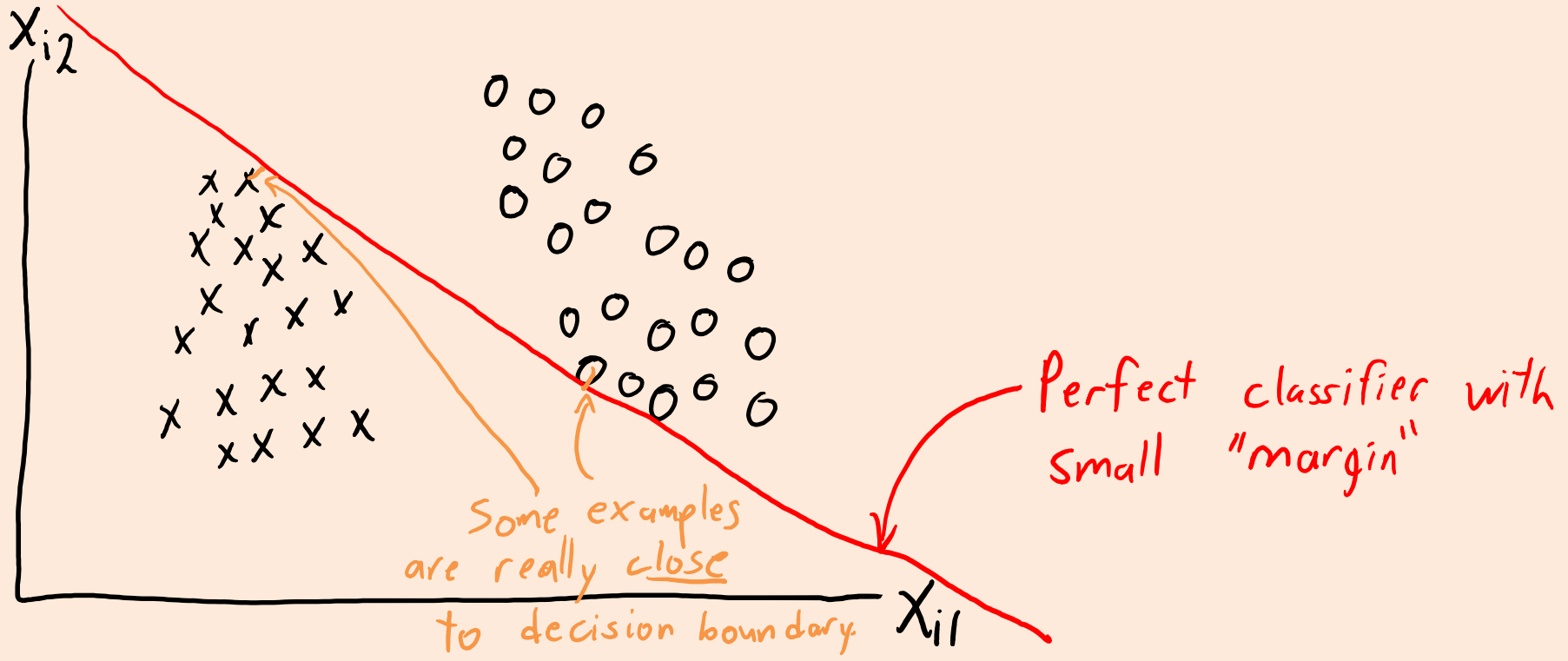
Maximum-Margin Perspective

- Consider a linearly-separable dataset.
  - Perceptron algorithm finds *some* classifier with zero error.
  - But are all zero-error classifiers equally good?



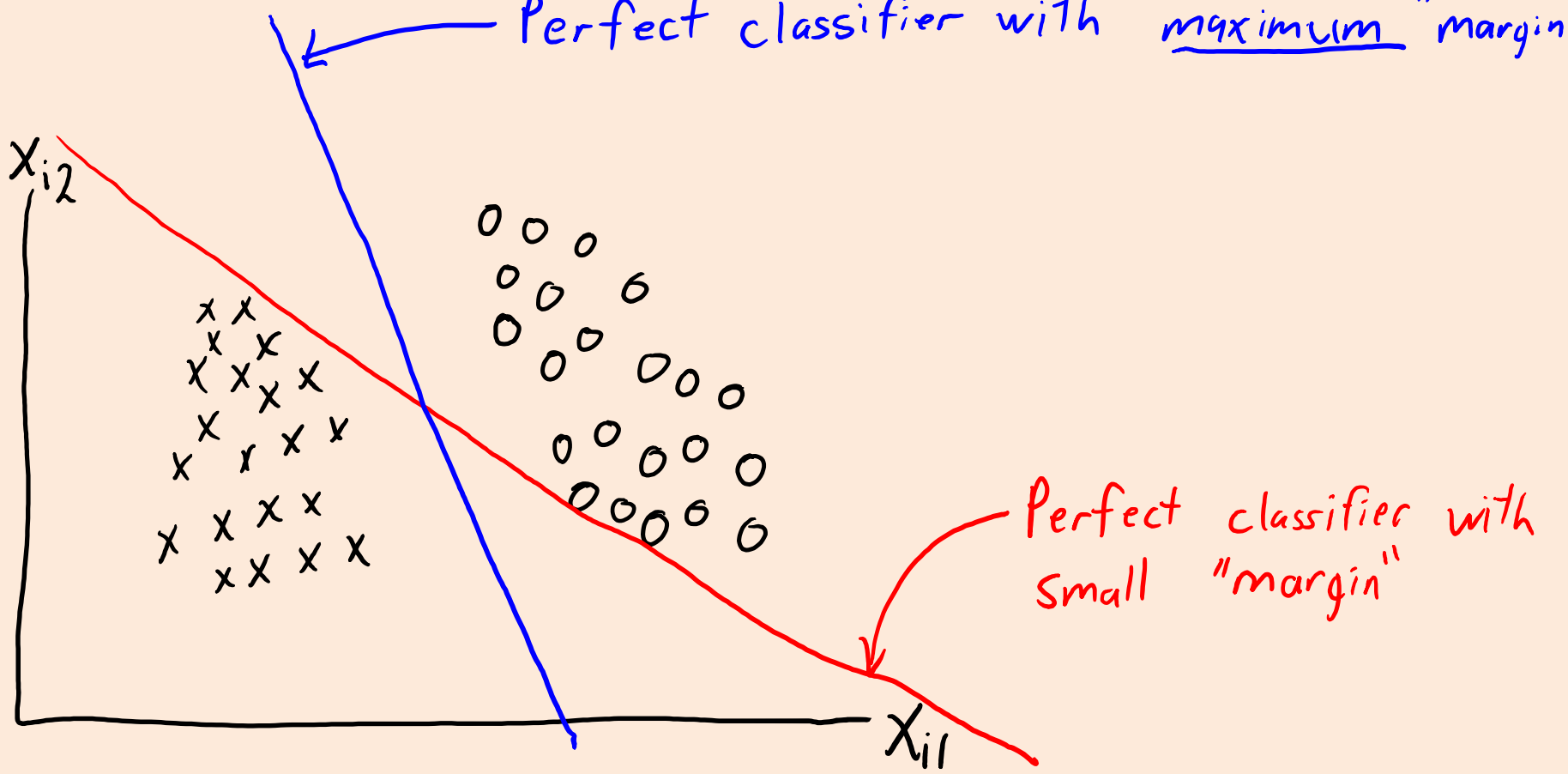
Maximum-Margin Perspective

- Consider a linearly-separable dataset.
  - Maximum-margin classifier: choose the farthest from both classes.



Maximum-Margin Perspective

- Consider a linearly-separable dataset.
  - Maximum-margin classifier: choose the farthest from both classes.

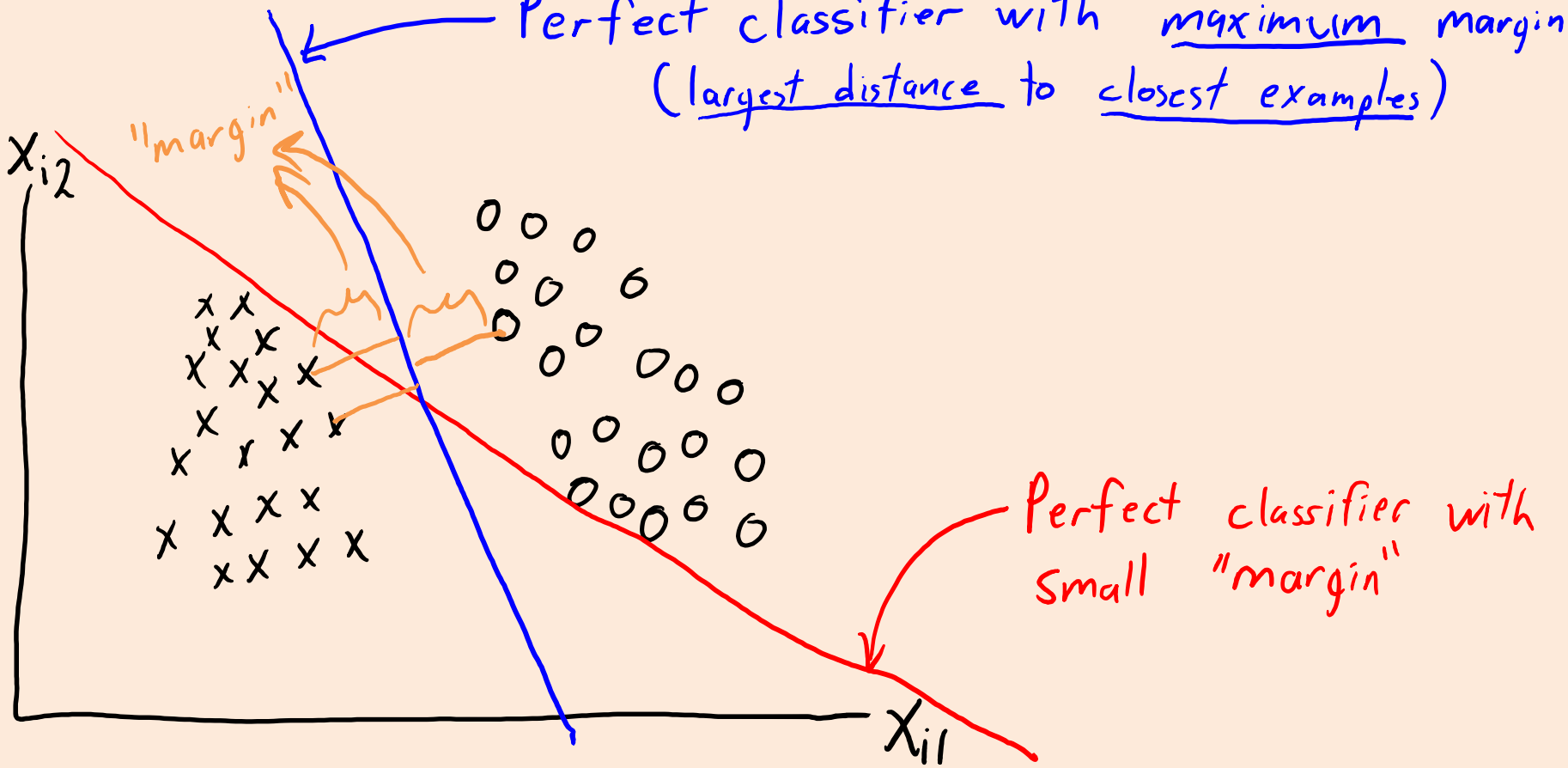


Maximum-Margin Perspective

- Consider a linearly-separable dataset.
  - Maximum-margin classifier: choose the farthest from both classes.

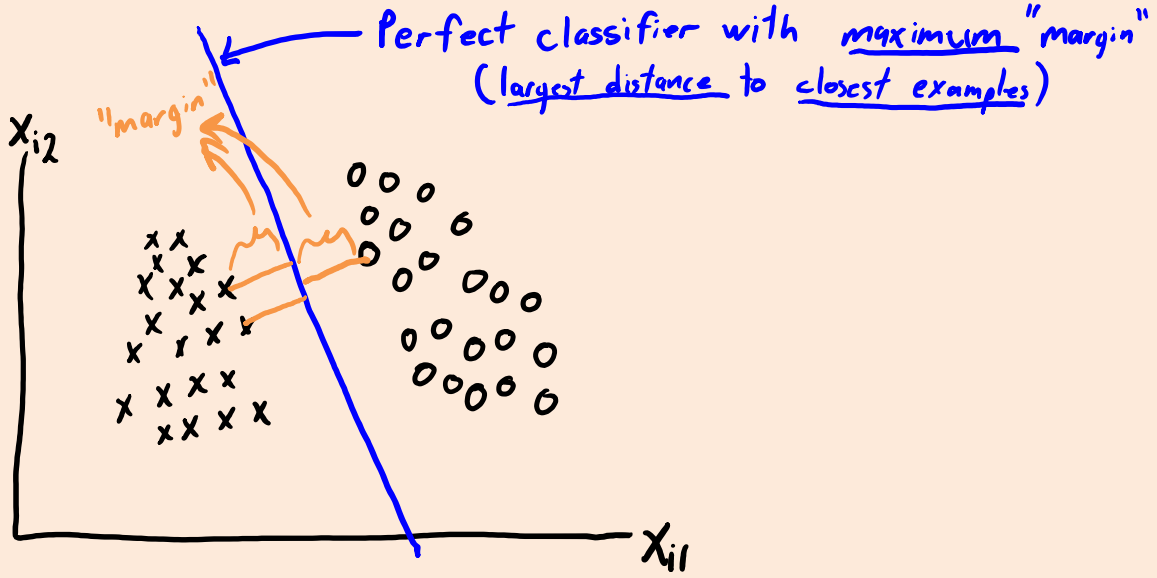
Why maximize margin?

If test data is close to training data, then max margin leaves more "room" before we make an error.



Maximum-Margin Perspective

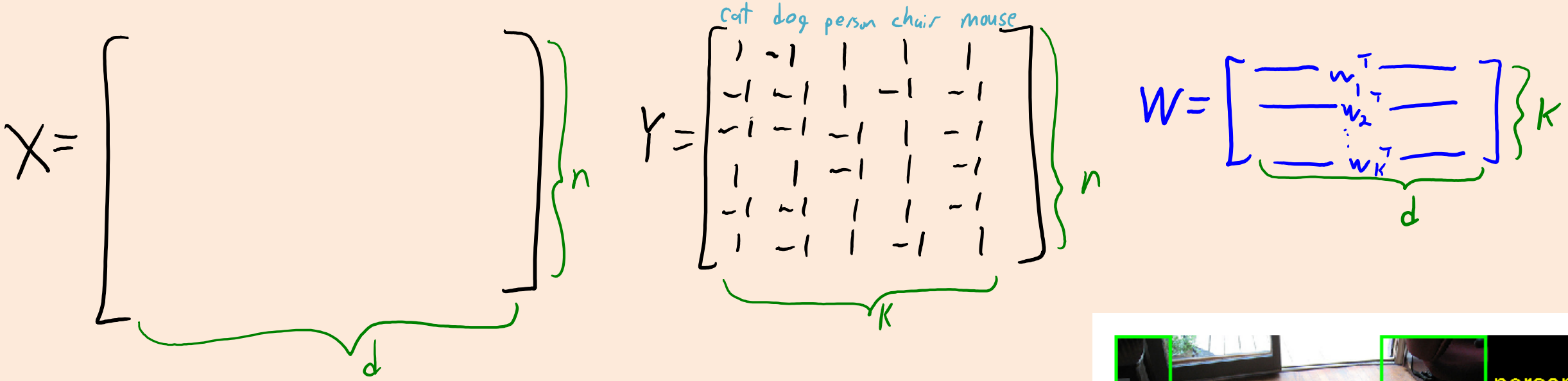
- For linearly-separable data:



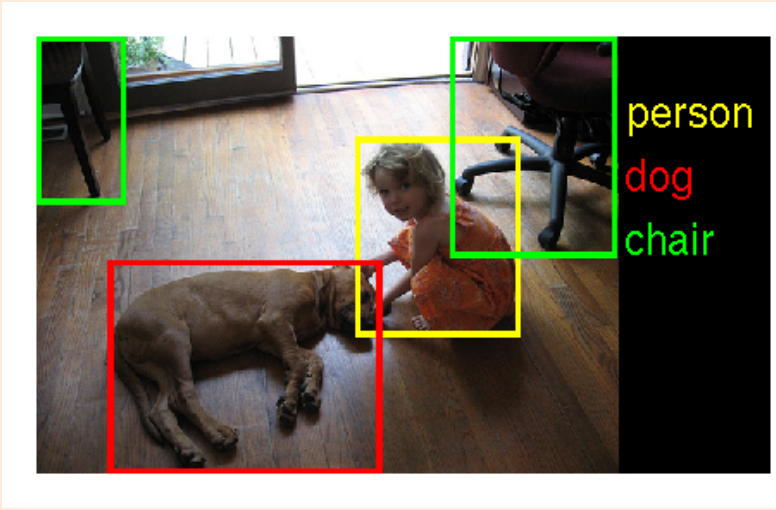
  - With small-enough $\lambda > 0$, SVMs find the maximum-margin classifier.
    - Need λ small enough that hinge loss is 0 in solution.
    - Origin of the name: the "support vectors" are the points closest to the line (see bonus).
  - Recent result: logistic regression also finds maximum-margin classifier.
    - With $\lambda=0$ and if you fit it with gradient descent (not true for many other optimizers).

Digression: Multi-Label Classification

- A related problem is multi-label classification:



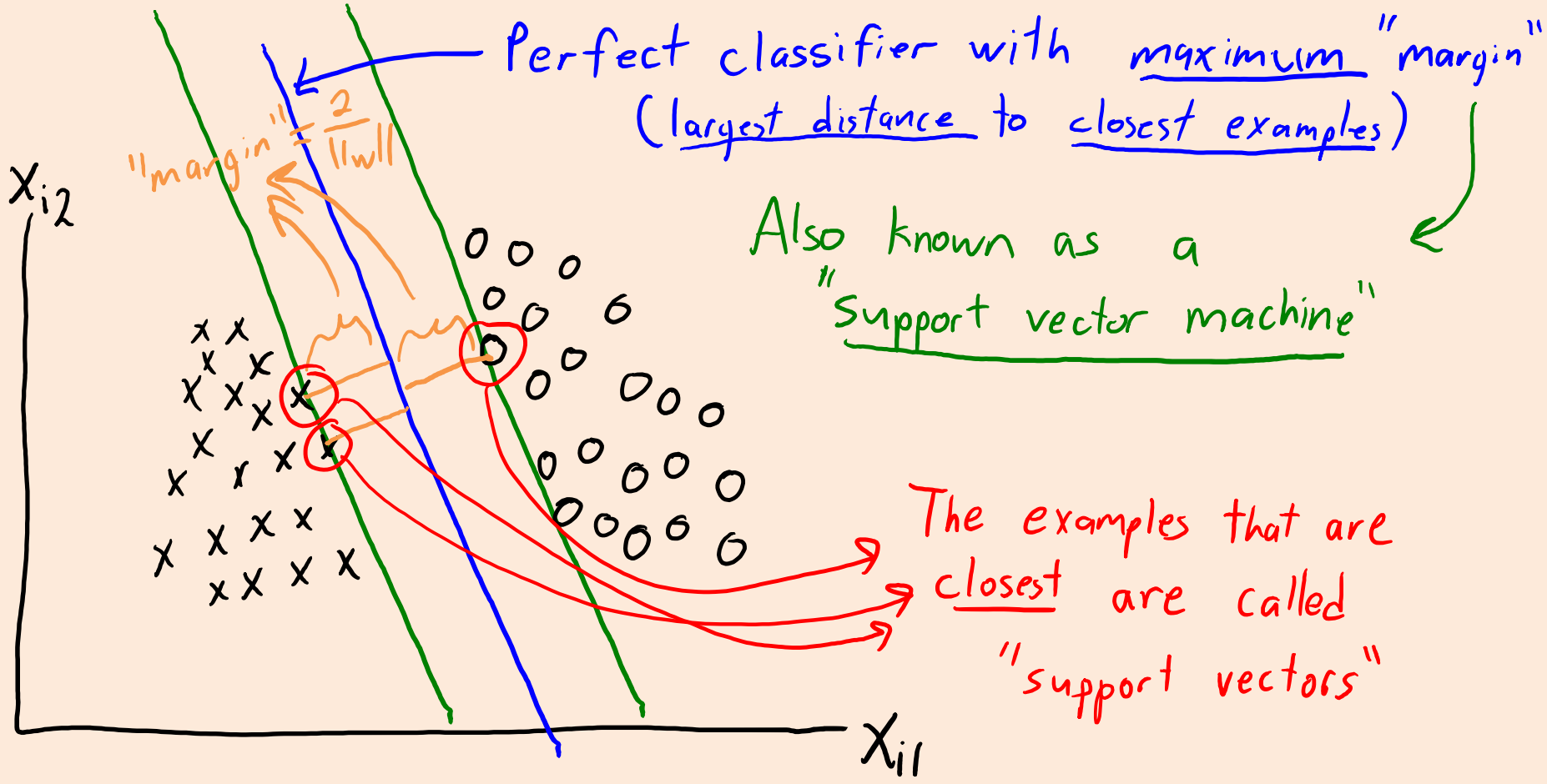
- Which of the 'k' objects are in this image?
  - There may be more than one "correct" class label.
  - Here we can also fit 'k' binary classifiers.
    - But we would take all the sign($w_c^T x_i$)=+1 as the labels.



http://image-net.org/challenges/LSVRC/2013/

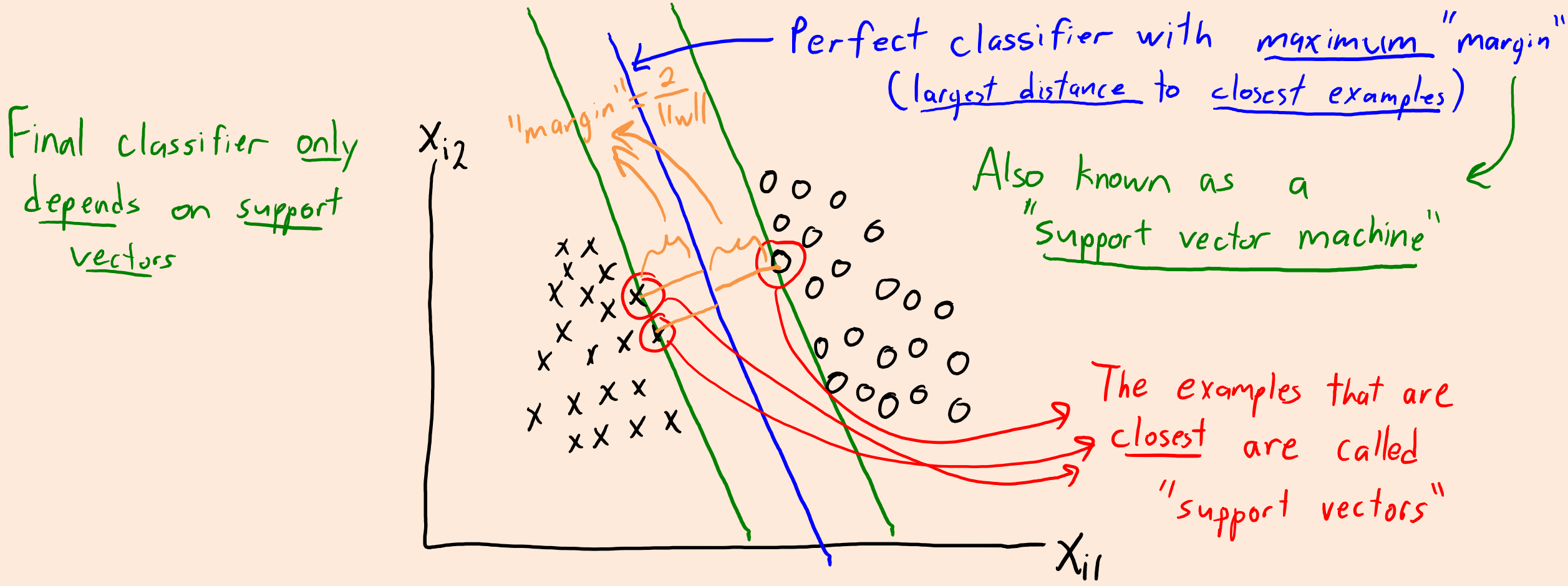
Maximum-Margin Classifier

- Consider a linearly-separable dataset.
  - Maximum-margin classifier: choose the farthest from both classes.



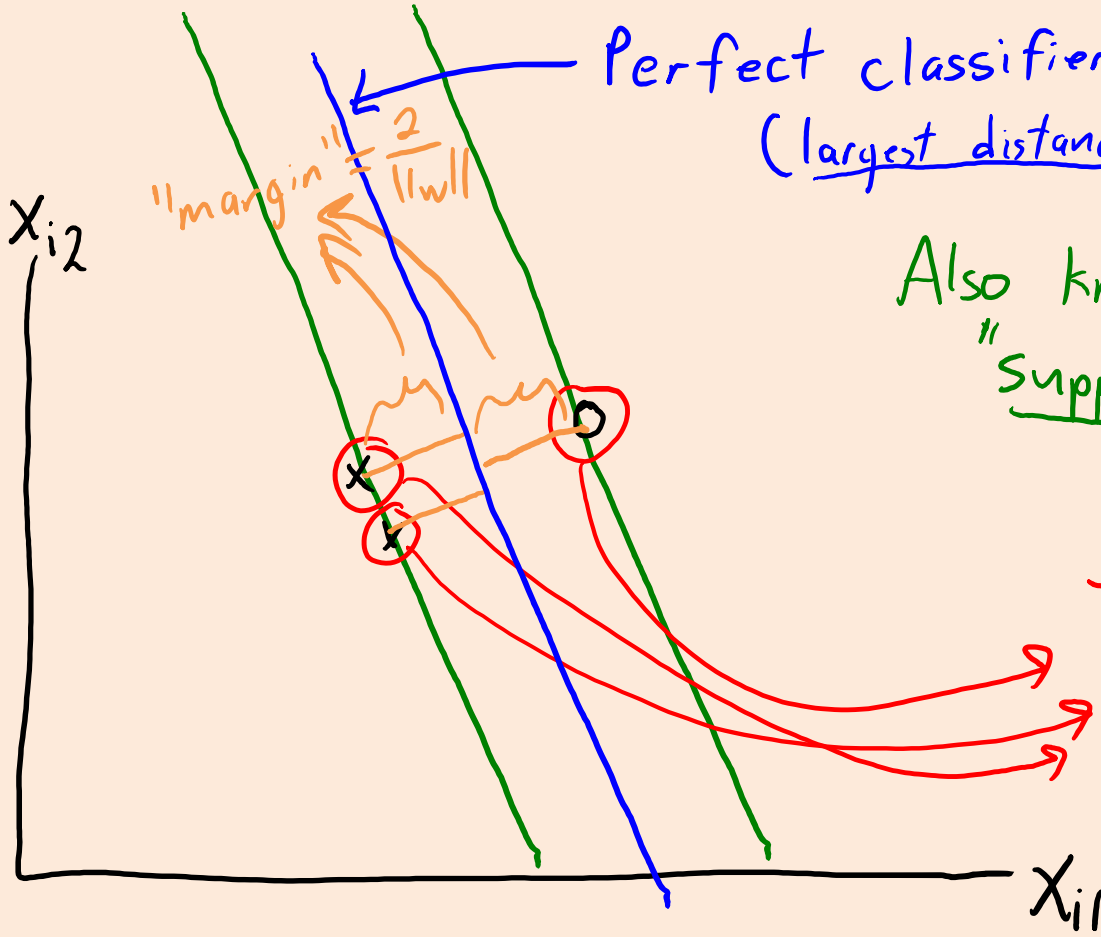
Maximum-Margin Classifier

- Consider a linearly-separable dataset.
  - Maximum-margin classifier: choose the farthest from both classes.



Maximum-Margin Classifier

- Consider a linearly-separable dataset.
  - Maximum-margin classifier: choose the farthest from both classes.



Perfect classifier with maximum "margin" (largest distance to closest examples)

Also known as a "support vector machine"

Final classifier only depends on support vectors

You could throw away the other examples and get the same classifier.

The examples that are closest are called "support vectors"

Support Vector Machines

- For linearly-separable data, SVM minimizes:

$$f(w) = \frac{1}{2}\|w\|^2$$ (equivalent to maximizing margin $\frac{2}{\|w\|}$)

  - Subject to the constraints that: (see Wikipedia/textbooks)

$$w^T x_i \geq 1 \quad \text{for } y_i = 1$$
$$w^T x_i \leq -1 \quad \text{for } y_i = -1$$

(classify all examples correctly)

- But most data is not linearly separable.
- For non-separable data, try to minimize violation of constraints:

If $w^T x_i \leq -1$ and $y_i = -1$ then "violation" should be zero.

If $w^T x_i \geq -1$ and $y_i = -1$ then we "violate constraint" by $1 + w^T x_i$

⋮

→ Constraint violation is the hinge loss.

Support Vector Machines

- Try to maximizing margin and also minimizing constraint violation:

$$f(w) = \sum_{i=1}^{n} \max\{0, 1 - y_i w^T x_i\} + \frac{1}{2}\|w\|^2$$

Hinge loss for example 'i':

if's the amount we violate "slack" $y_i w^T x_i \geq 1$

Original SVM objective: encourages large margin.

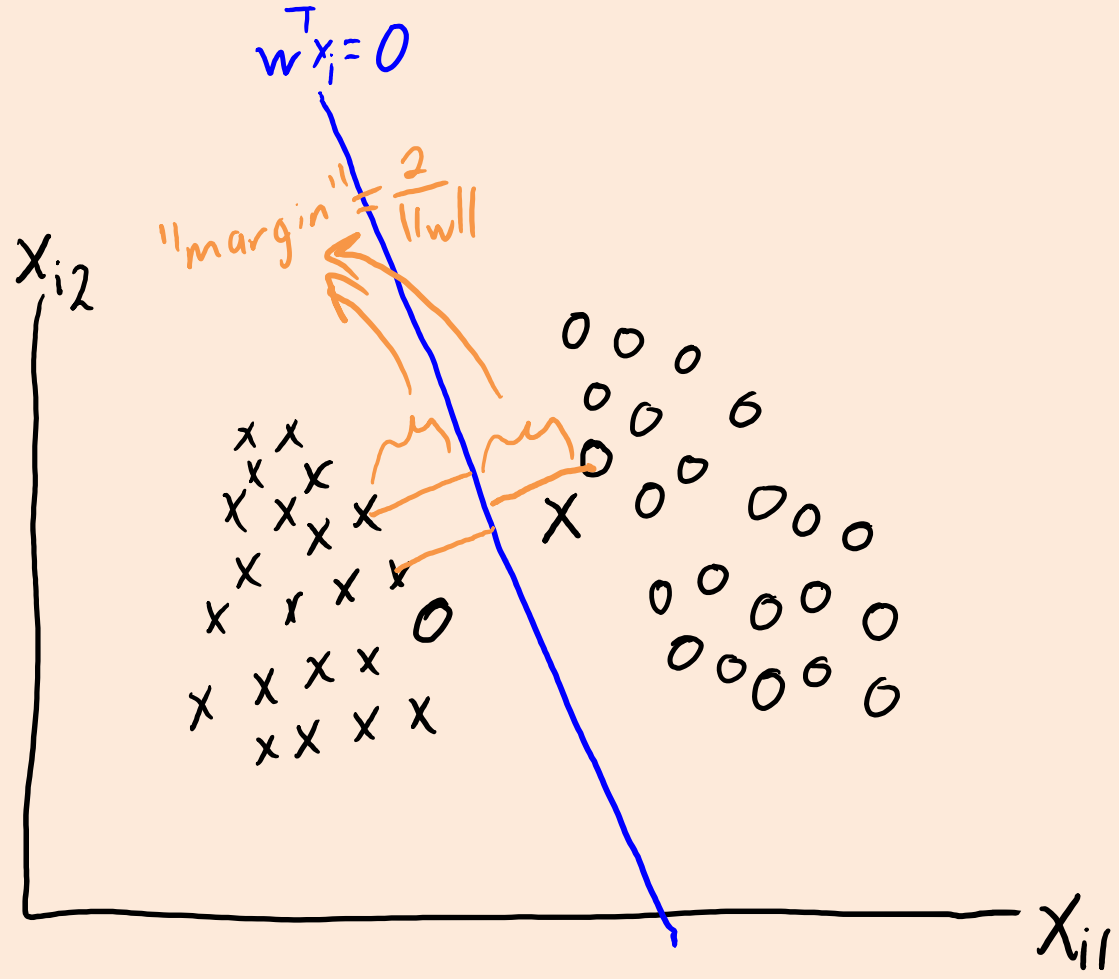
- We typically control margin/violation trade-off with parameter "λ":

$$f(w) = \sum_{i=1}^{n} \max\{0, 1 - y_i w^T x_i\} + \frac{\lambda}{2}\|w\|^2$$

- This is the standard SVM formulation (L2-regularized hinge).
  - Some formulations use $\lambda = 1$ and multiply hinge by 'C' (equivalent).

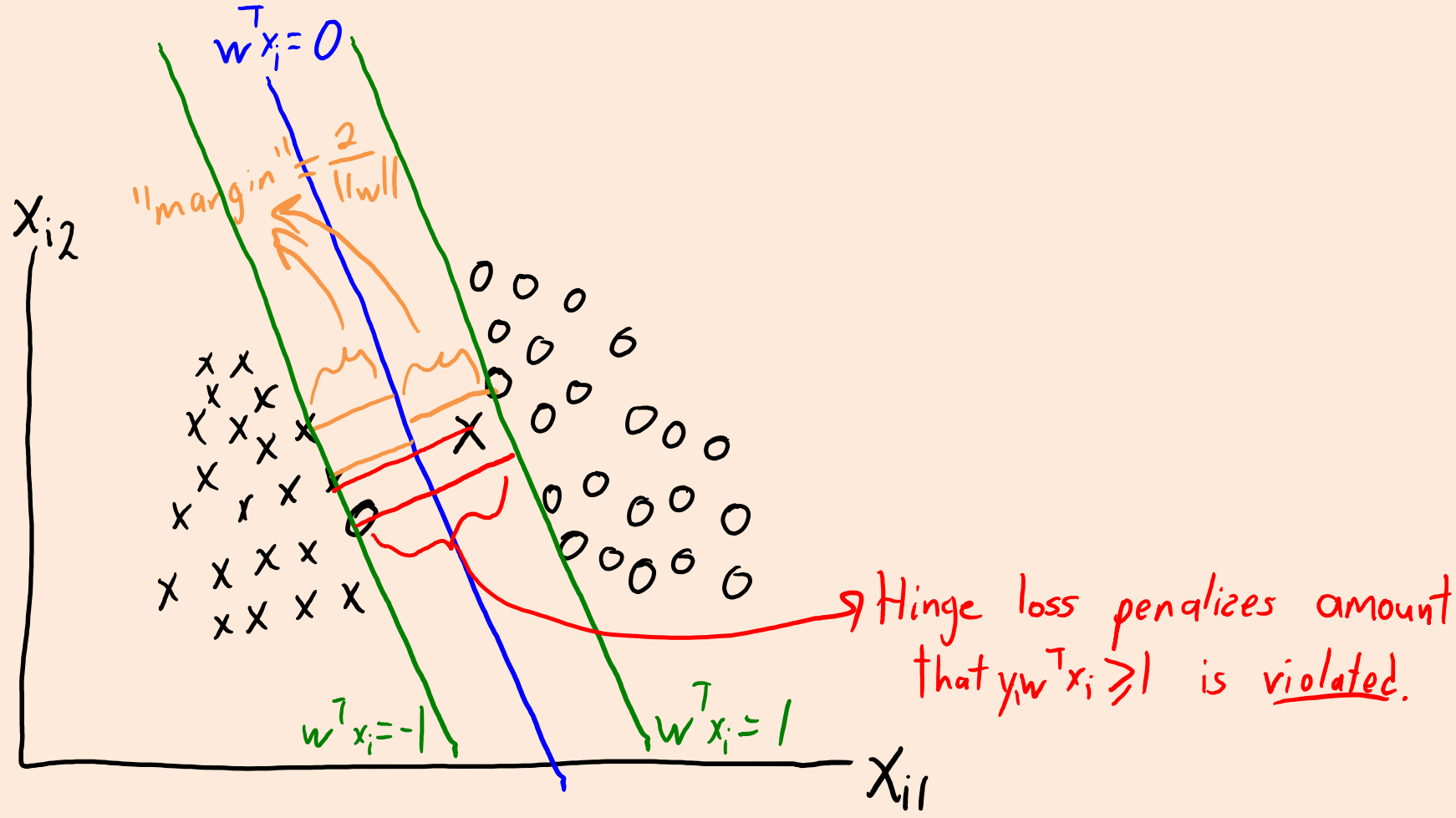
Support Vector Machines for Non-Separable

- Non-separable case:



Support Vector Machines for Non-Separable

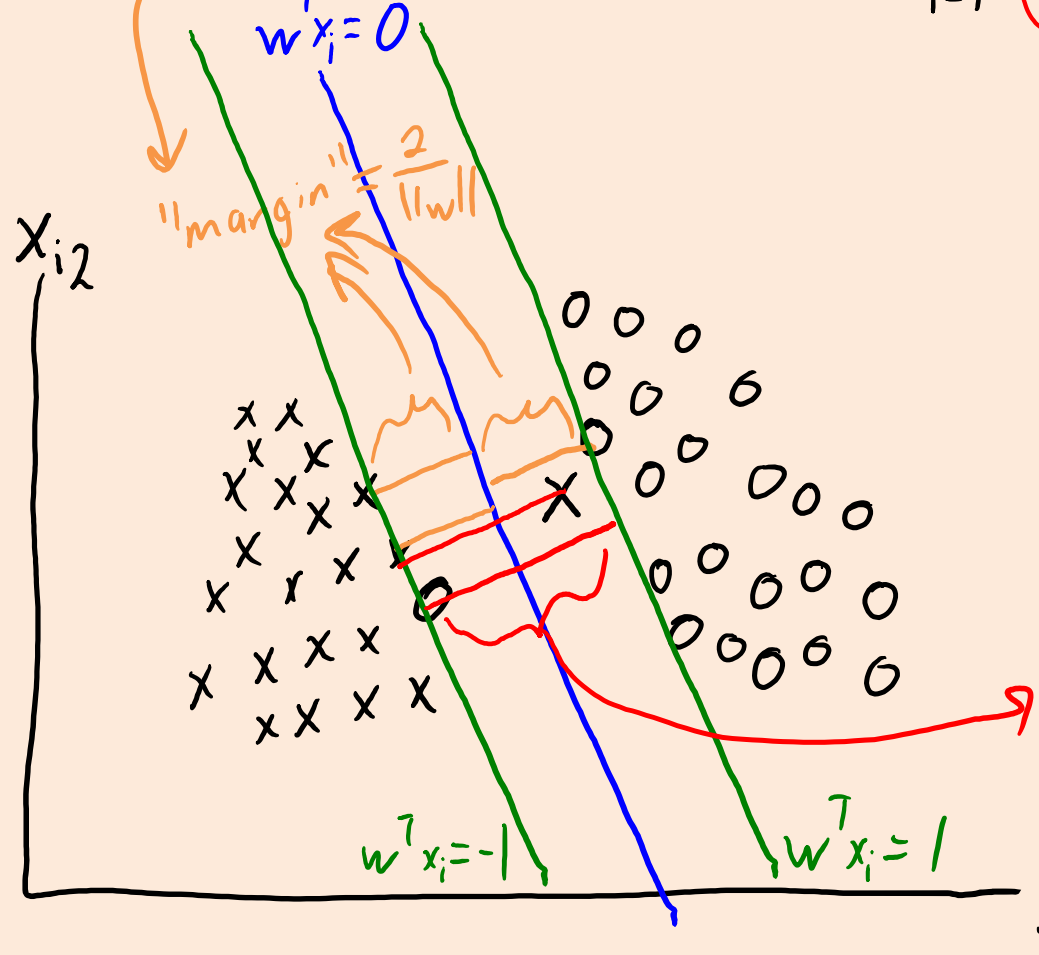
- Non-separable case:



Support Vector Machines for Non-Separable

- Non-separable case:

$$f(w) = \sum_{i=1}^{n} \max\{0, 1 - y_i w^T x_i\} + \frac{\lambda}{2}\|w\|^2$$



Logistic regression can be viewed as smooth approximation to SVMs.

But, no concept of "support vectors" with logistic loss.

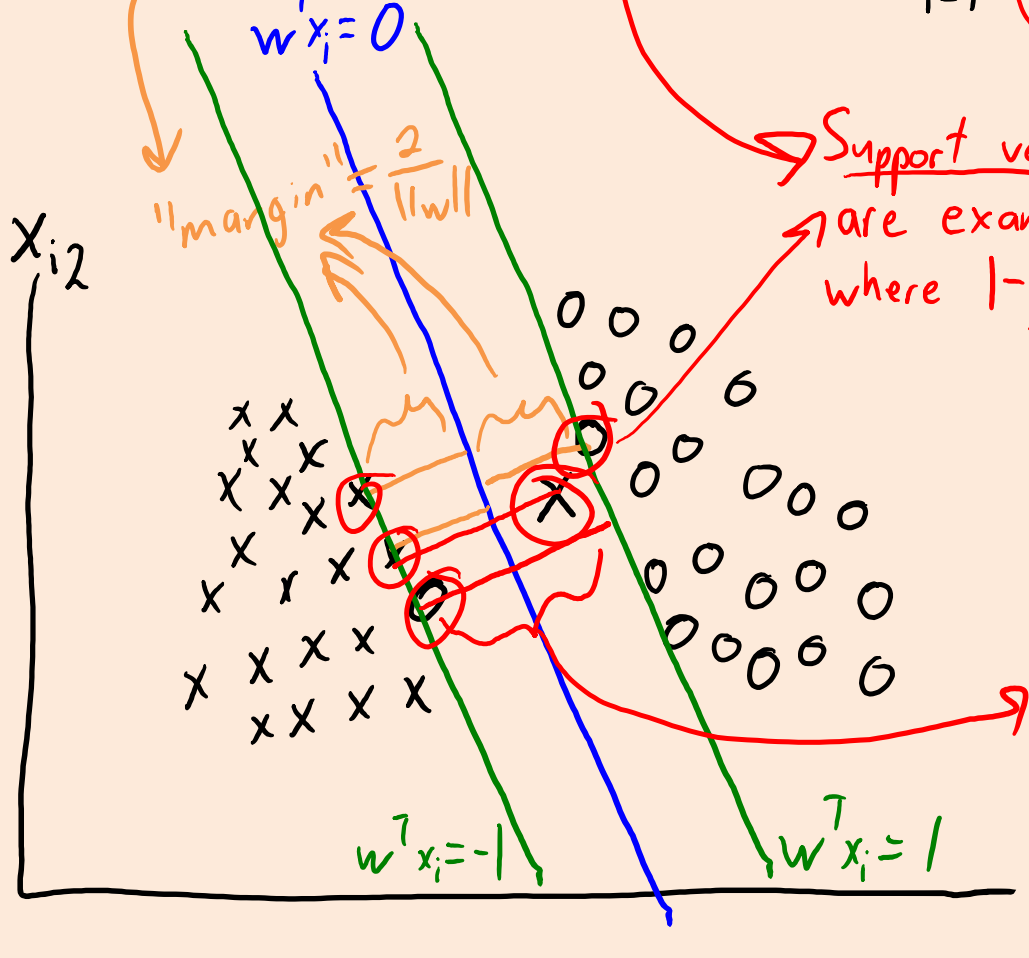
λ controls trade-off between having large margin and classifying examples correctly.

Hinge loss penalizes amount that $y_i w^T x_i \geq 1$ is violated.

Support Vector Machines for Non-Separable

- Non-separable case:

$$f(w) = \sum_{i=1}^{n} \max\{0, 1 - y_i w^T x_i\} + \frac{\lambda}{2}\|w\|^2$$



Support vectors are examples 'i' where $1 - y_i w^T x_i \geq 0$

λ controls trade-off between having large margin and classifying examples correctly.

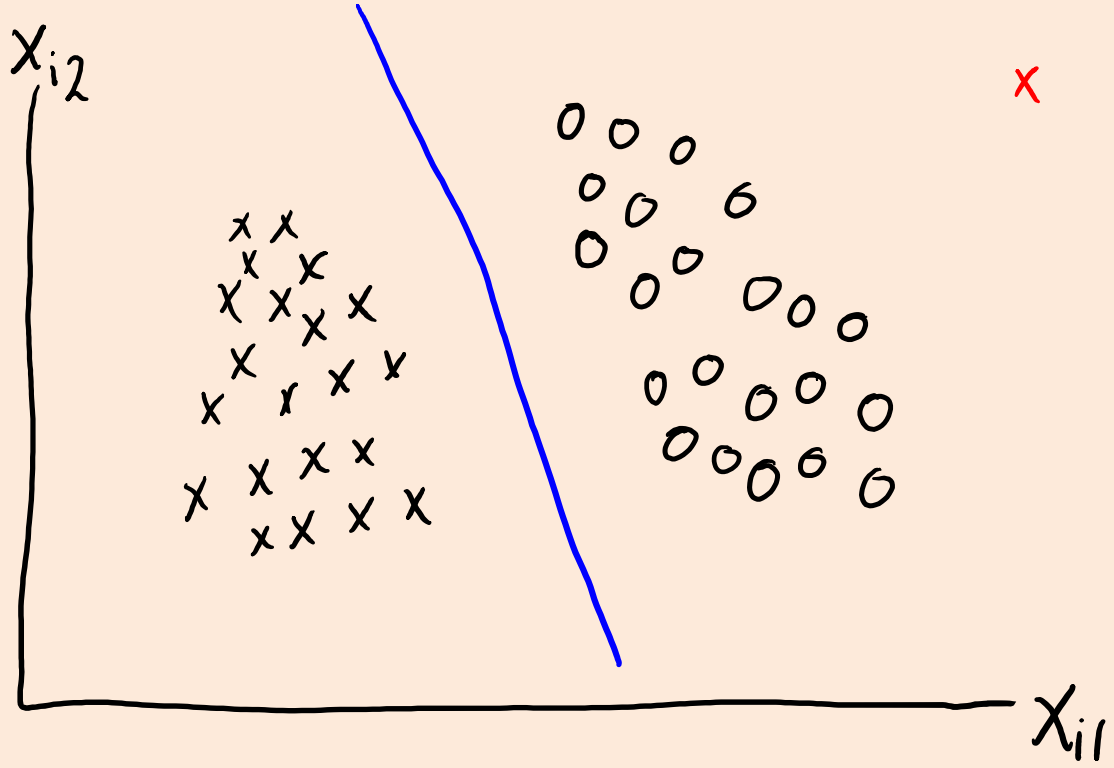
Hinge loss penalizes amount that $y_i w^T x_i \geq 1$ is violated.

Logistic regression can be viewed as smooth approximation to SVMs.

But, no concept of "support vectors" with logistic loss.

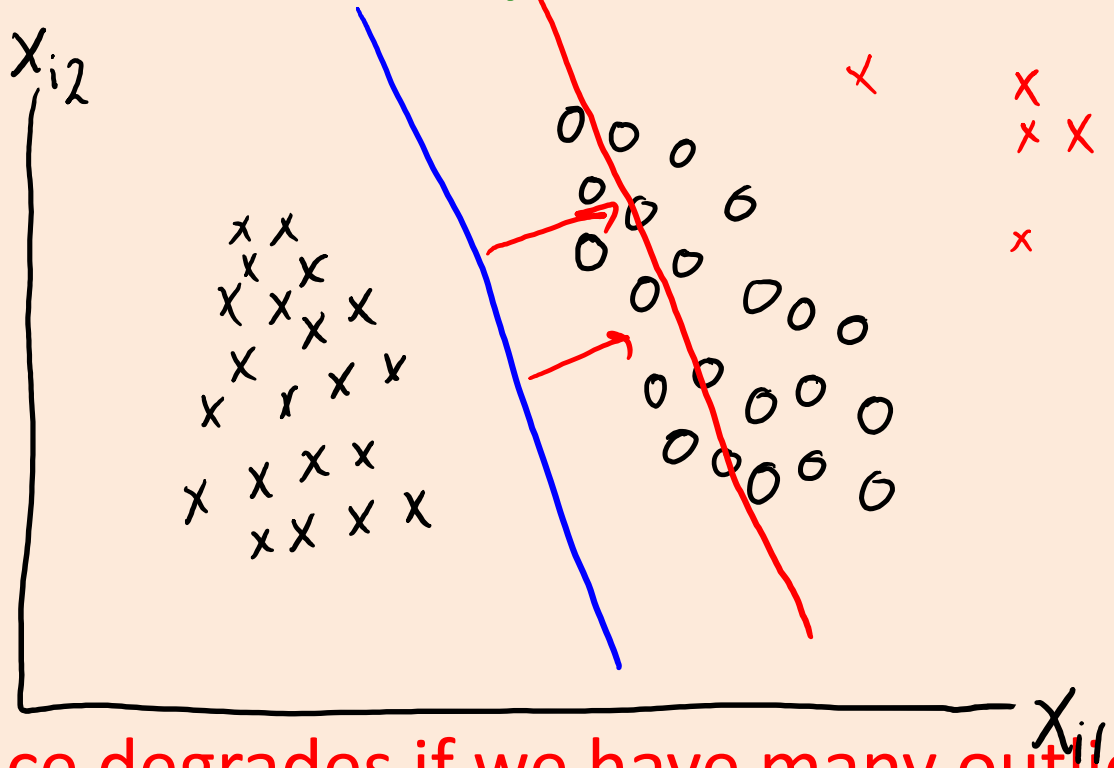
Robustness and Convex Approximations

- Because the hinge/logistic grow like absolute value for mistakes, they tend not to be affected by a small number of outliers.



Robustness and Convex Approximations

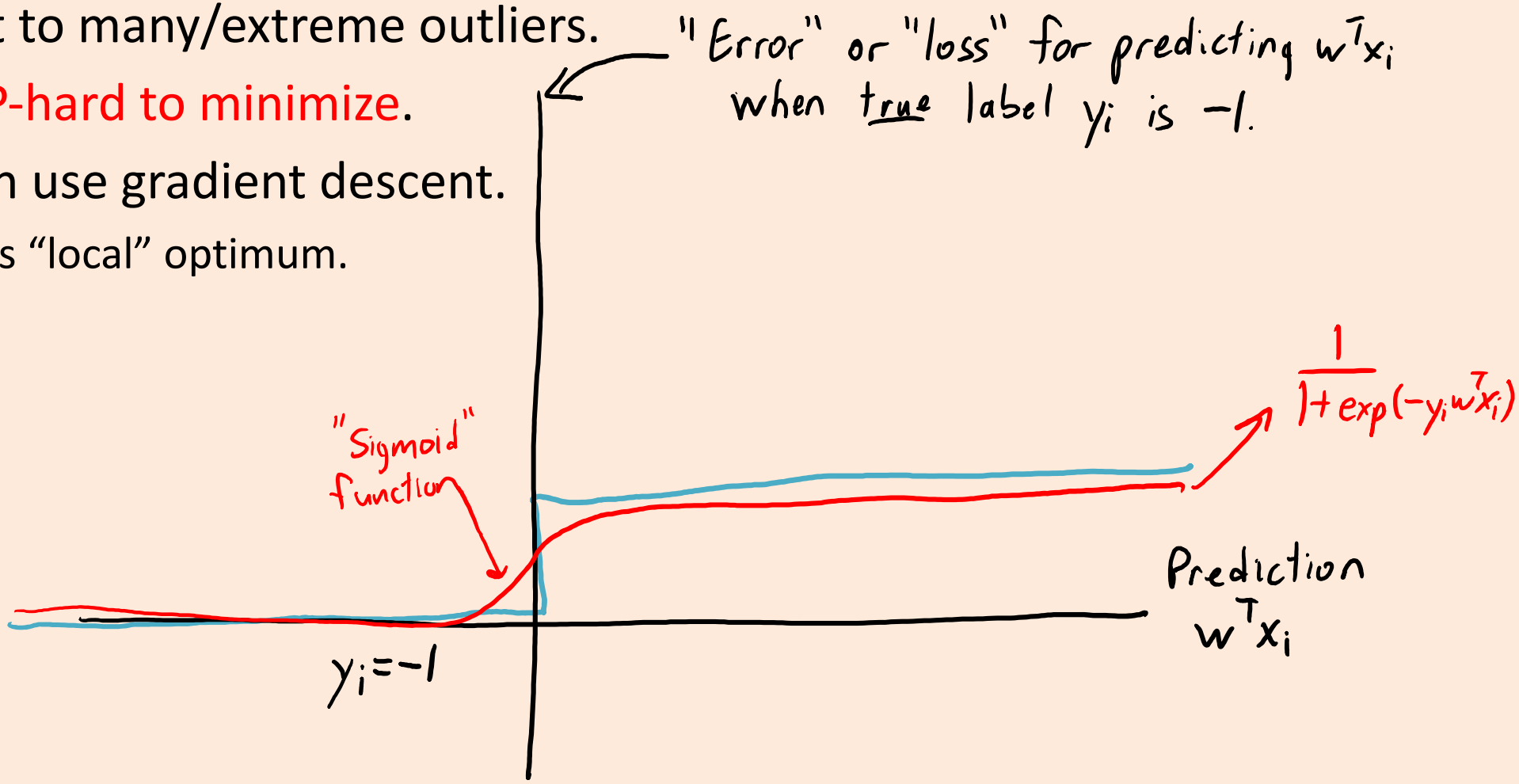
- Because the hinge/logistic grow like absolute value for mistakes, they tend not to be affected by a small number of outliers.



- But performance degrades if we have many outliers.

Non-Convex 0-1 Approximations

- There exists some smooth non-convex 0-1 approximations.
  - Robust to many/extreme outliers.
  - Still NP-hard to minimize.
  - But can use gradient descent.
    - Finds "local" optimum.



“Robust” Logistic Regression

- A recent idea: add a “fudge factor” v_i for each example.

$$f(w,v) = \sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i w^T x_i + v_i\right)\right)$$

- If $w^T x_i$ gets the sign wrong, we can “correct” the mis-classification by modifying v_i.
  - This makes the training error lower but doesn’t directly help with test data, because we won’t have the v_i for test data.
  - But having the v_i means the ‘w’ parameters don’t need to focus as much on outliers (they can make $|v_i|$ big if $\text{sign}(w^T x_i)$ is very wrong).

"Robust" Logistic Regression

- A recent idea: add a "fudge factor" v_i for each example.

$$f(w,v) = \sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i w^T x_i + v_i\right)\right)$$

- If $w^T x_i$ gets the sign wrong, we can "correct" the mis-classification by modifying v_i.
- A problem is that we can ignore the 'w' and get a tiny training error by just updating the v_i variables.
- But we want most v_i to be zero, so "robust logistic regression" puts an L1-regularizer on the v_i values:

$$f(w,v) = \sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i w^T x_i + v_i\right)\right) + \lambda \|v\|_1$$

- You would probably also want to regularize the 'w' with different λ.

“All-Pairs” and ECOC Classification

- Alternative to “one vs. all” to convert binary classifier to multi-class is “all pairs”.
  - For each pair of labels ‘c’ and ‘d’, fit a classifier that predicts +1 for examples of class ‘c’ and -1 for examples of class ‘d’ (so each classifier only trains on examples from two classes).
  - To make prediction, take a vote of how many of the (k-1) classifiers for class ‘c’ predict +1.
  - Often works better than “one vs. all”, but not so fun for large ‘k’.
- A variation on this is using “error correcting output codes” from information theory (see Math 342).
  - Each classifier trains to predict +1 for some of the classes and -1 for others.
  - You setup the +1/-1 code so that it has an “error correcting” property.
    - It will make the right decision even if some of the classifiers are wrong.

Motivation: Dog Image Classification

- Suppose we're classifying images of dogs into breeds:

- What if we have images where class label isn't obvious?
  - Syberian husky vs. Inuit dog?

https://www.slideshare.net/angjoo/dog-breed-classification-using-part-localization
https://ischlag.github.io/2016/04/05/important-ILSVRC-achievements

Learning with Preferences

- Do we need to throw out images where label is ambiguous?
  - We don't have the y_i.

  - We want classifier to prefer Syberian husky over bulldog, Chihuahua, etc.
    - Even though we don't know if these are Syberian huskies or Inuit dogs.

  - Can we design a loss that enforces preferences rather than "true" labels?

https://ischlag.github.io/2016/04/05/important-ILSVRC-achievements

Learning with Pairwise Preferences (Ranking)

- Instead of y_i, we're given list of (c_1,c_2) preferences for each 'i':

We want $w_{c_1}^T x_i > w_{c_2}^T x_i$ for these particular (c_1,c_2) values

- Multi-class classification is special case of choosing (y_i,c) for all 'c'.

- By following the earlier steps, we can get objectives for this setting:

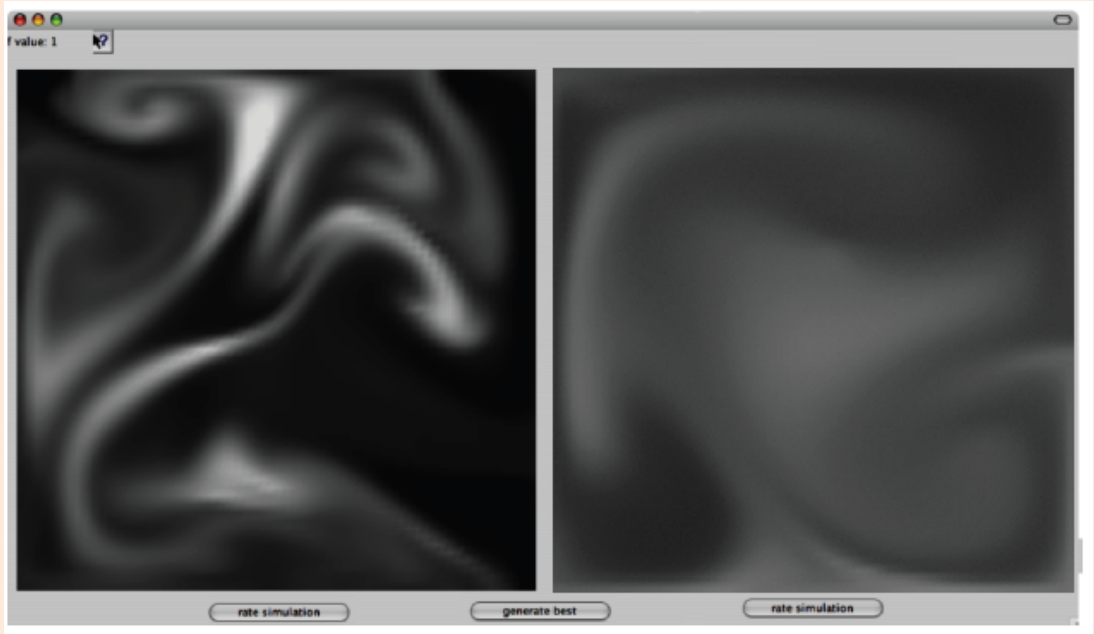
$$\sum_{i=1}^{n} \sum_{(c_1,c_2)} \max\{0, 1 - w_{c_1}^T x_i + w_{c_2}^T x_i\} + \frac{\lambda}{2} \|W\|_F^2$$

"sum" version of multi-class SVM

https://ischlag.github.io/2016/04/05/important-ILSVRC-achievements

Learning with Pairwise Preferences (Ranking)

- Pairwise preferences for computer graphics:
  - We have a smoke simulator, with several parameters:

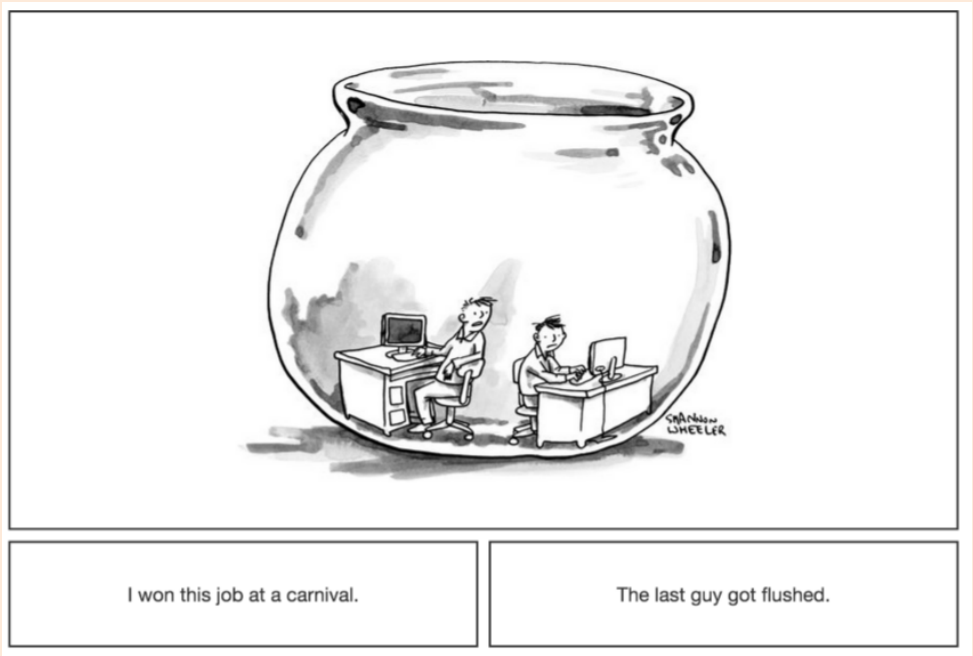


  - Don’t know what the optimal parameters are, but we can ask the artist:
    - “Which one looks more like smoke”?

https://circle.ubc.ca/bitstream/handle/2429/30519/ubc_2011_spring_brochu_eric.pdf?sequence=3

Learning with Pairwise Preferences (Ranking)

- Pairwise preferences for humour:
  - New Yorker caption contest:



  - “Which one is funnier”?

https://homes.cs.washington.edu/~jamieson/resources/next.pdf

Risk Scores

- In medicine/law/finance, risk scores are sometimes used to give probabilities:

1.	**C**ongestive Heart Failure	1 point	...
2.	**H**ypertension	1 point	+ ...
3.	**A**ge $\geq$ 75	1 point	+ ...
4.	**D**iabetes Mellitus	1 point	+ ...
5.	Prior **S**troke or Transient Ischemic Attack	**2** points	+
		SCORE	=

SCORE	0	1	2	3	4	5	6
RISK	1.9%	2.8%	4.0%	5.9%	8.5%	12.5%	18.2%

Figure 1: $CHADS_2$ risk score of Gage et al. (2001) to assess stroke risk (see www.mdcalc.com for other medical scoring systems). The variables and points of this model were determined by a panel of experts, and the risk estimates were computed empirically from data.

- Get integer-valued "points" for each "risk factor", and probability is computed from data based on people with same number of points.
- Less accurate than fancy models, but interpretable and can be done by hand.
  - Some work on trying to "learn" the whole thing (like doing feature selection then rounding).

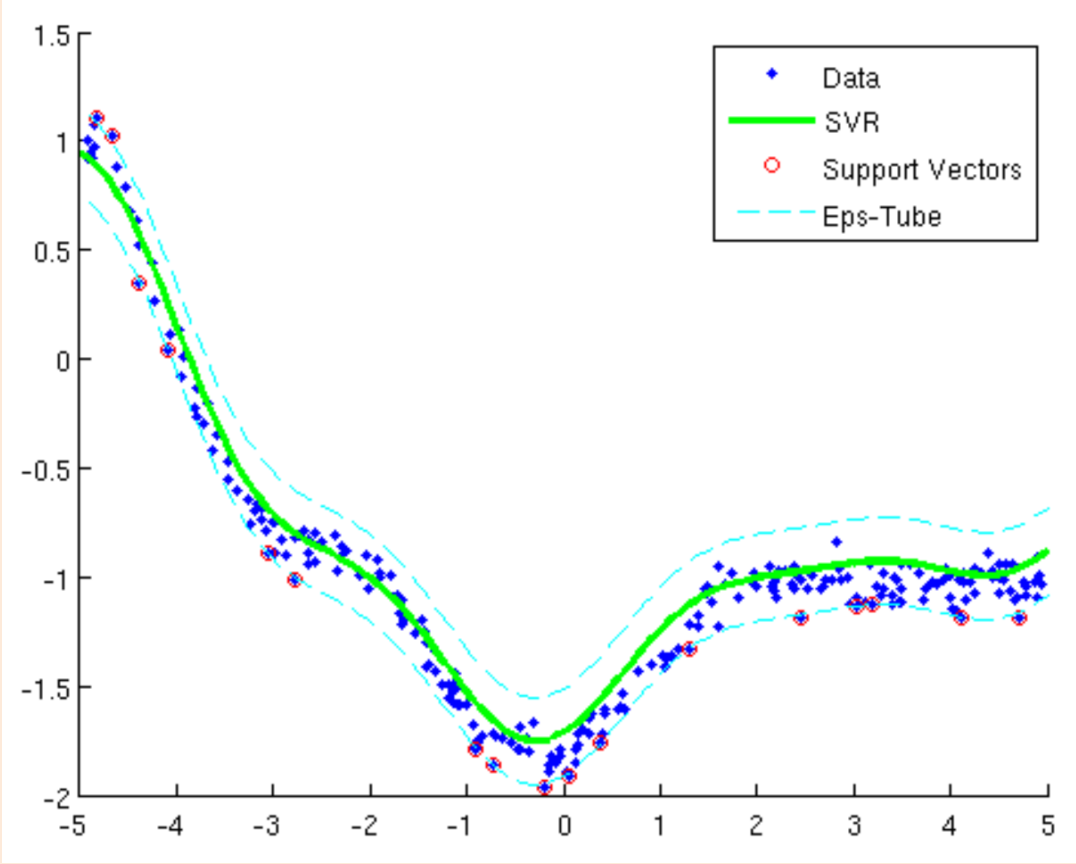
https://arxiv.org/pdf/1610.00168.pdf

Support Vector Regression

- Support vector regression objective (with hyper-parameter ϵ):

$$f(w) = \sum_{i=1}^{n} \max\{0, |w^T x_i - y_i| - \epsilon\} + \frac{\lambda}{2}\|w\|^2$$

  - Looks like L2-regularized robust regression with the L1-loss.
  - But have loss of 0 if $\hat{y}_i$ within ϵ of $\tilde{y}_i$.
    - So doesn't try to fit data exactly.
      - This can help fight overfitting.
  - Support vectors are points with loss>0.
    - Points outside the "epsilon-tube".
  - Example with Gaussian-RBFs as features:

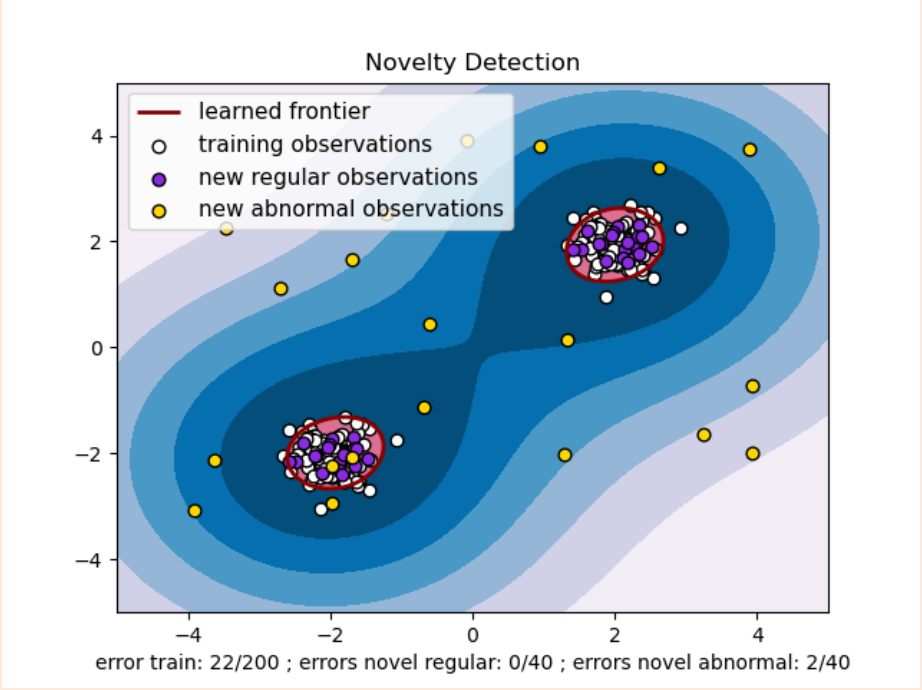


1-Class SVMs

- 1-class SVMs for outlier detection.

$$f(w, w_0) = \sum_{i=1}^{N} [\max\{0, w_0 - w^T x_i\} - w_0] + \frac{\lambda}{2}\|w\|_2^2$$

  - Variables are ‘w’ (vector) and ‘w_0’ (scalar).
  - Only trains on “inliers”.
    - Tries to make $w^T x_i$ bigger than w_0 for inliers.
    - At test time: says “outlier” if $w^T x_i < w_0$.
    - Usually used with RBFs.



https://scikit-learn.org/stable/auto_examples/svm/plot_oneclass.html