CPSC 340/540 Tutorial 8

Winter 2024 Term 1

T1A: Tuesday 16:00-17:00;
T1C: Thursday 10:00-11:00;
Office Hour: Wednesday 15:00-16:00

Slides can be found at Piazza and my personal page after T1C.

Yi (Joshua) Ren

https://joshua-ren.github.io/
renyi.joshua@gmail.com
PhD with Danica

Machine Learning:
Learning dynamics, LLM, Compositional Generalization

More helpful on theory

Less helpful on coding

Slides Credit: To various pervious TA’s of this course

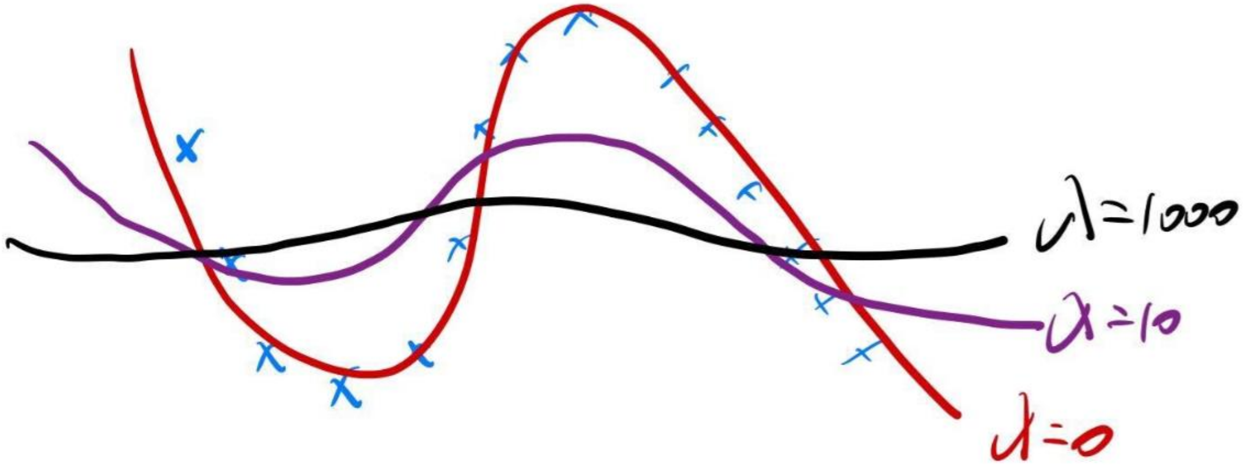
Sorry for canceling this week’s tutorial due to some health issues.

- Regularization
- RBF
- Linear Classifiers

Regularization: start from L2-norm

$$f(\boldsymbol{w};\boldsymbol{x}) = \left|\left|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{Y}\right|\right|_2^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

- Already explained it from “feature selection” perspective (L1, L0)
- Here we understand from the model’s capacity (L2)



$$\boldsymbol{y} = \boldsymbol{w}^T\boldsymbol{x}$$

$$\|\boldsymbol{y}\| = \left\|\boldsymbol{w}^T\boldsymbol{x}\right\| \leq \|\boldsymbol{w}\| * \|\boldsymbol{x}\|$$

Regularization: compare L0, L1, L2

L0- vs. L1- vs. L2-Regularization

	Sparse 'w' (Selects Features)	Speed	Unique 'w'	Coding Effort	Irrelevant Features
L0-Regularization	Yes	Slow	No	Few lines	Not Sensitive
L1-Regularization	Yes*	Fast*	No	1 line*	Not Sensitive
L2-Regularization	No	Fast	Yes	1 line	A bit sensitive

Regularization: L2 v.s. L1

	Sparse 'w' (Selects Features)	Unique 'w'
L0-Regularization	Yes	No
L1-Regularization	Yes*	No
L2-Regularization	No	Yes

L2-Regularization vs. L1-Regularization

- L2-Regularization:
  - Insensitive to changes in data.
  - Decreased variance:
    - Lower test error.
  - Closed-form solution.
  - Solution is unique.
  - All 'w_j' tend to be non-zero.
  - Can learn with *linear* number of irrelevant features.
    - E.g., only O(d) relevant features.
- L1-Regularization:
  - Insensitive to changes in data.
  - Decreased variance:
    - Lower test error.
  - Requires iterative solver/subgradient methods.
  - Solution is not unique.
  - Many 'w_j' tend to be zero.
  - Can learn with **exponential** number of irrelevant features.
    - E.g., only O(log(d)) relevant features.

    Paper on this result by Andrew Ng

Regularization: L1 norm

L1-loss vs. L1-regularization

- Don't confuse the L1 loss with L1-regularization!
  - L1-loss is robust to outlier data points.
    - You can use this instead of removing outliers.
  - L1-regularization is robust to irrelevant features.
    - You can use this instead of removing features.
- And note that you can be robust to outliers and irrelevant features:

$$f(w) = \underbrace{\|Xw - y\|_1}_{L_1-\text{loss}} + \lambda \underbrace{\|w\|_1}_{L_1-\text{regularizer}}$$

- Can we smooth and use "Huber regularization"?
  - Huber regularizer is still robust to irrelevant features.
  - But it's the non-smoothness that sets weights to exactly 0.

Regularization: L0 and L1 regularization

In linear models, setting $w_j = 0$ is the same as removing feature 'j':

$$\hat{y}_i = w_1 x_{i1} + \underbrace{w_2 x_{i2}}_{\text{set } w_2 = 0} + w_3 x_{i3} + \cdots + w_d x_{id}$$

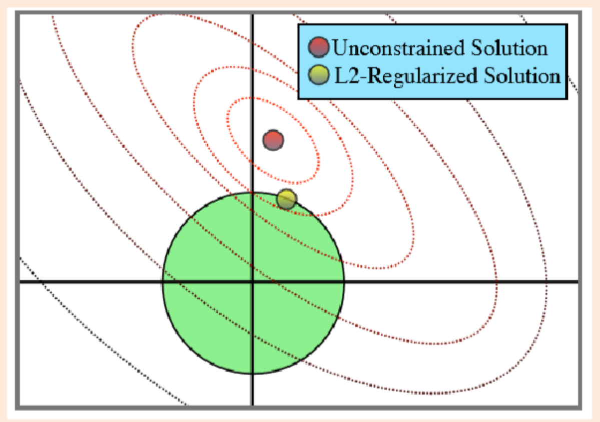
$$\hat{y}_i = w_1 x_{i1} + \underbrace{0}_{\text{ignore } x_{i2}} + w_3 x_{i3} + \cdots + w_d x_{id}$$

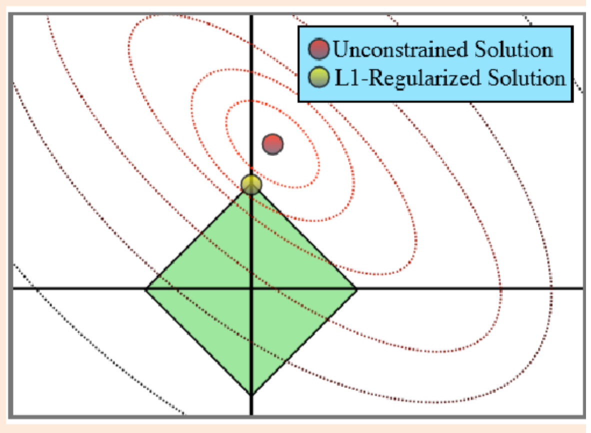
The L0 "norm" is the number of non-zero values ($||w||_0 = \text{size}(S)$).

If $w = \begin{bmatrix} 1 \\ 0 \\ 2 \\ 0 \\ 3 \end{bmatrix}$ then $||w||_0 = 3$ If $w = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ then $||w||_0 = 0$.

- But it's hard to find the 'w' minimizing this objective.
- We discussed forward selection, but requires fitting $O(d^2)$ models.
- If 'd' is large, forward selection is too slow:
  - For least squares, need to fit $O(d^2)$ models at cost of $O(nd^2 + d^3)$.
  - Total cost $O(nd^4 + d^5)$, and even if you are clever still costs $O(nd^2 + d^4)$.

	Speed
L0-Regularization	Slow
L1-Regularization	Fast*
L2-Regularization	Fast





- Regularization
- RBF
- Linear Classifiers

RBF: what is Gaussian Radial Basis Functions

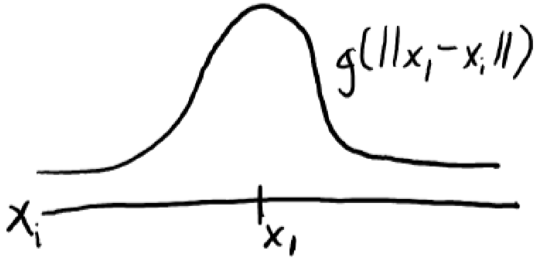
- What is a radial basis functions (RBFs)?
  - A set of non-parametric bases that depend on distances to training points.

distance of Feature vector to example 1

Distance to example 2

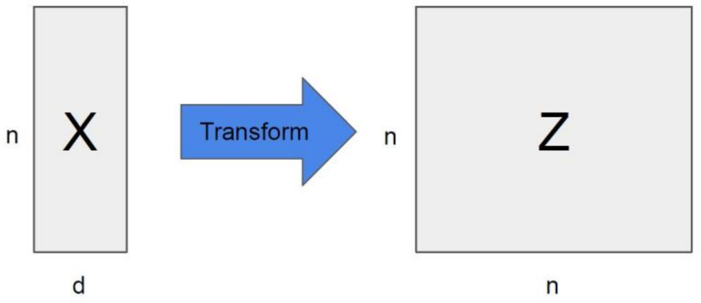
Replace $x_i = (\underbrace{x_{i1}, x_{i2}, \dots, x_{id}}_{\text{'d' features}})$ with $z_i = (\underbrace{g(\|x_i - x_1\|), g(\|x_i - x_2\|), \dots, g(\|x_i - x_n\|)}_{\text{'n' features}})$

  - Have 'n' features, with feature 'j' depending on distance to example 'i'.
    - Typically the feature will decrease as the distance increases:



RBF: implementation

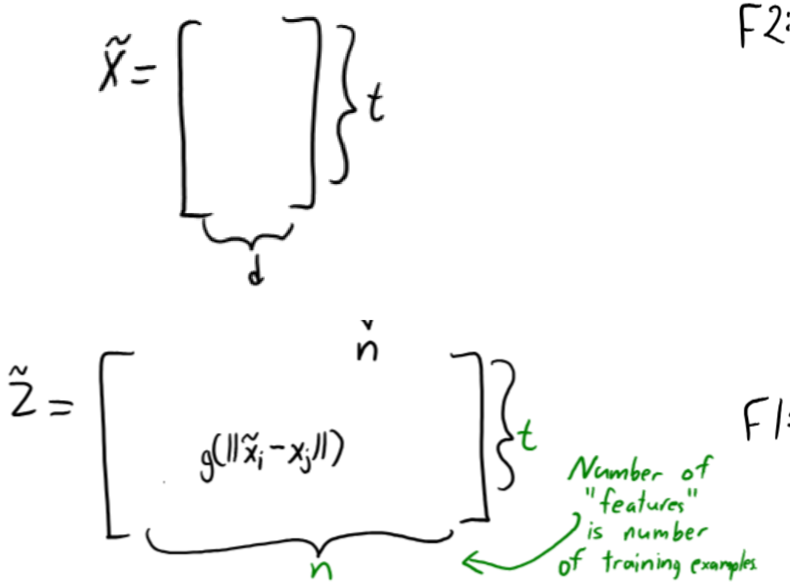
- Featue i is the “similarity” of the current example to the i-th data.



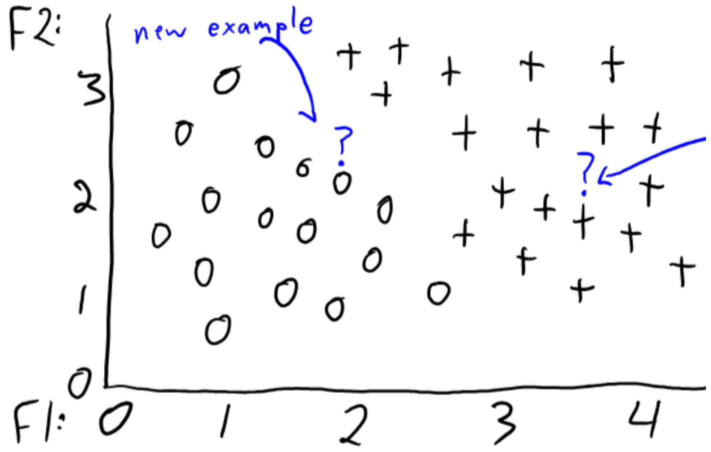
$$x_i = \begin{bmatrix} x_{i1} & x_{i2} & \cdots & x_{id} \end{bmatrix}$$
$$z_i = \begin{bmatrix} z_{i1} & z_{i2} & \cdots & z_{in} \end{bmatrix}$$

where

$$z_{ij} = g(||x_i - x_j||^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{||x_i - x_j||^2}{2\sigma^2})$$



- Also: Recall KNN



- Pseudo code:

```
z = np.zeros(n,n)
for i in range(n):
    for j in range(n):
        z[i,j] = np.exp(-norm(x[i,:], x[j,:])/2*eta^2)

z_test = np.zeros(t,n)
for i in range(t):
    for j in range(n):
        z_test[i,j] = np.exp(-norm(x_test[i,:], x[j,:])/2*eta^2)
```

RBF: implementation

RBFs, Regularization, and Validation

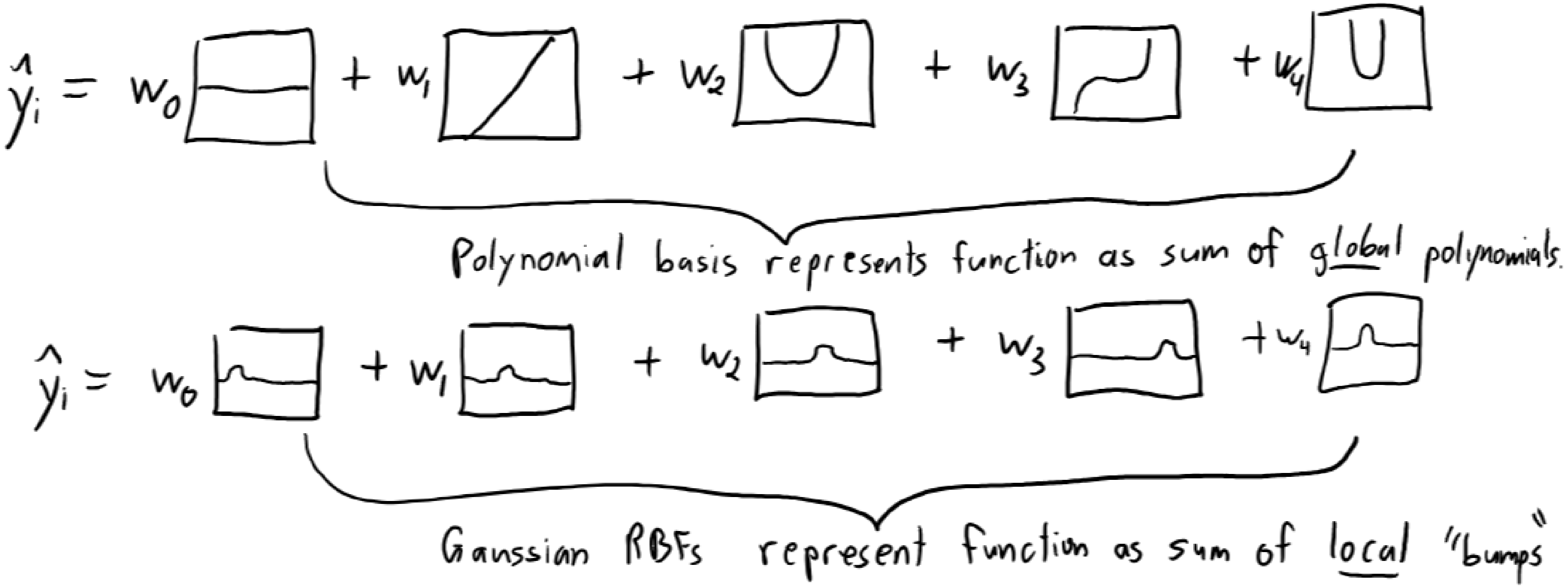
- A model that is hard to beat:
  - RBF basis with L2-regularization and cross-validation to choose σ and λ.
  - Flexible non-parametric basis, magic of regularization, and tuning for test error.

- Pseudo code:

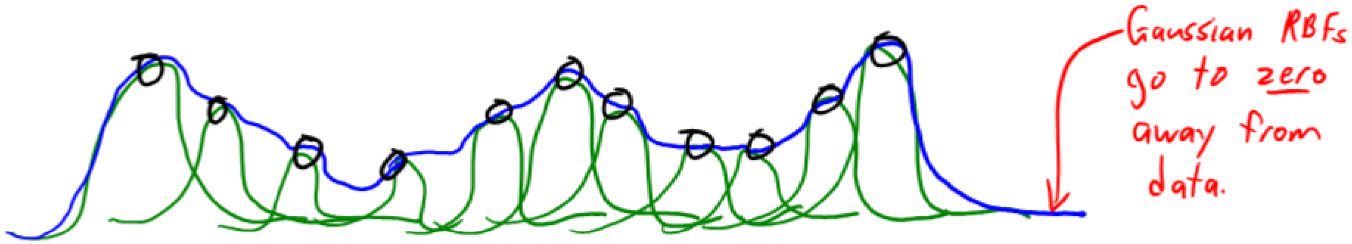
```
for each value of λ and θ:
    Z = get_z_matrix(X, λ, θ)
    w = (Z^T Z + λI)^(-1) Z^T y
    Z_val = get_z_matrix(X_val, λ, θ)
    y_vpred = Z_val*w
    val_err = np.norm(y_vpred, y_val)
    update λ, θ with the current best val_err
```

RBF: related to linear regression

Gaussian RBFs: A Sum of “Bumps”



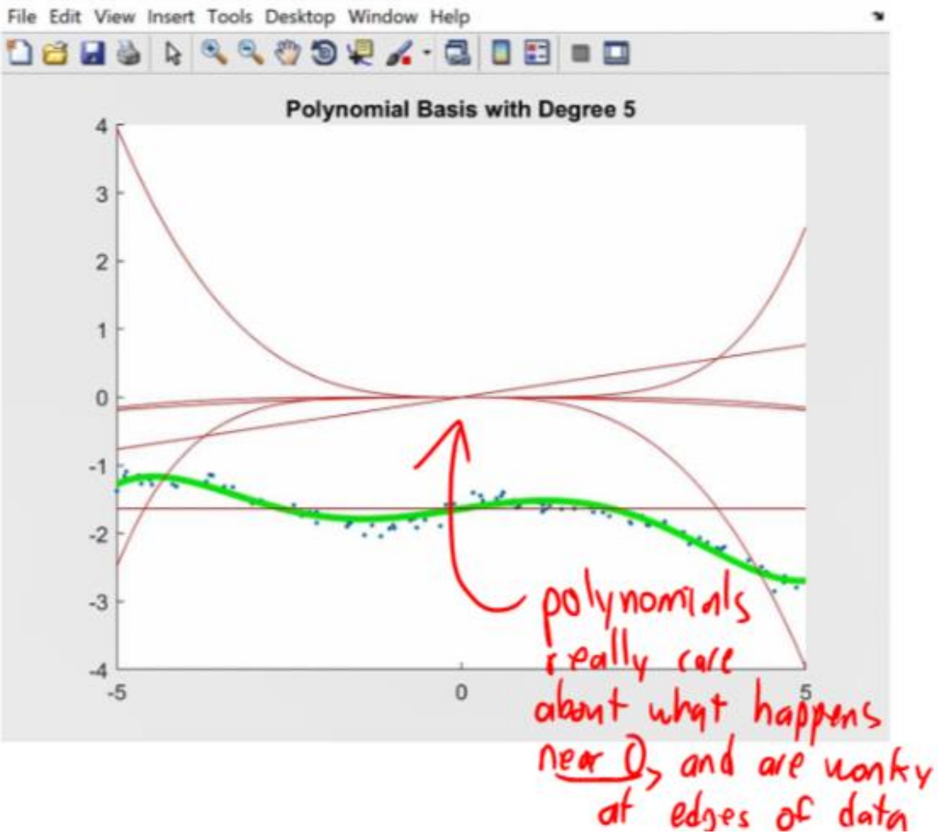
- Constructing a function from bumps (“smooth histogram”):

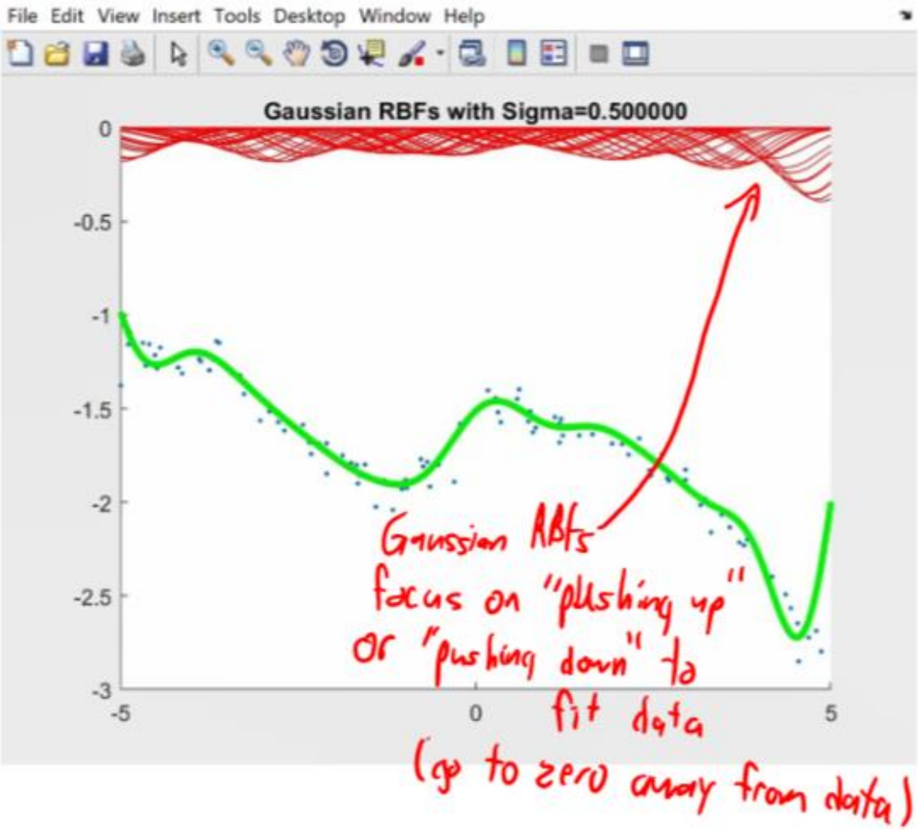


RBF: related to linear regression

Gaussian RBFs: A Sum of "Bumps"

- Red is weight*feature, green is prediction (sum of red lines):

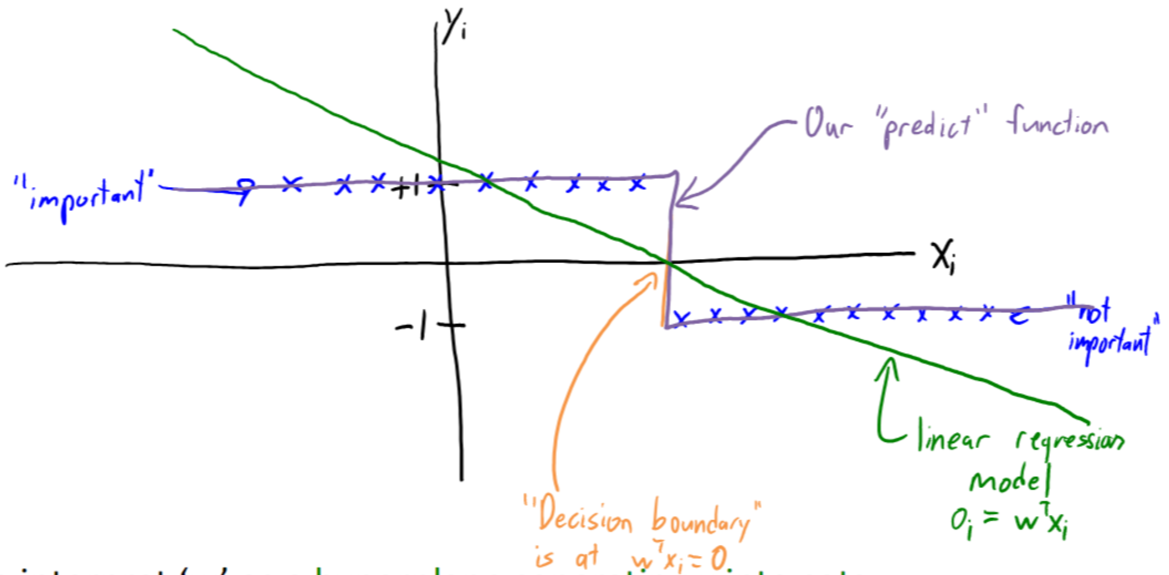




- Regularization
- RBF
- Linear Classifiers

Linear Classification: what is decision boundary

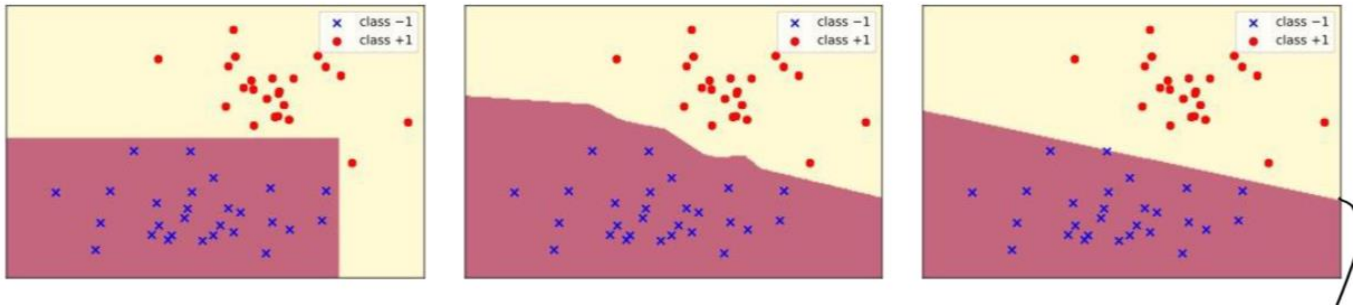
- Decision Boundary in 1D



- We can interpret 'w' as a hyperplane separating x into sets:
  - Set where $w^T x_i > 0$ and set where $w^T x_i < 0$.

- Decision Boundary in 2D

decision tree | KNN | linear classifier



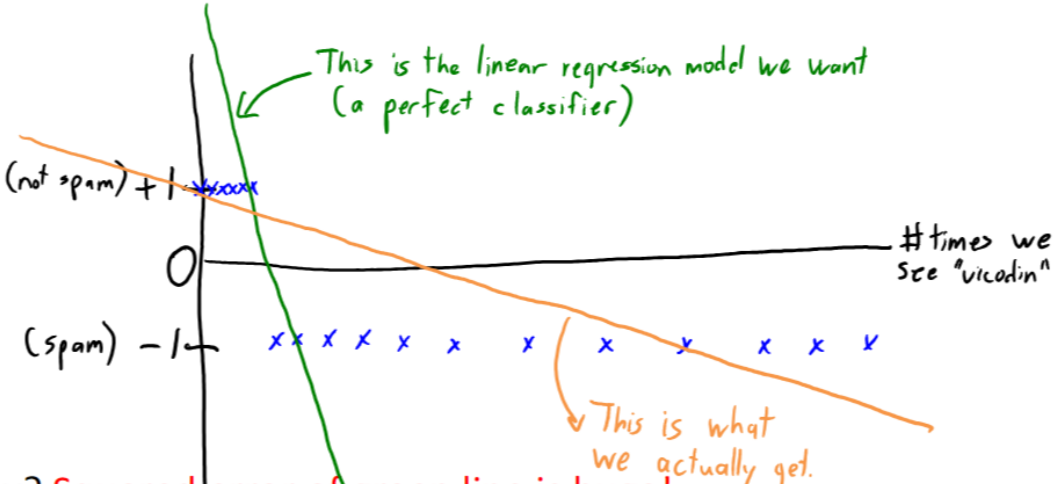
Linear Classification: why using regression directly is not good

- Consider training by minimizing squared error with y_i that are +1 or

$$f(w) = \frac{1}{2}\|Xw - y\|^2 \qquad y = \begin{bmatrix} +1 \\ +1 \\ -1 \\ +1 \\ -1 \end{bmatrix}$$

- If we predict $o_i = +0.9$ and $y_i = +1$, error is small: $(0.9 – 1)^2 = 0.01$.
- If we predict $o_i = -0.8$ and $y_i = +1$, error is bigger: $(-0.8 – 1)^2 = 3.24$.
- If we predict $o_i = +100$ and $y_i = +1$, error is huge: $(100 – 1)^2 = 9801$.
  – But it shouldn't be, the prediction is correct.

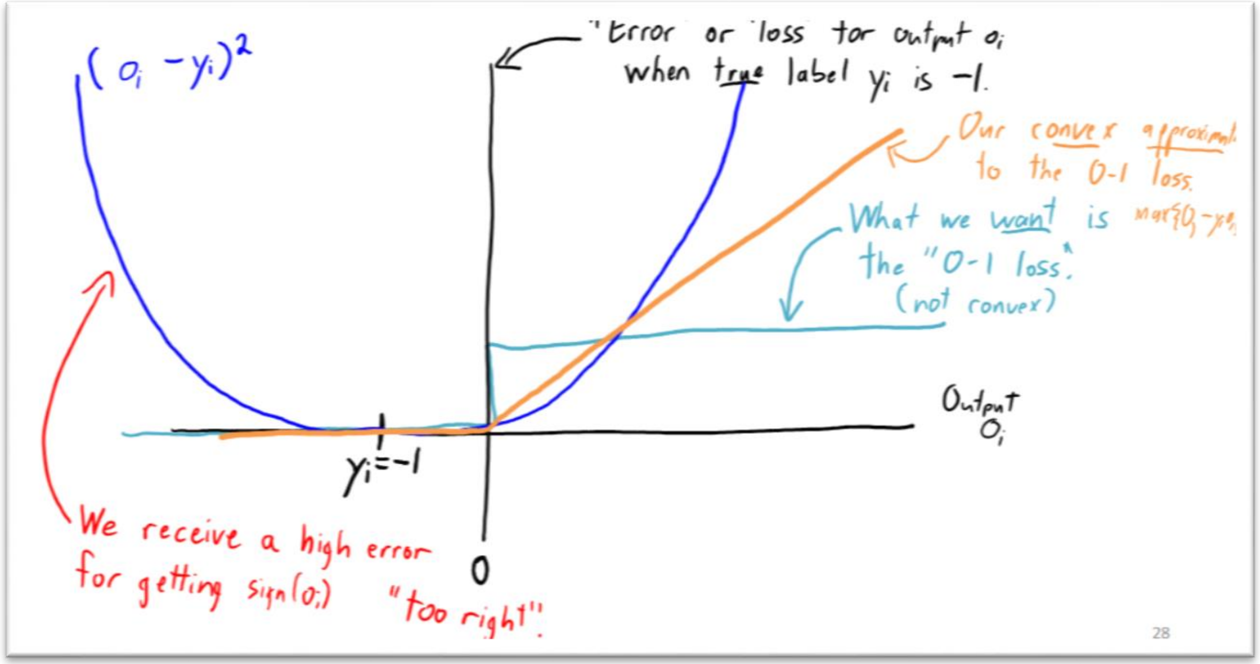
- Least squares can behave weirdly when applied to classification:

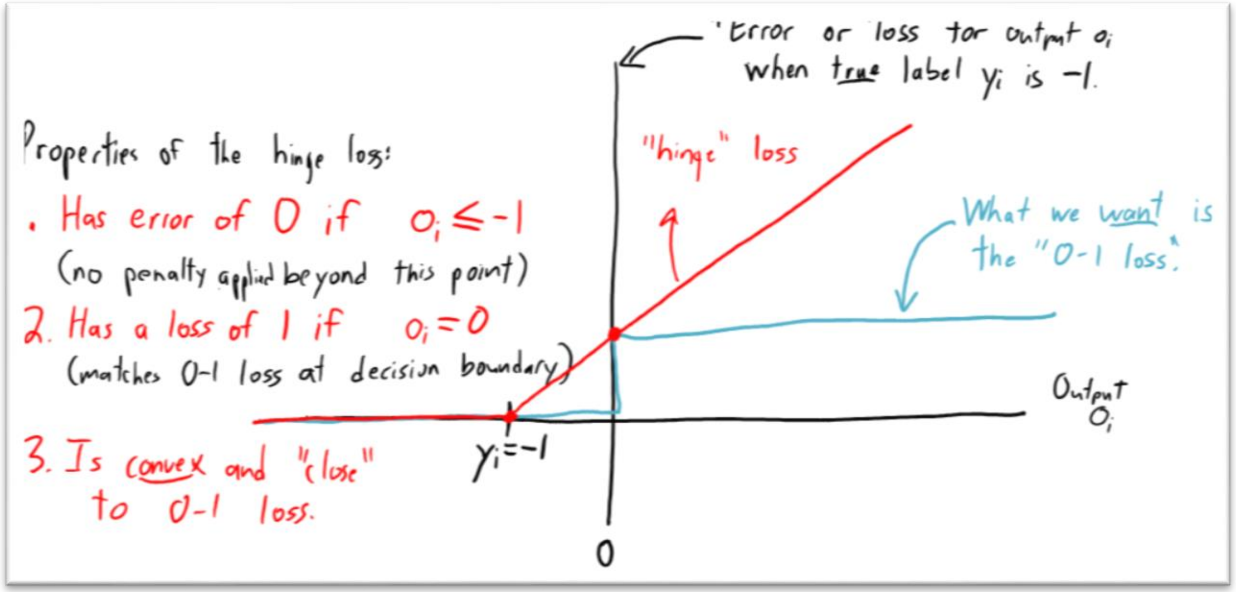


- Why? Squared error of green line is huge!
  – Make sure you understand why the green line achieves an error of 0.

Linear Classification: if not L2 loss, what should we use?

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





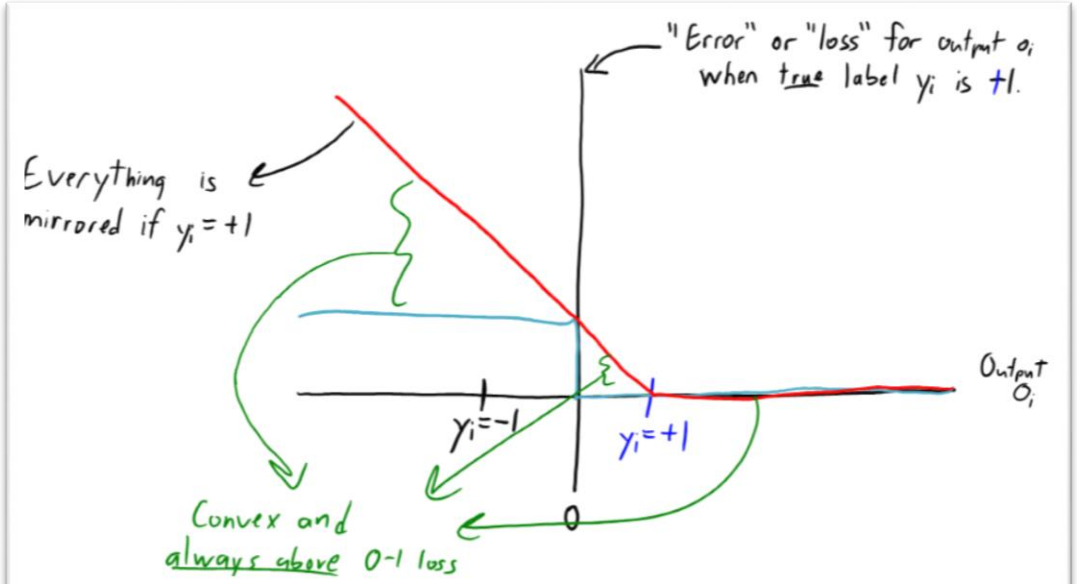
We could train by minimizing sum over all examples:

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

But this has a degenerate solution:

– We have f(0) = 0, and this is the lowest possible value of ‘f’.

There are two standard fixes: hinge loss and logistic loss.



Thanks for your time!
Questions?