Logistic Regression

Pontus Giselsson

Learning goals

- Understand logistic regression and its purpose – classification
- Understand that the training problem is convex
- Understand the problem of overparameterization and overfitting
- Understand the purpose and need for regularization
- Familiar with the effect of some convex regularization choices
- Understand the use and purpose of feature maps
- Understand hyperparameters and how they can be chosen

Supervised learning

- Let (x, y) represent object and label pairs
  - Object $x \in \mathcal{X} \subseteq \mathbb{R}^n$
  - Label $y \in \mathcal{Y} \subseteq \mathbb{R}^K$
- Available: Labeled training data (training set) $\{(x_i, y_i)\}_{i=1}^N$
  - Data $x_i \in \mathbb{R}^n$ are called *examples* (often n large)
  - Labels $y_i \in \mathbb{R}^K$ are called *response variables* (often $K = 1$)

Objective:

- Find data to label transformation $\psi : \mathcal{X} \to \mathcal{Y}$ such that

$$\psi(x) \approx y$$

  for all data label pairs (x, y)
- Learn ψ from training data, but should *generalize* to all (x, y)

Notation

- (Primal) Optimization variable notation:
  - Optimization literature: x, y, z (as in first part of course)
  - Statistics literature: β
  - Machine learning literature: θ, w, b
- Reason: data, labels in statistics and machine learning are x, y
- Will use machine learning notation in these lectures
- We collect training data in matrices (one example per row)

$$X = \begin{bmatrix} x_1^T \\ \vdots \\ x_N^T \end{bmatrix} \qquad Y = \begin{bmatrix} y_1^T \\ \vdots \\ y_N^T \end{bmatrix}$$

- Columns X_j of data matrix $X = [X_1, \ldots, X_n]$ are called *features*

Regression vs Classification

There are two main types of supervised learning tasks:

- Regression:
  - Predicts quantities
  - Example: Least squares
- Classification:
  - Predicts class beloning
  - Example: Logistic regression (dispite its name)

Classification

- $\mathcal{Y}$ has finite cardinality (finite number of outcomes, e.g., $\{0, 1\}$)
- The data to label transformation ψ is a composition $\psi = \sigma \circ m$
  - $m : \mathbb{R}^n \to \mathbb{R}^K$ is parameter dependent data regression model
  - $\sigma : \mathbb{R}^K \to \mathcal{Y}$ is a *fixed* function that:
    - maps regression model output to response range $\mathcal{Y}$ (or $\operatorname{conv}\mathcal{Y}$)
    - is typically the (sub)gradient of a convex function (monotone)
- Classification problems try to find model parameters θ such that

$$\sigma(m(x;\theta)) \approx y \qquad \Longleftrightarrow \qquad \sigma(m(x;\theta)) - y \approx 0$$

  for all data and label pairs (x, y) by solving training problem

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} L(m(x_i;\theta), y_i)$$

- Learn model from training data, but should *generalize* to all (x, y)

Classification loss function

- Many different classification loss functions exist
- This lecture: specific construction that gives logistic regression

First – Does least squares loss work?

- Objective: Find model paramters θ such that for all (x, y):

$$\sigma(m(x;\theta)) - y \approx 0$$

- Let model output $u = m(x;\theta)$; Least squares loss:

$$L(u, v) = \|\sigma(u) - y\|_2^2$$

  typically *not* convex in u due to nonlinear σ
- This works for regression ($\sigma = I$) but *not* used for classification!

A classification loss function

- Objective: Find model parameters θ such that for all (x, y):

$$\sigma(m(x;\theta)) - y \approx 0$$

- Require σ to be the (sub-)gradient of a convex function
- Convex function is $\int \sigma(u)du$ where $\int$ denotes primitive function[1]
- Let model output $u = m(x;\theta)$; and define loss:

$$L(u, y) = \int \sigma(u)du - y^T u + C,$$

  where C is arbitrary constant
- Loss is convex in u since integral and $-y^T u$ are convex

---

[1] Primitive function should really be $(\int \sigma(v)dv)(u)$

Loss function mimimum

- Loss function $L(u, y) = \int \sigma(u)du - y^T u + C$
- Subdifferential[1] (by definition) $\partial L(u, y) = \sigma(u) - y$
- Fermat's rule says loss function minimized if and only if

$$0 \in \sigma(u) - y$$

  and if σ single-valued, i.e., the cost is differentiable:

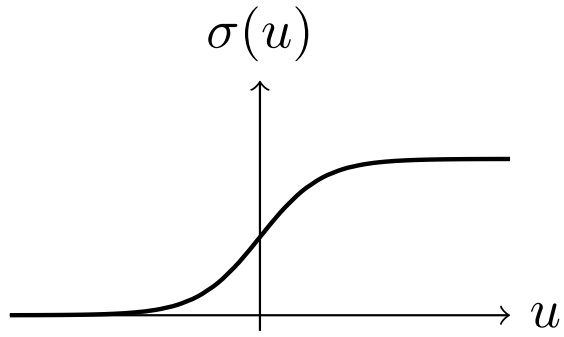
$$0 = \sigma(u) - y$$

- Loss minimized if model output $u = m(x; \theta)$ equals response y
- How to choose σ?

[1] We use notation $\partial L(u, y)$ for $\partial L(\cdot, y)(u)$, i.e., w.r.t. first argument

Logistic regression – σ-function

- Response $y \in \{0, 1\}$, approximation $\sigma(m(x)) \approx y$
- Uses *logistic* function $\sigma(u) = \frac{1}{1+e^{-u}}$ (also called *sigmoid*):



- Function is *maximal monotone* and gradient of convex function
- Range is $(0, 1)$, i.e., interior of probability interval $[0, 1]$

Logistic regression – Loss function

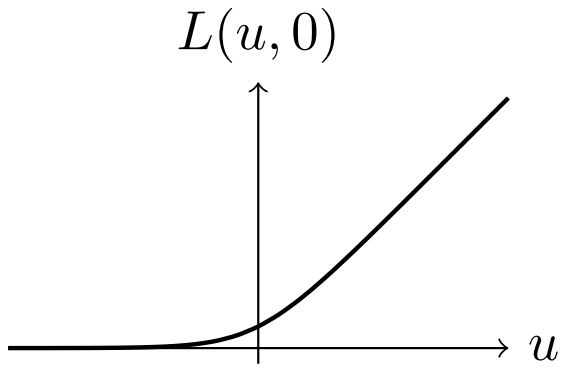
- Primitive function of σ:

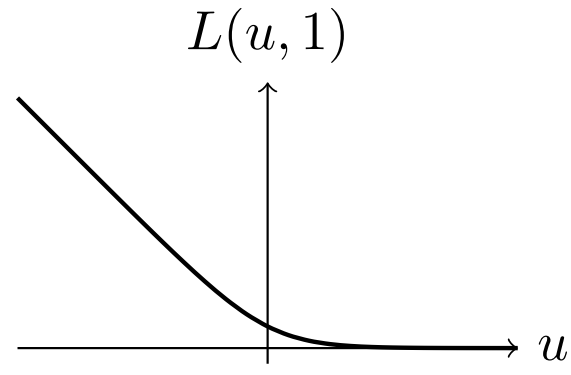
$$\int \sigma(u)du = \int \frac{1}{1+e^{-u}}du = \log(1+e^{u}) + C$$

- Let $C = 0$ to get

$$L(u, y) = \int \sigma(u)du - yu + C = \log(1+e^{u}) - yu$$

- Loss function for $y = 0$ and $y = 1$:





Alternative labels

- Multiply responses by 2 and subtract 1 to get $y \in \{-1, 1\}$
- Multiply logistic function by 2 and subtract 1 to get new

$$\sigma(u) := \frac{2}{1+e^{-u}} - 1 = \frac{1-e^{-u}}{1+e^{-u}} = \frac{e^{u/2} - e^{-u/2}}{e^{u/2} + e^{-u/2}} = \tanh(u/2)$$

- Loss function

$$L(u, y) = \int \sigma(u) du - yu + C = 2\log(1 + e^u) - u - yu$$

  equivalent (modulo scaling by 2) to before for $y = -1$ and $y = 1$
- If y only -1 or 1, loss function (scaled by $1/2$) can be written as

$$L(u, y) = \log(1 + e^{-yu})$$

  (this formula is not equivalent to the above for other values of y!)

Logistic regression – Model and training problem

- Logistic regression uses affine model $m(x;\theta) = w^T x + b$
- Training problem:

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} L(m(x_i;\theta), y_i) = \sum_{i=1}^{N} \left(\log(1 + e^{x_i^T w + b}) - y_i(x_i^T w + b) \right)$$

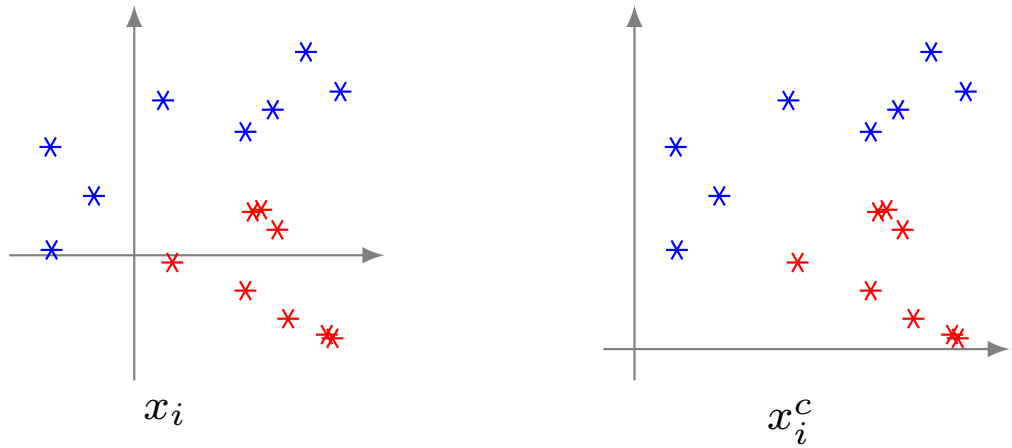
- Training problem convex in $\theta = (w, b)$ since model affine
- $y = 0$: low cost for $m(x;\theta) \ll 0$, $y = 1$: low cost for $m(x;\theta) \gg 0$

The bias term

- The model $m(x;\theta) = w^T x + b$ bias term is b
- Least squares: optimal b has simple formula
- No simple formula to remove bias term here!

Bias term gives shift invariance

- Assume all data points shifted $x_i^c := x_i - c$
- We want same hyperplane to separate data, but shifted



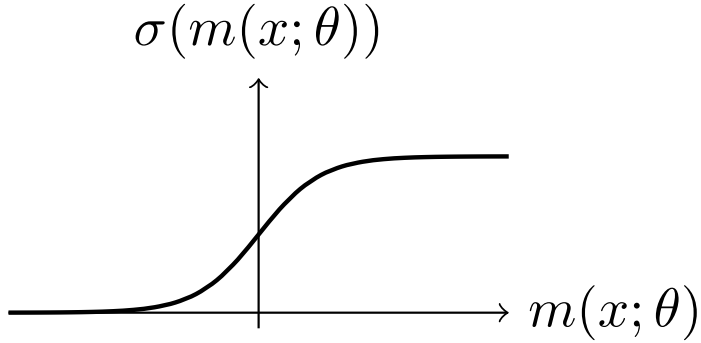
- Assume $(w, b) = (\bar{w}, \bar{b})$ is optimal for (x_i, y_i)
- Then $(w, b) = (\bar{w}, \bar{b}_c)$ with $\bar{b}_c = \bar{b} + \bar{w}^T c$ optimal for (x_i^c, y_i)
- Why? Model outputs the same:
  - Model output $m(x_i; \theta) = w^T x_i + b$
  - Model output $m(x_i^c; \theta) = w^T x_i^c + b = w^T x_i + (b + w^T c)$
  - Exactly the same output by shifting bias term with $w^T c$

Prediction

- Assume we have trained model m and want to predict label using

$$\sigma(m(x;\theta)) \approx y$$

- If $m(x;\theta) \gg 0$, then label estimate $\sigma(m(x;\theta)) \approx 1$
- If $m(x;\theta) \ll 0$, then label estimate $\sigma(m(x;\theta)) \approx 0$
- Logistic function assigns probabilities for class beloning:
  - probability $\sigma(m(x;\theta))$ that x has label 1
  - probability $1 - \sigma(m(x;\theta))$ that x has label 0
- Predict label of x by thresholding $u = m(x;\theta)$ at 0 (prob. 0.5)

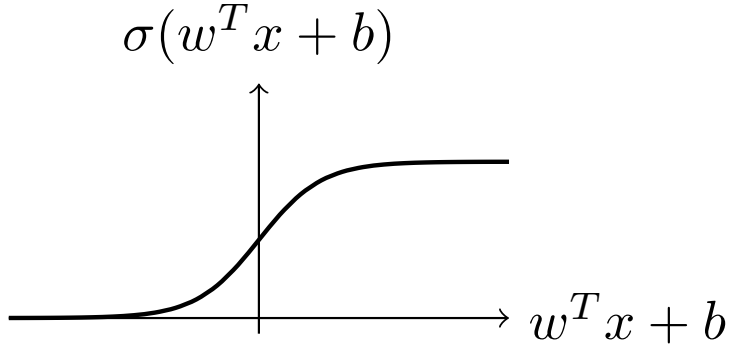


Prediction

- Since affine model $m(x;\theta) = w^T x + b$, prediction for x becomes:
  - If $w^T x + b < 0$, predict corresponding label $y = 0$
  - If $w^T x + b > 0$, predict corresponding label $y = 1$
  - If $w^T x + b = 0$, predict either $y = 0$ or $y = 1$ (equally probable)
- Therefore, the hyperplane (decision boundary)

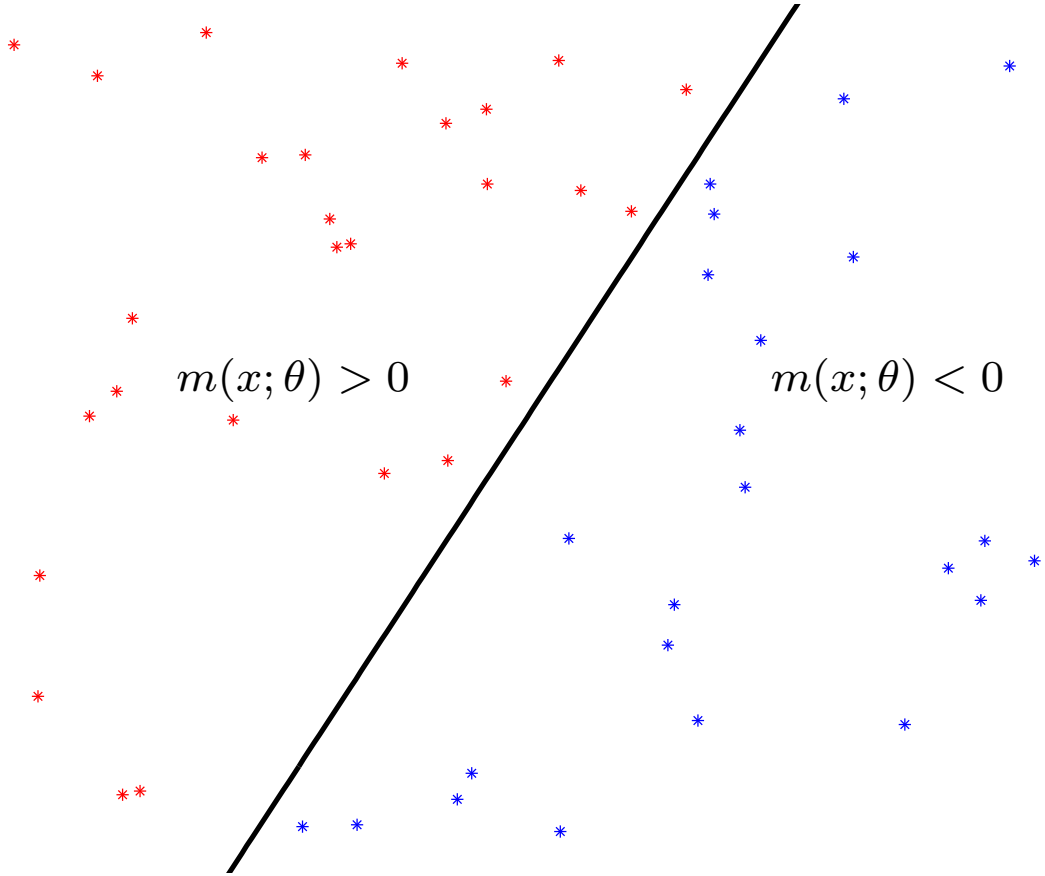
$$H := \{x : w^T x + b = 0\}$$

separates class predictions



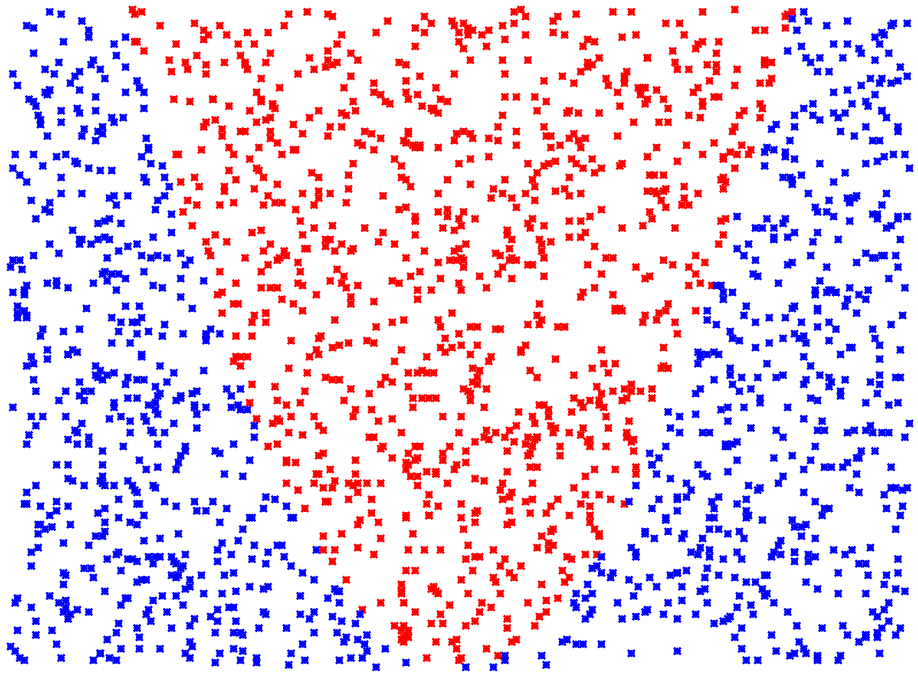
Example

- Fully separable training data
- Decision boundary decides label estimate for new data
- Optimal model parameters define normal to hyperplane



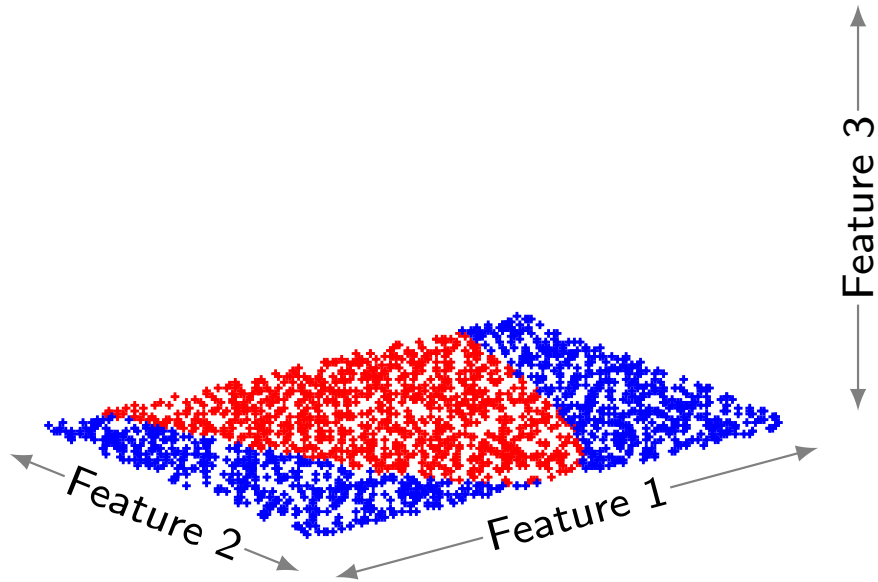
Logistic regression – Nonlinear example

- Logistic regression tries to affinely separate data
- Can nonlinear boundary be approximated by logistic regression?
- Introduce features (perform lifting)



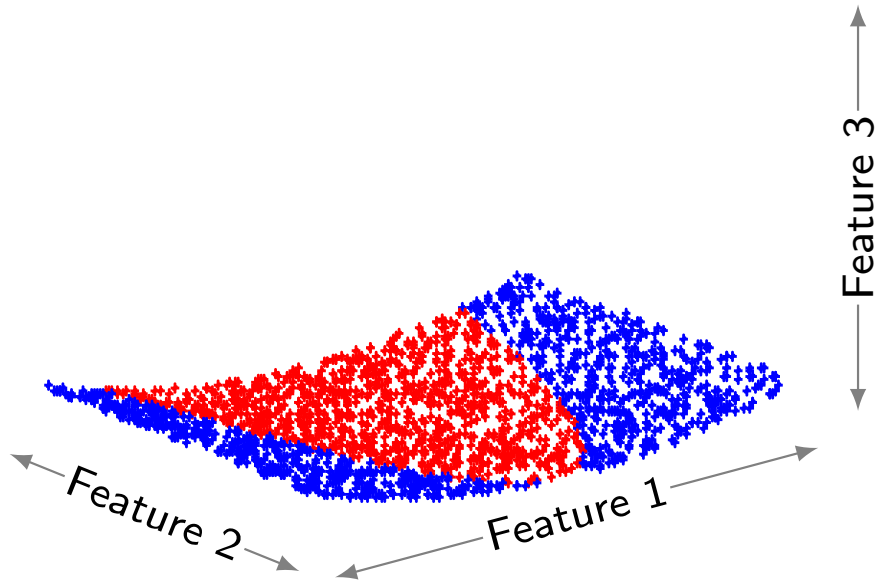
Logistic regression – Example

- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



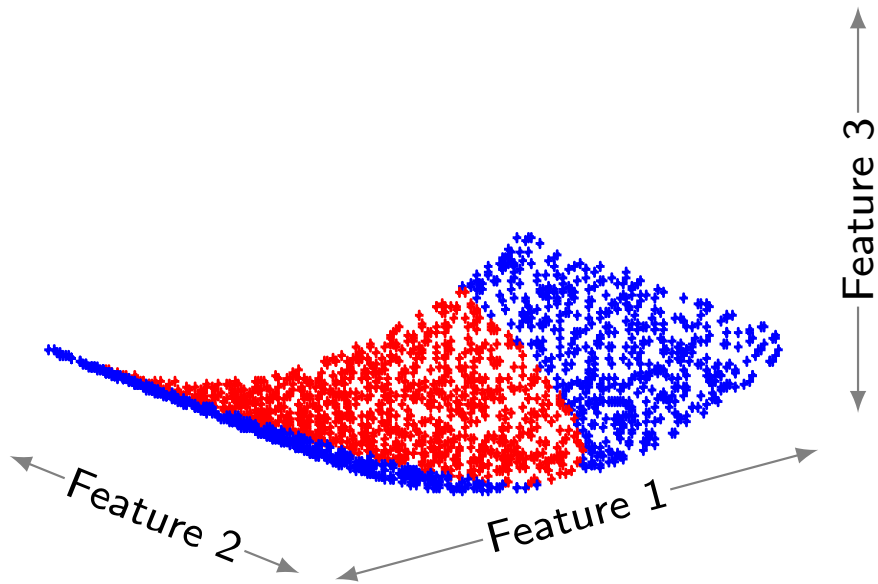
Logistic regression – Example

- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



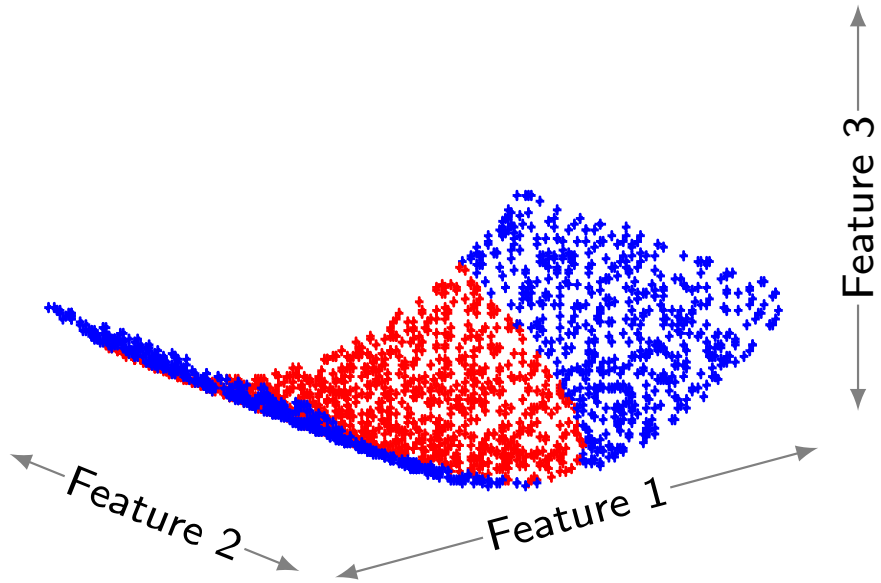
Logistic regression – Example

- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



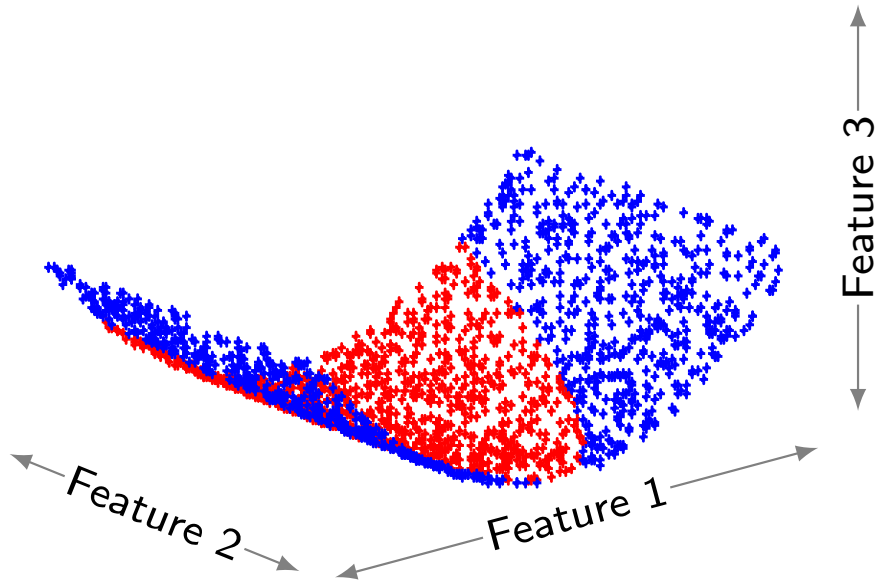
Logistic regression – Example

- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



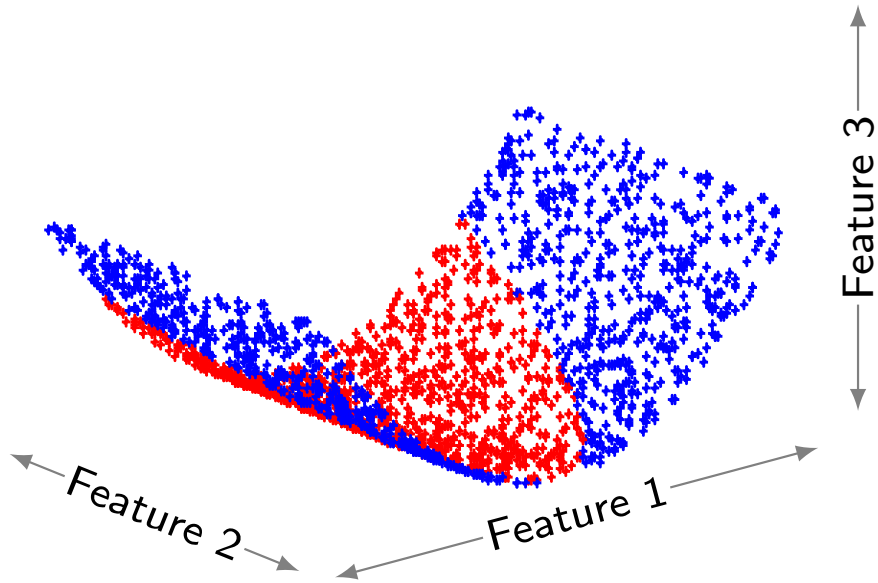
Logistic regression – Example

- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



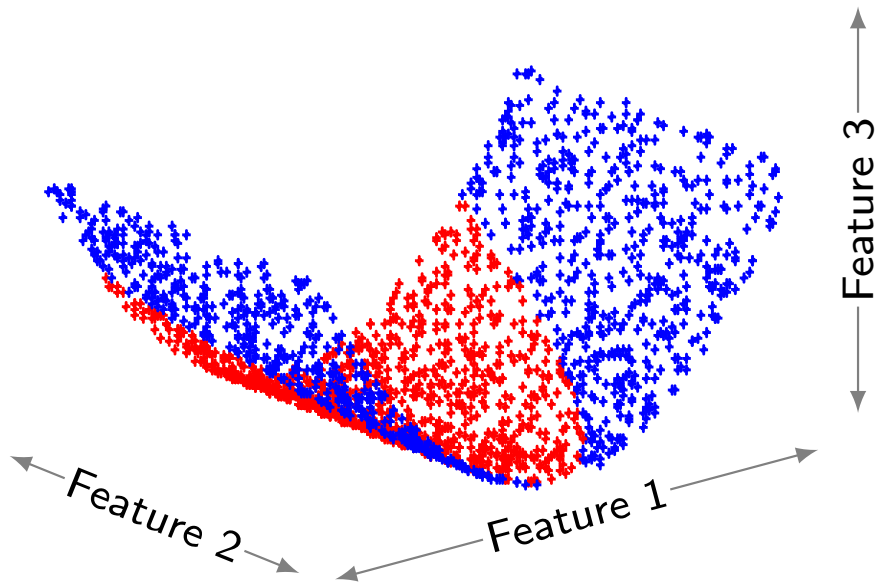
Logistic regression – Example

- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



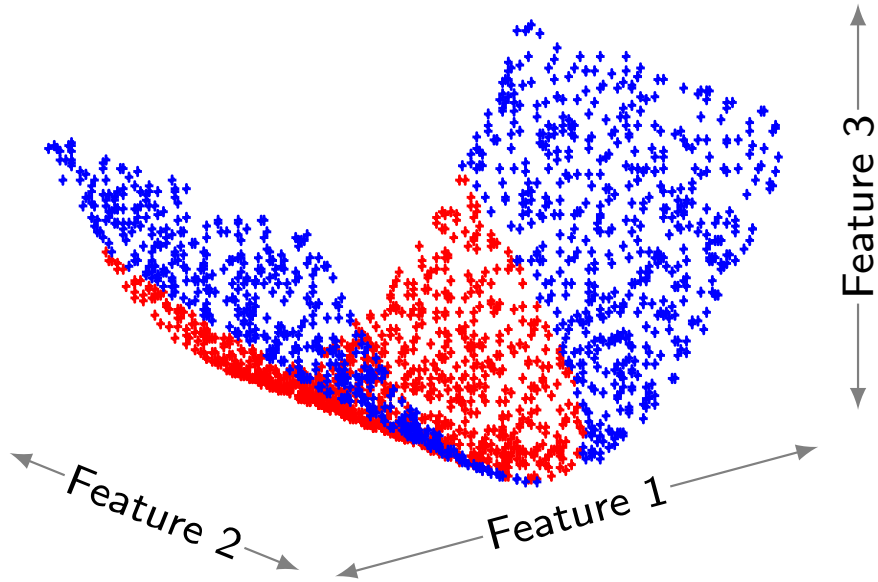
Logistic regression – Example

- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



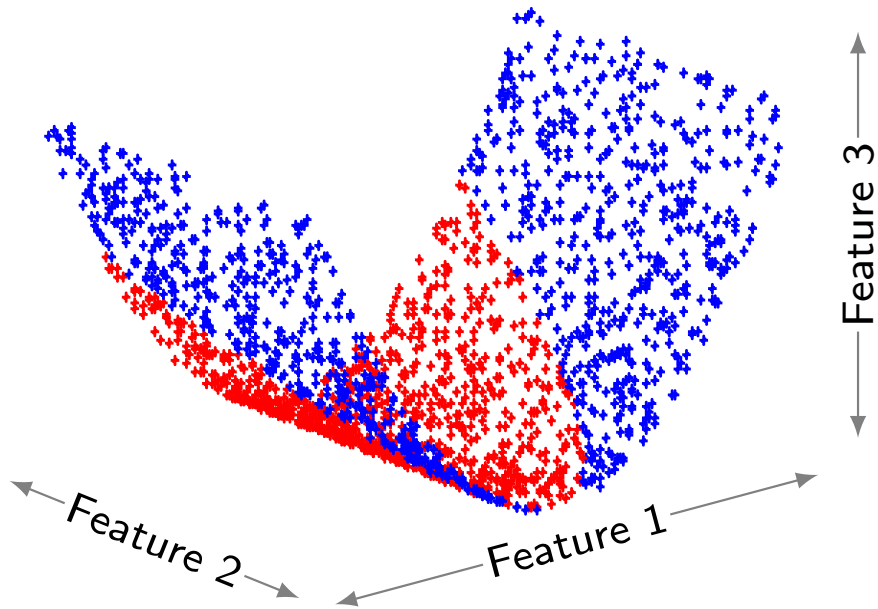
Logistic regression – Example

- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



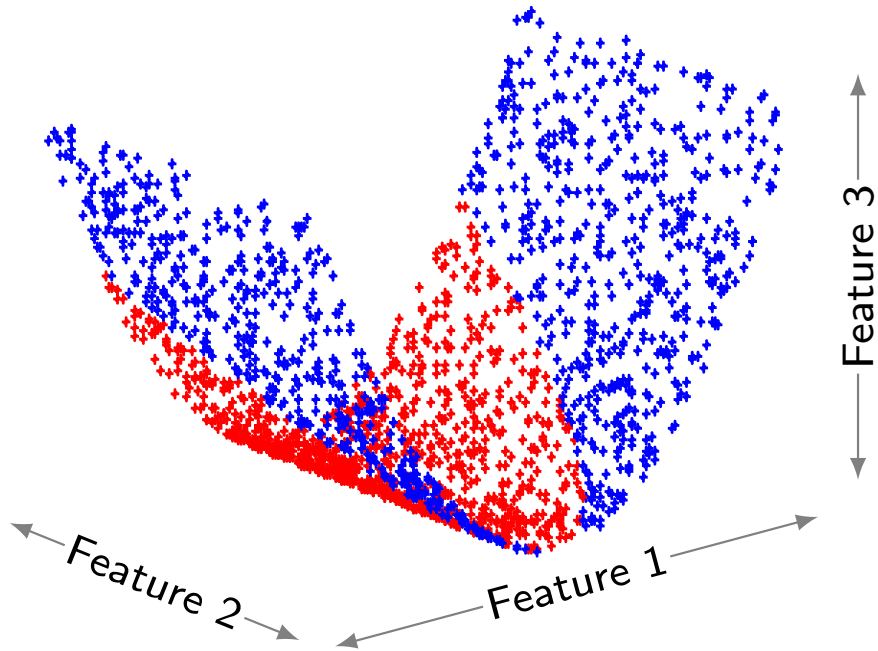
Logistic regression – Example

- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



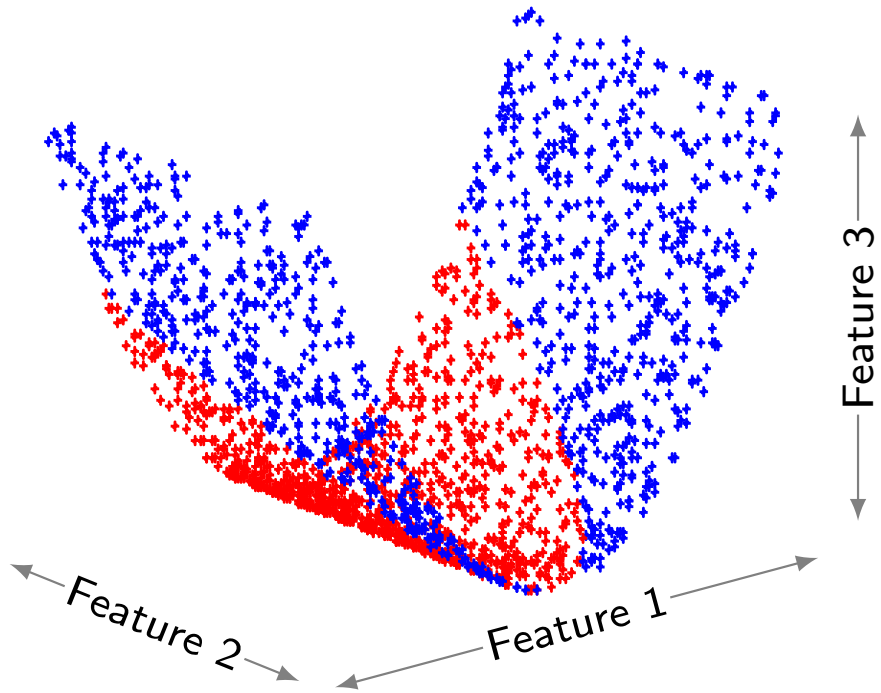
Logistic regression – Example

- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



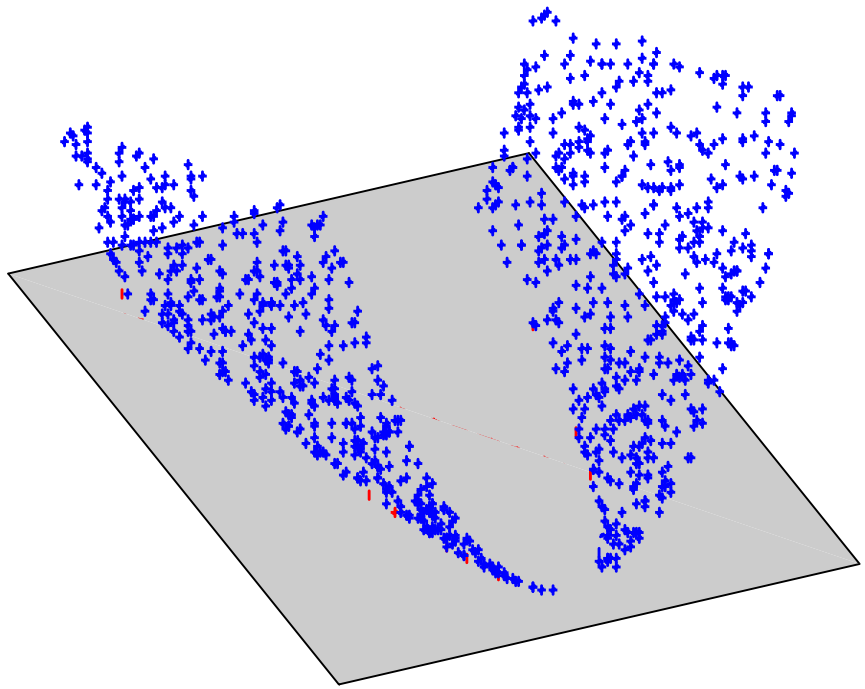
Logistic regression – Example

- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



Logistic regression – Example

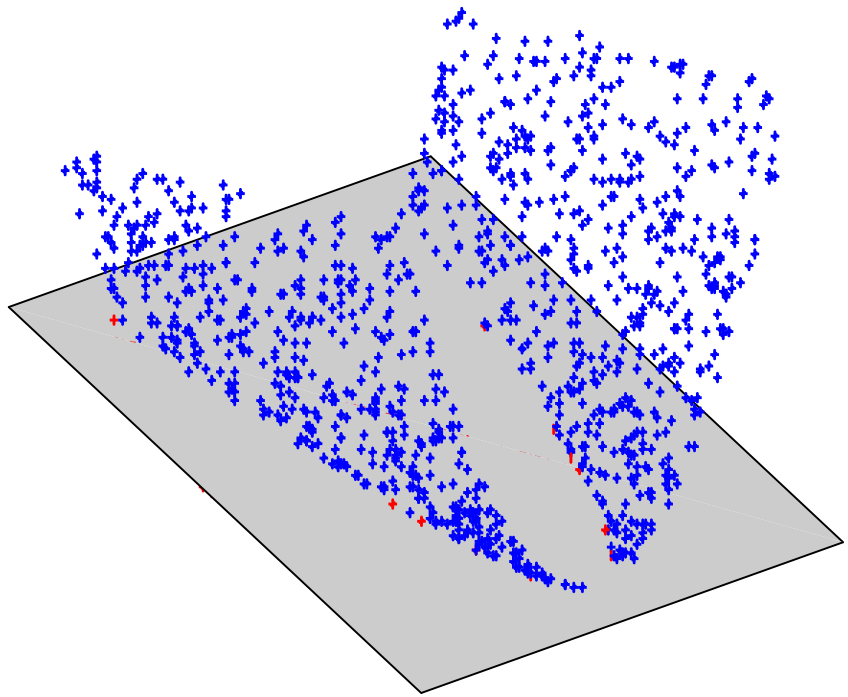
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

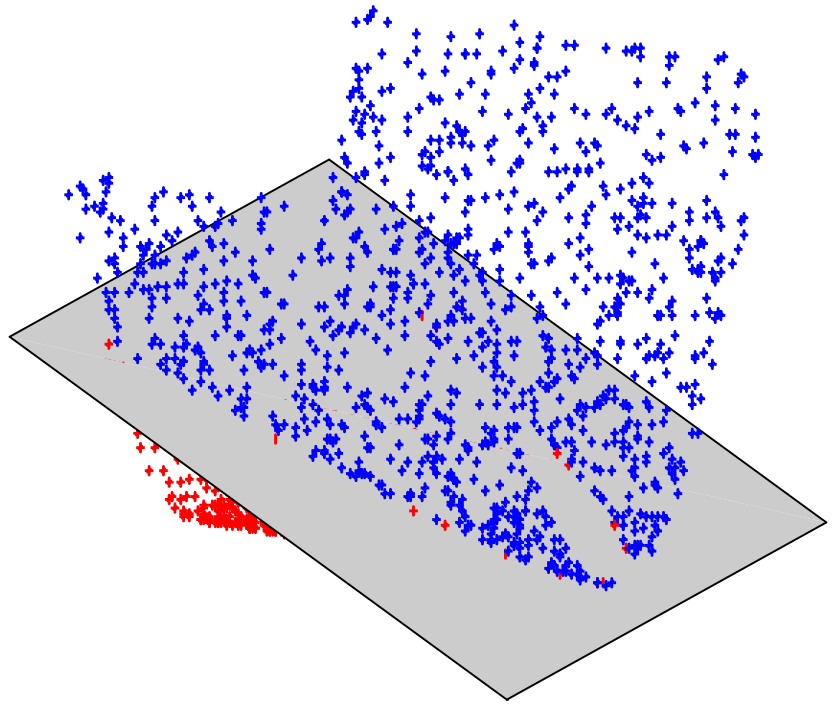
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

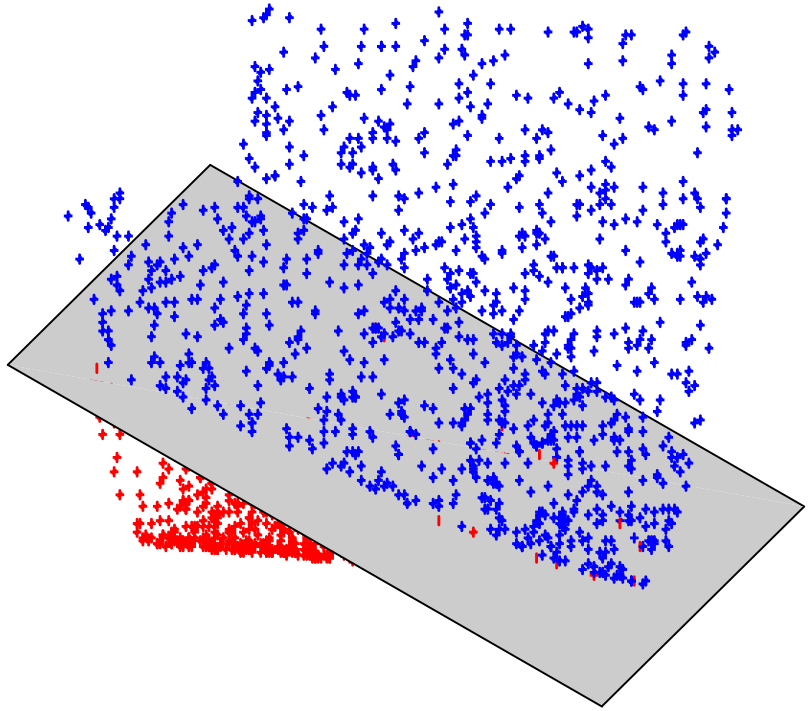
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

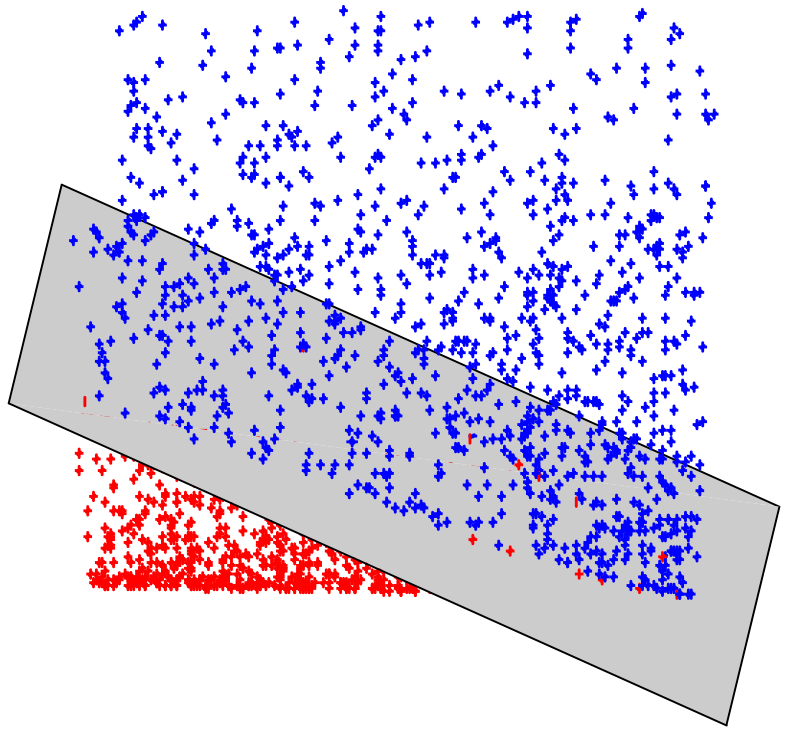
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

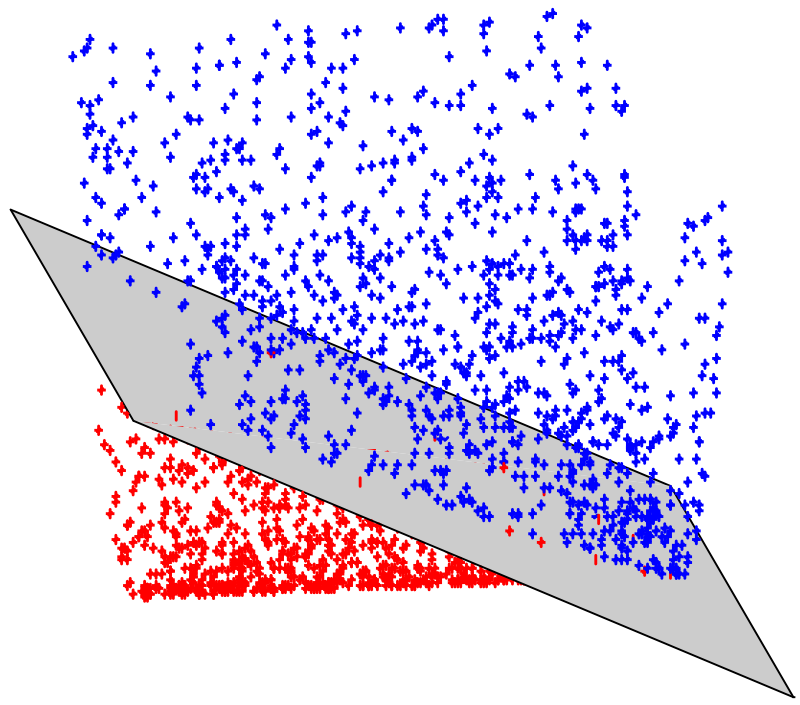
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

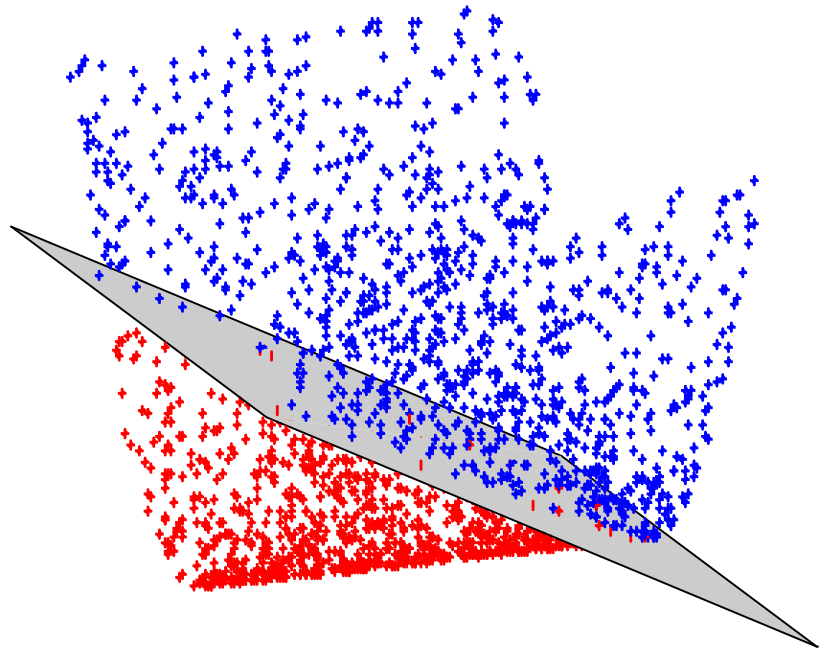
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

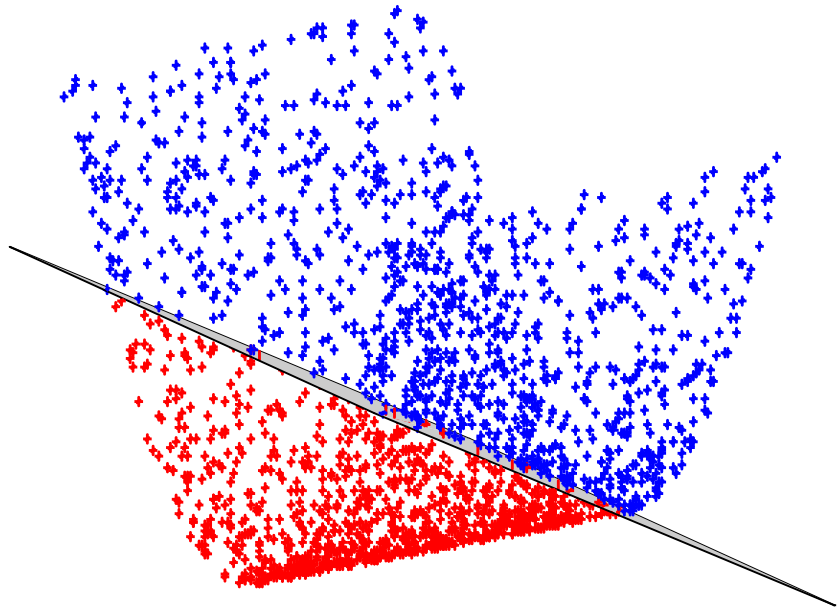
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

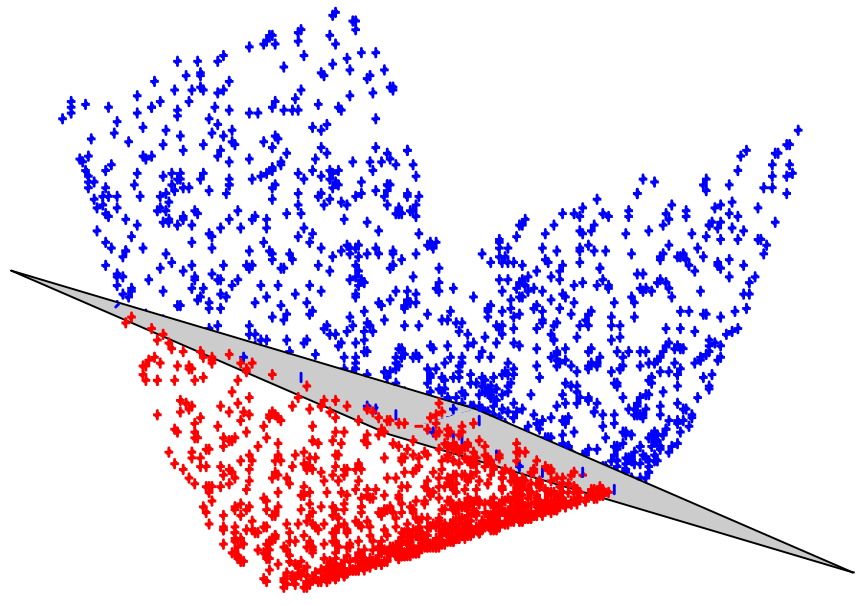
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

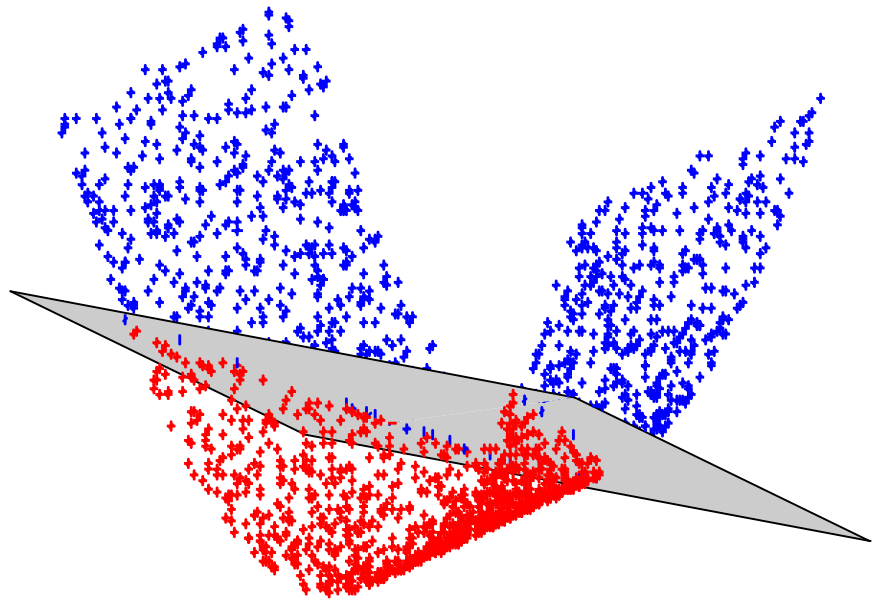
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

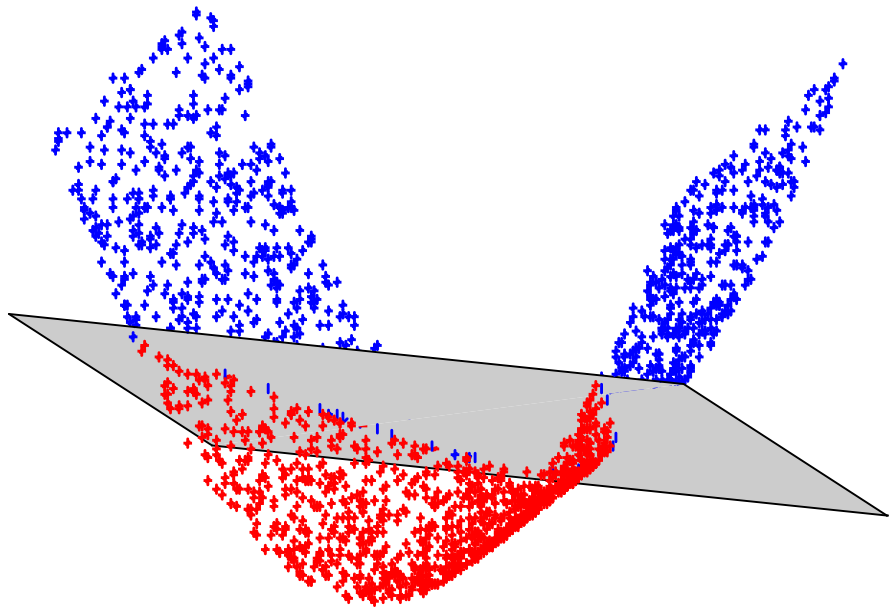
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Nonlinear models – Features

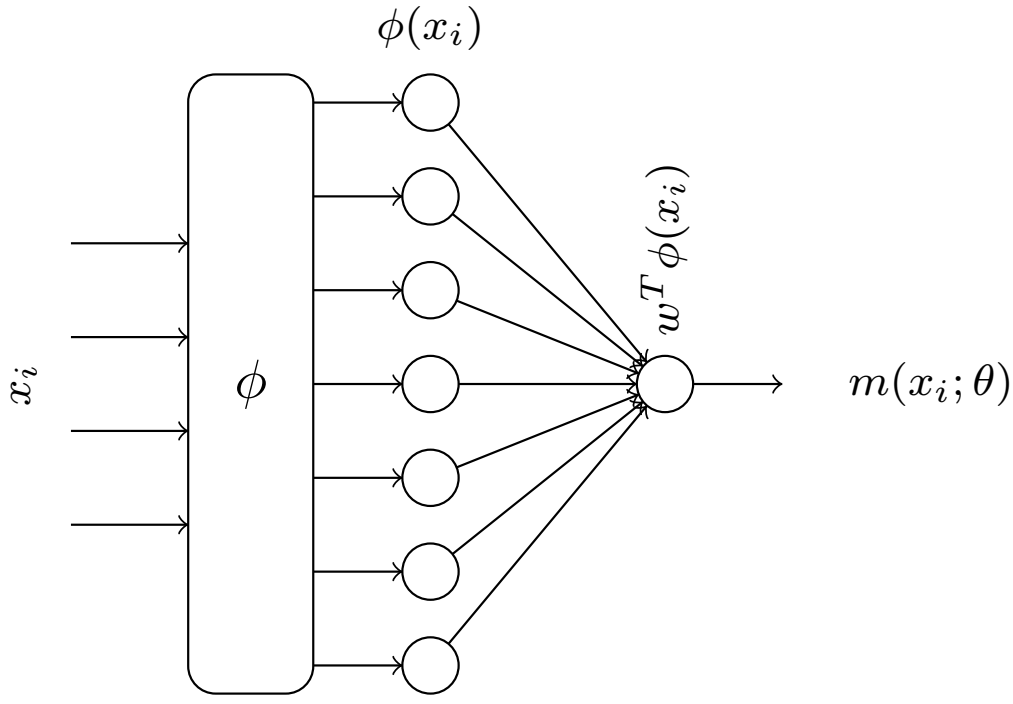
- Create feature map $\phi : \mathbb{R}^n \to \mathbb{R}^p$ of training data
- Data points $x_i \in \mathbb{R}^n$ replaced by featured data points $\phi(x_i) \in \mathbb{R}^p$
- New model: $m(x;\theta) = w^T\phi(x) + b$, still linear in parameters
- Feature can include original data x
- We can add feature 1 and remove bias term b
- Logistic regression training problem

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} \Big(\log(1 + e^{\phi(x_i)^T w + b}) - y_i(\phi(x_i)^T w + b) \Big)$$

  same as before, but with features as inputs

Graphical model representation

- A graphical view of model $m(x;\theta) = w^T\phi(x)$:



- The input x_i is transformed by *fixed* nonlinear features ϕ
- Feature-transformed input is multiplied by model parameters θ
- Model output is then fed into cost $L(m(x_i;\theta), y)$
- Problem convex since L convex and model affine in θ

Polynomial features

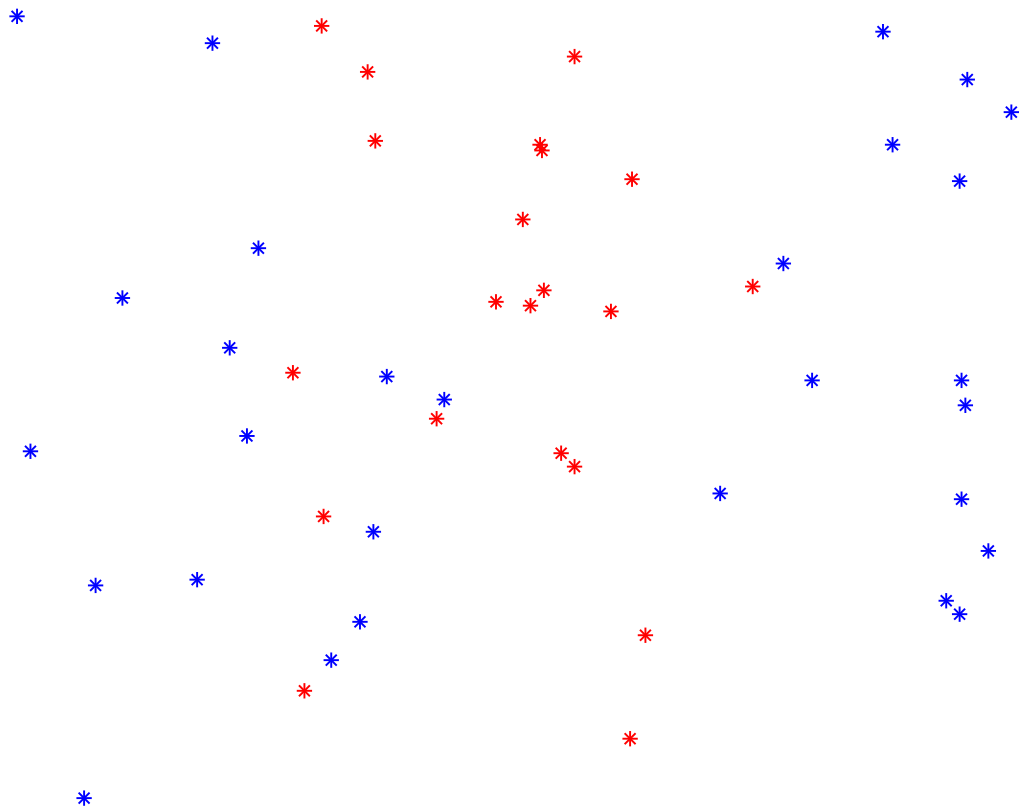
- Polynomial feature map for $\mathbb{R}^n$ with $n = 2$ and degree $d = 3$

$$\phi(x) = (x_1, x_2, x_1^2, x_1 x_2, x_2^2, x_1^3, x_1^2 x_2, x_1 x_2^2, x_2^3)$$

  (note that original data is also there)
- New model: $m(x; \theta) = w^T \phi(x) + b$, still linear in parameters
- Number of features $p + 1 = \binom{n+d}{d} = \frac{(n+d)!}{d!n!}$ grows fast!
- Training problem has $p + 1$ instead of $n + 1$ decision variables

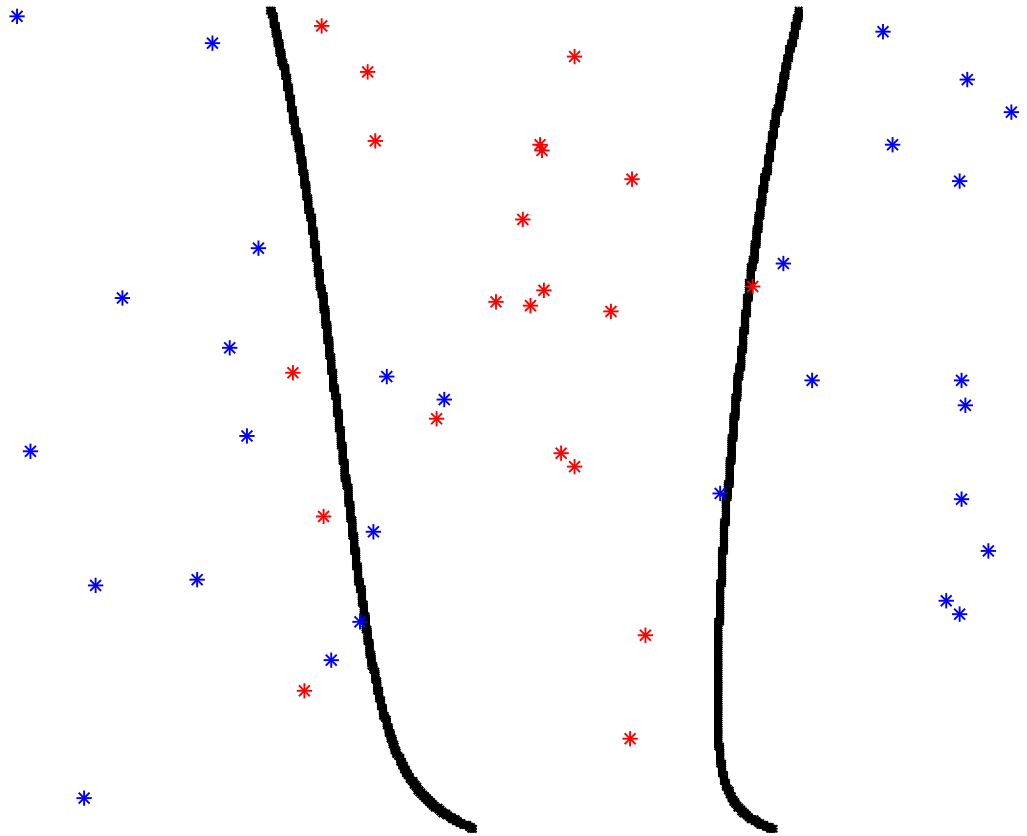
Example – Different polynomial model orders

- "Lifting" example with fewer samples and some mislabels
- Logistic regression (no regularization) polynomial features of degree:



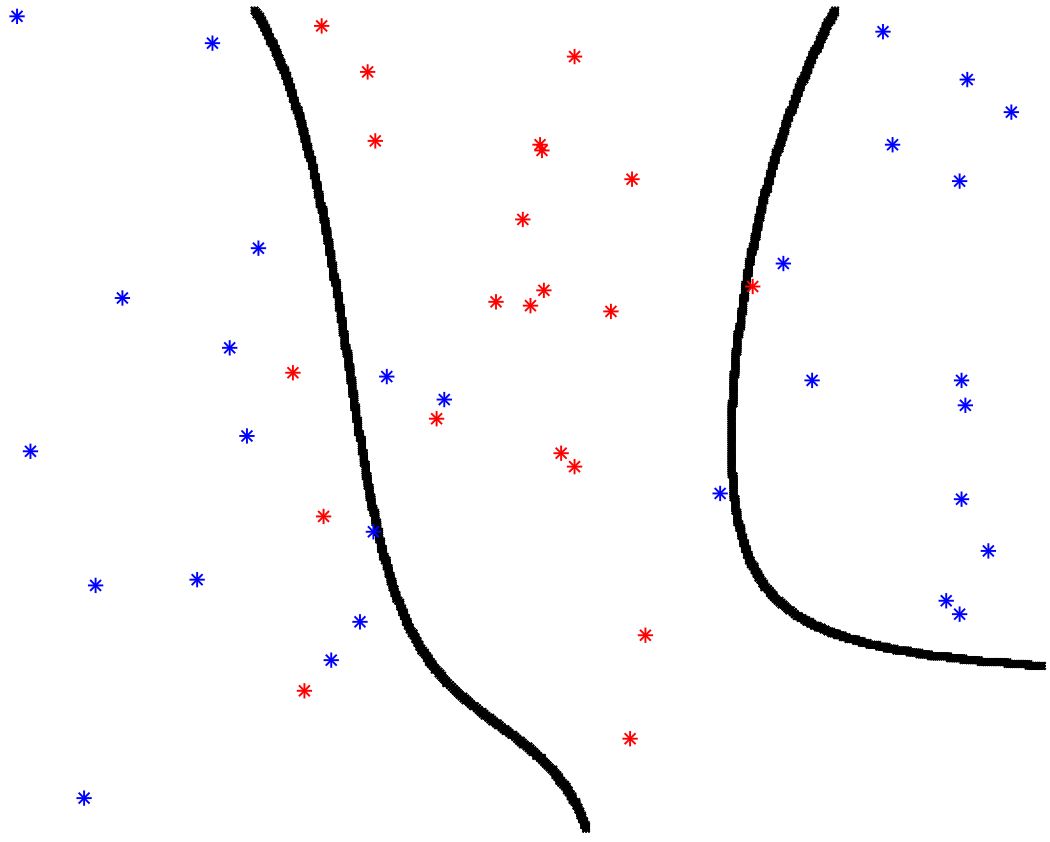
Example – Different polynomial model orders

- "Lifting" example with fewer samples and some mislabels
- Logistic regression (no regularization) polynomial features of degree: 2



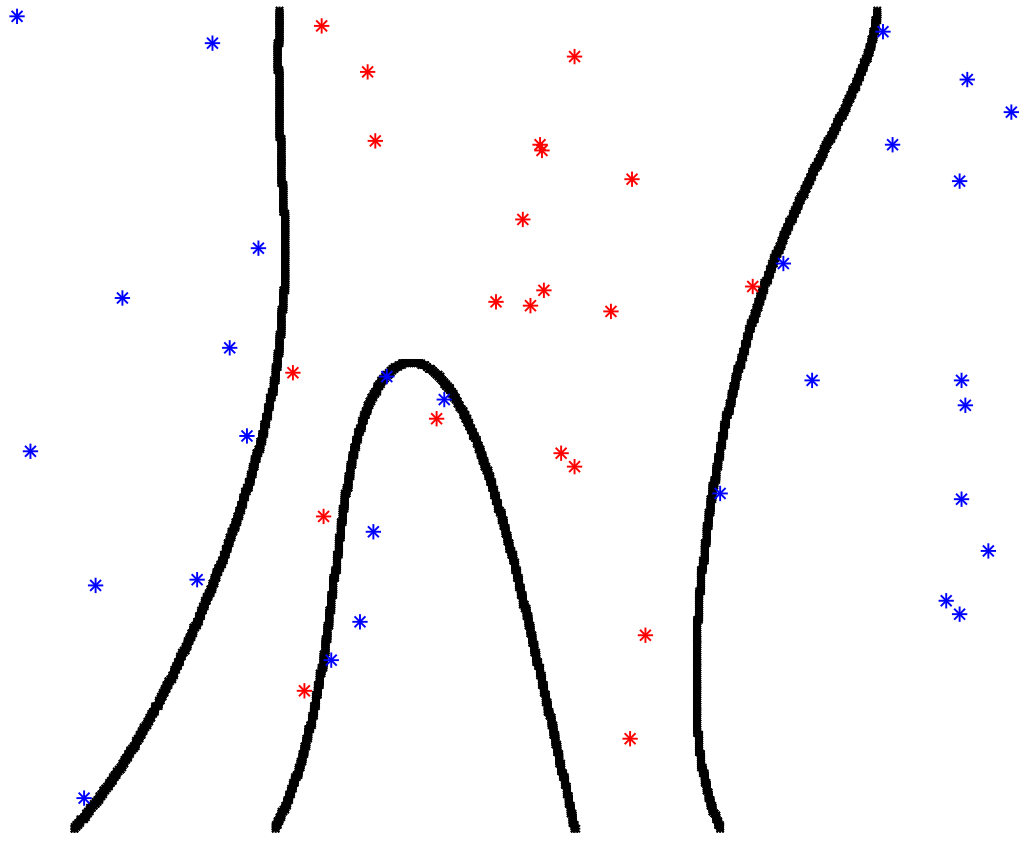
Example – Different polynomial model orders

- "Lifting" example with fewer samples and some mislabels
- Logistic regression (no regularization) polynomial features of degree: 3



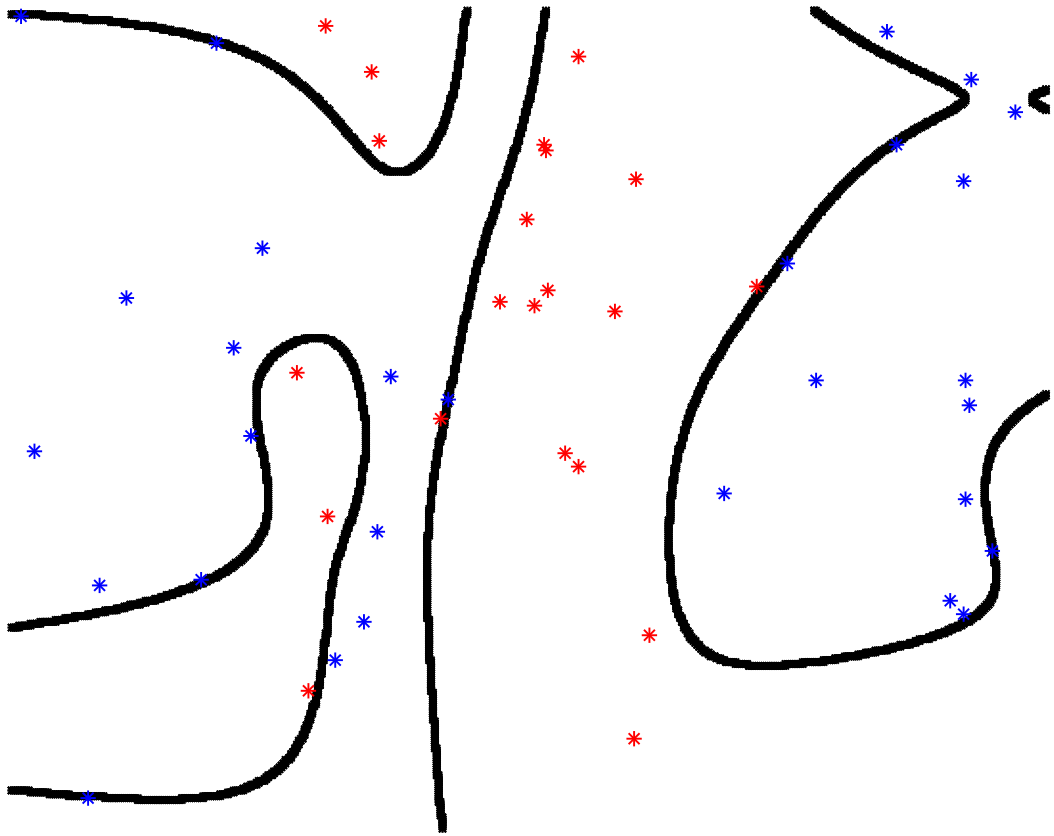
Example – Different polynomial model orders

- "Lifting" example with fewer samples and some mislabels
- Logistic regression (no regularization) polynomial features of degree: 4



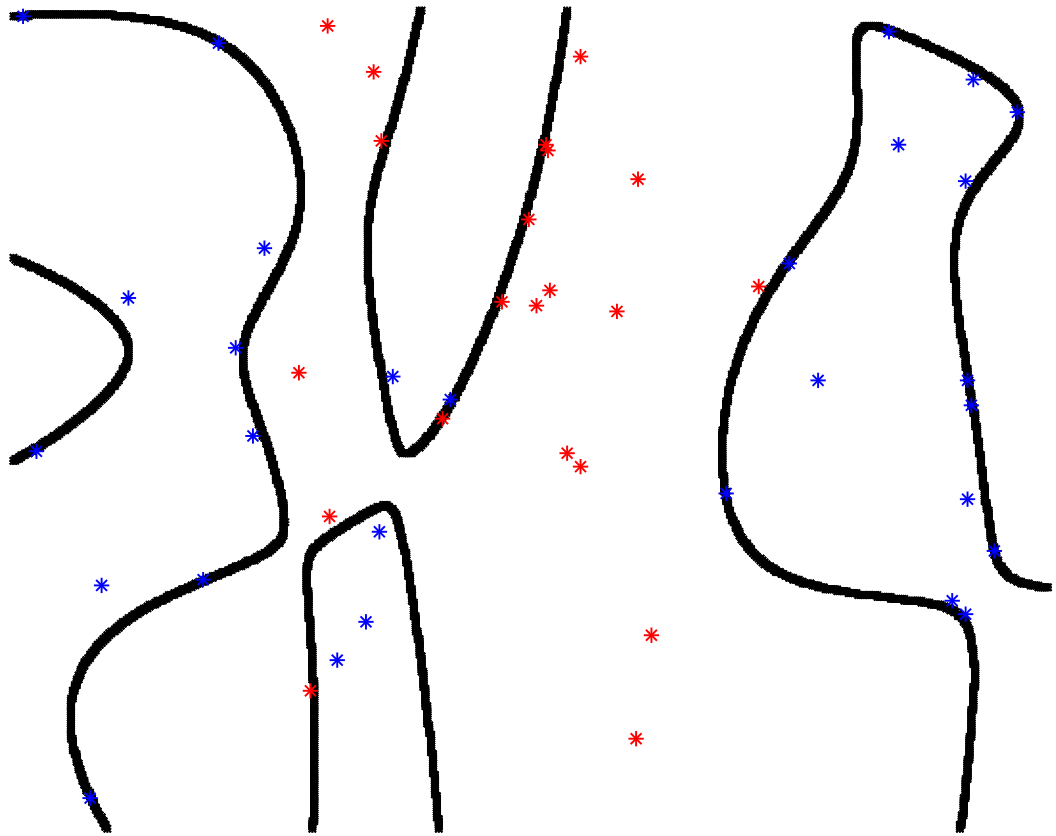
Example – Different polynomial model orders

- "Lifting" example with fewer samples and some mislabels
- Logistic regression (no regularization) polynomial features of degree: 5



Example – Different polynomial model orders

- "Lifting" example with fewer samples and some mislabels
- Logistic regression (no regularization) polynomial features of degree: 6



Overfitting

- Models with higher order polynomials overfit
- Use, e.g., Tikhonov regularization to reduce overfitting

Tikhonov regularization

Regularized problem:

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} \Big(\log(1 + e^{x_i^T w + b}) - y_i (x_i^T w + b) \Big) + \lambda \|w\|_2^2$$

Regularization:

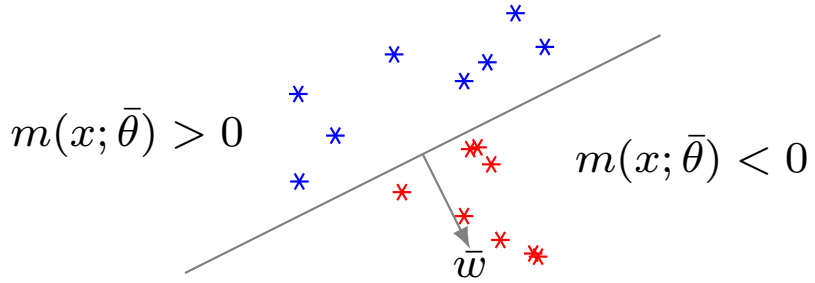
- Regularize only w and not the bias term b
- Why? Model looses shift invariance if also b regularized

Problem properties

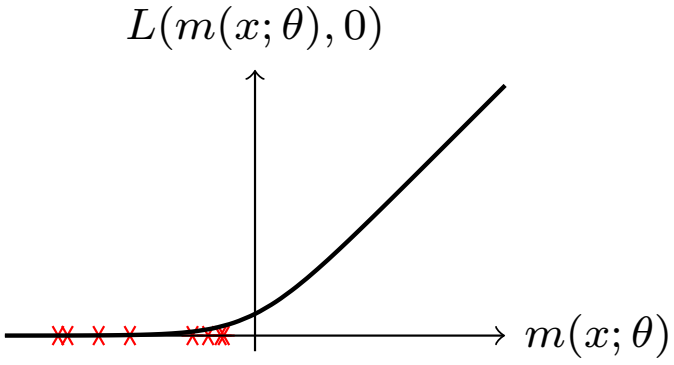
- Problem is strongly convex in $w \Rightarrow$ optimal w exists and is unique
- Optimal b is bounded if examples from both classes exist

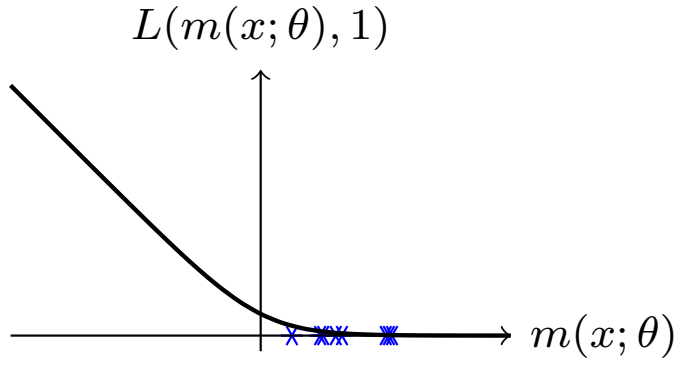
Fully separable data

- Regularization also useful when linearly fully separable data



- Suppose data is fully separable and no regularization, then
  - Smallest possible logistic regression cost is 0 (no regularization)
  - Normal vector $\bar{w}$ to hyperplane exists that fully separates data
  - Normal vector $t\bar{w}$ also separates where $t \to \infty$
  - Model outputs $\to -\infty$ (*) or $\to \infty$ (*) as $t \to \infty \Rightarrow$ cost $\to 0$



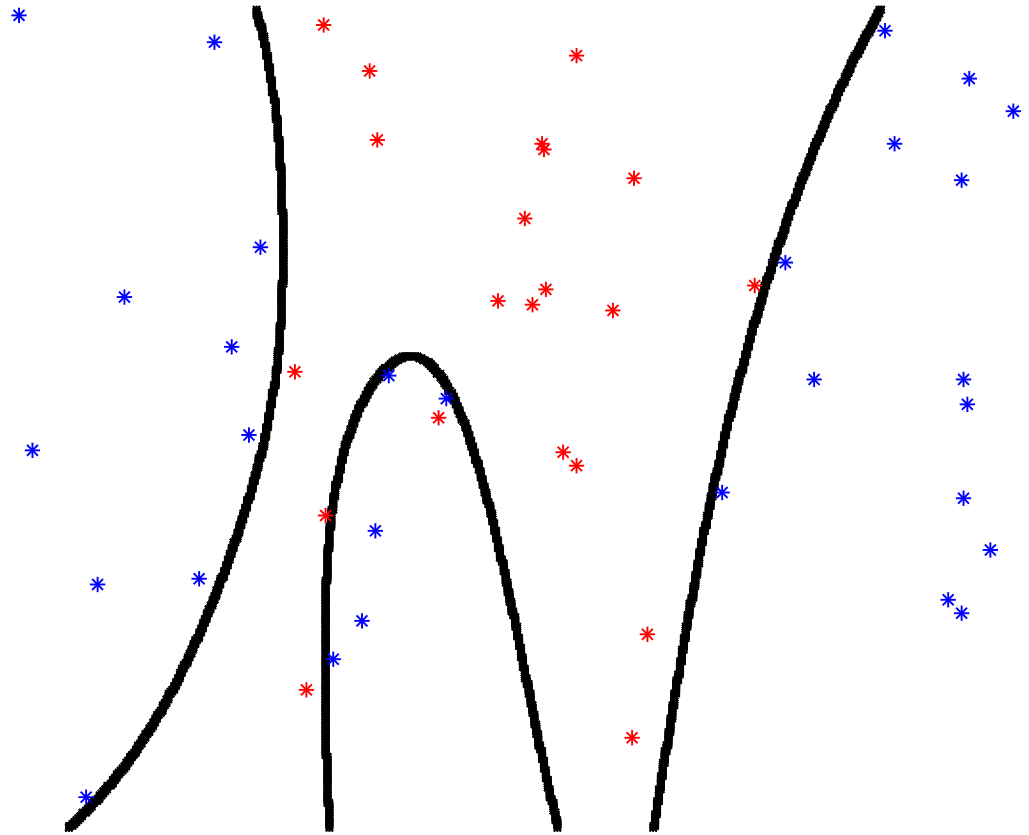


- Regularization gives bounded solution

Example – Different regularization

- Regularized logistic regression and polynomial features of degree 6
- Regularization parameter λ, training cost J, # mislabels in training

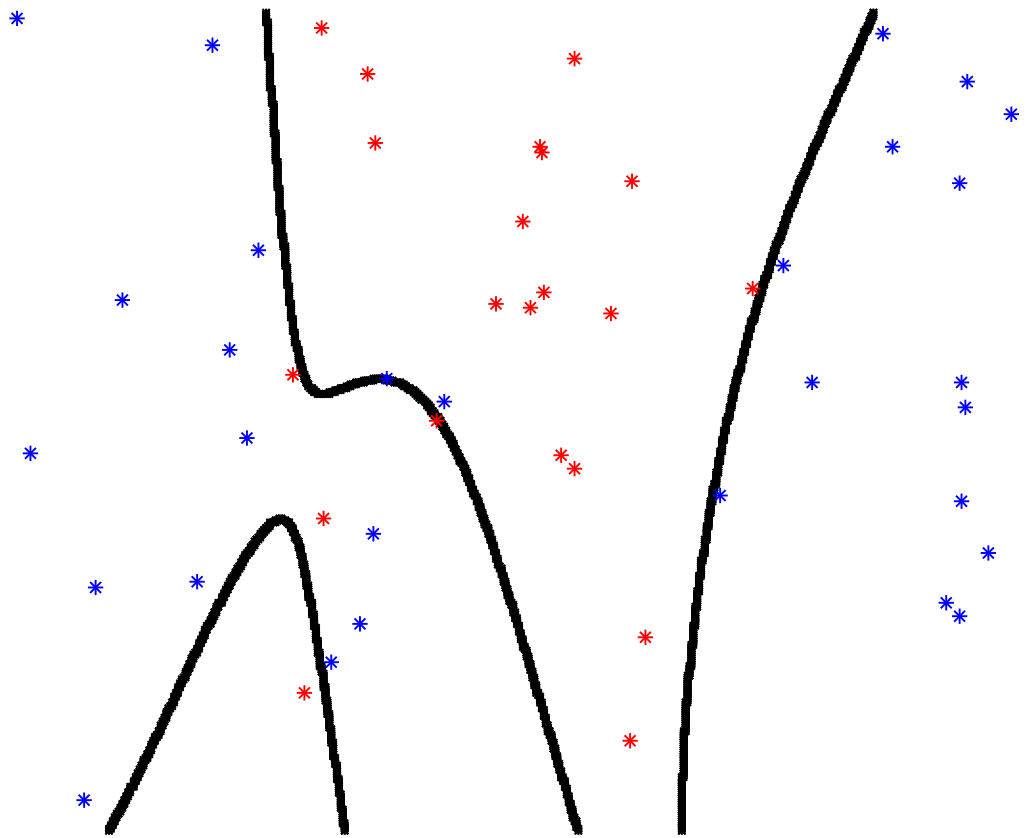
λ	J	# mislabels
0.00001	4.60	1



Example – Different regularization

- Regularized logistic regression and polynomial features of degree 6
- Regularization parameter λ, training cost J, # mislabels in training

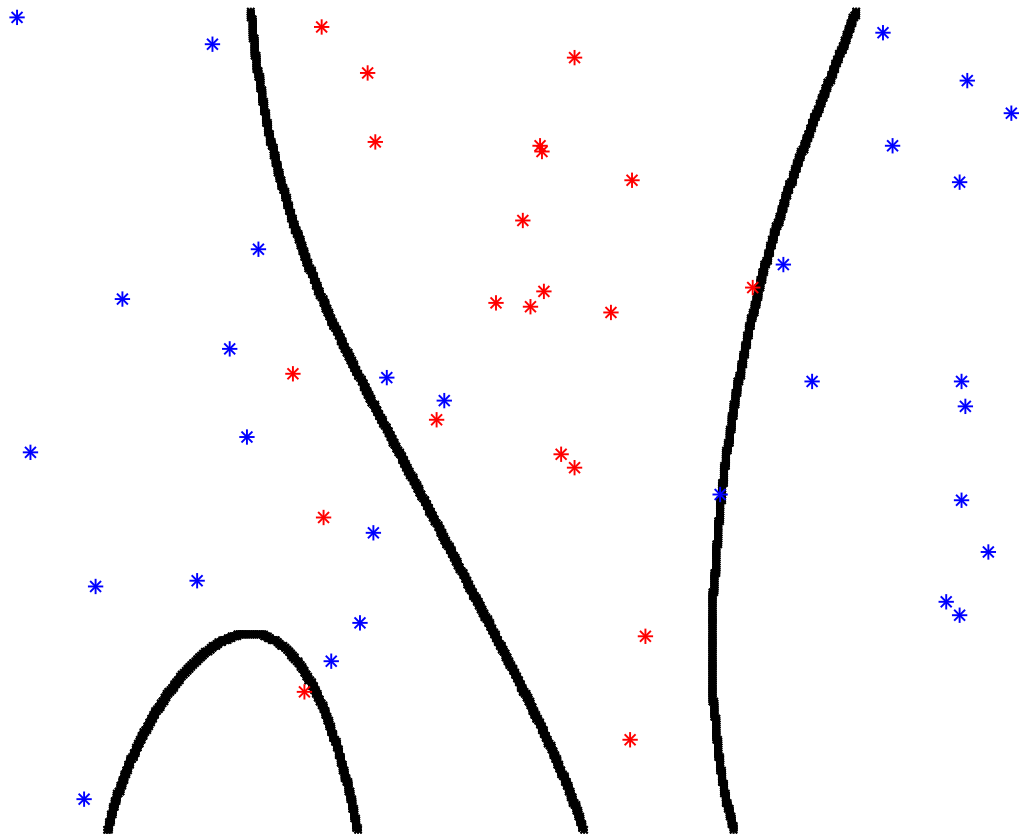
λ	J	# mislabels
0.00006	6.83	5



Example – Different regularization

- Regularized logistic regression and polynomial features of degree 6
- Regularization parameter λ, training cost J, # mislabels in training

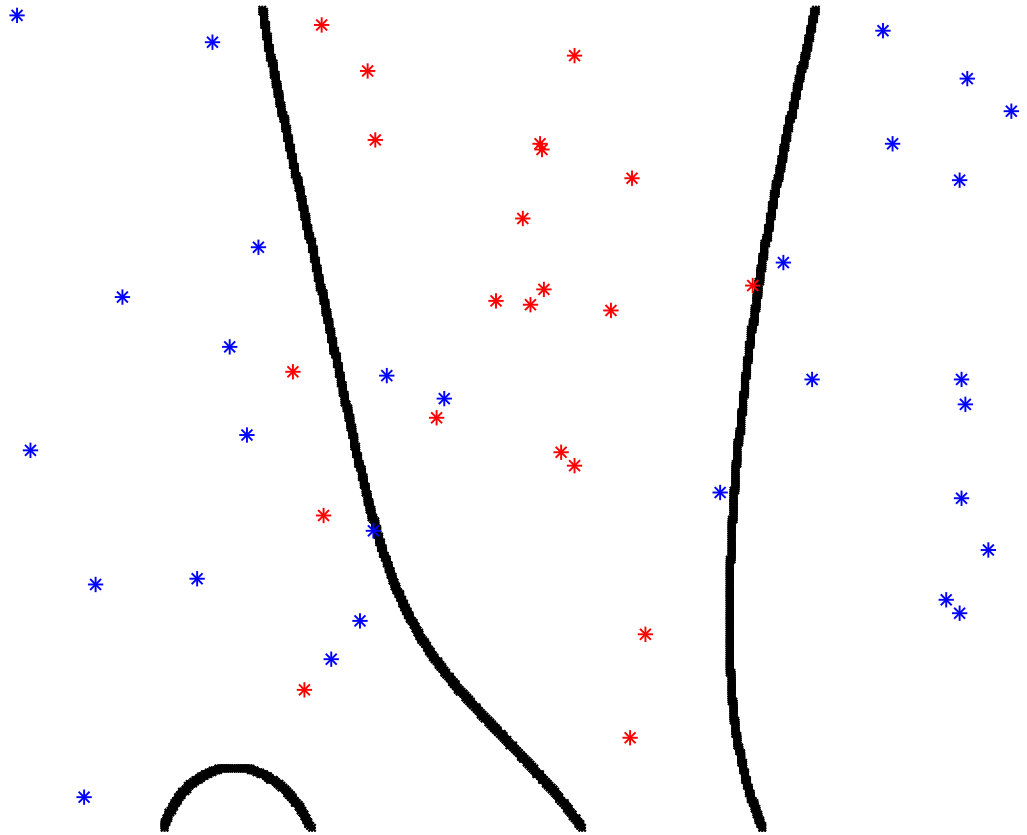
λ	J	# mislabels
0.00036	9.94	5



Example – Different regularization

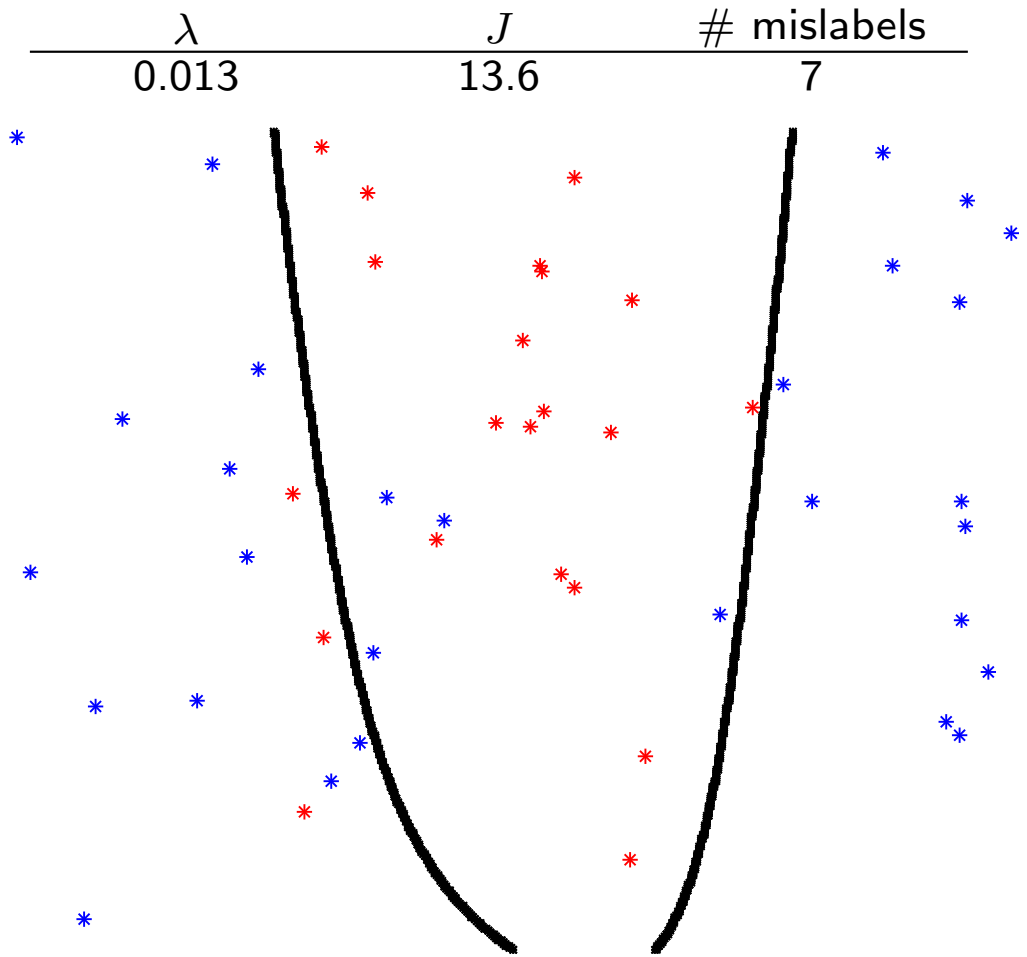
- Regularized logistic regression and polynomial features of degree 6
- Regularization parameter λ, training cost J, # mislabels in training

λ	J	# mislabels
0.0021	12.1	6



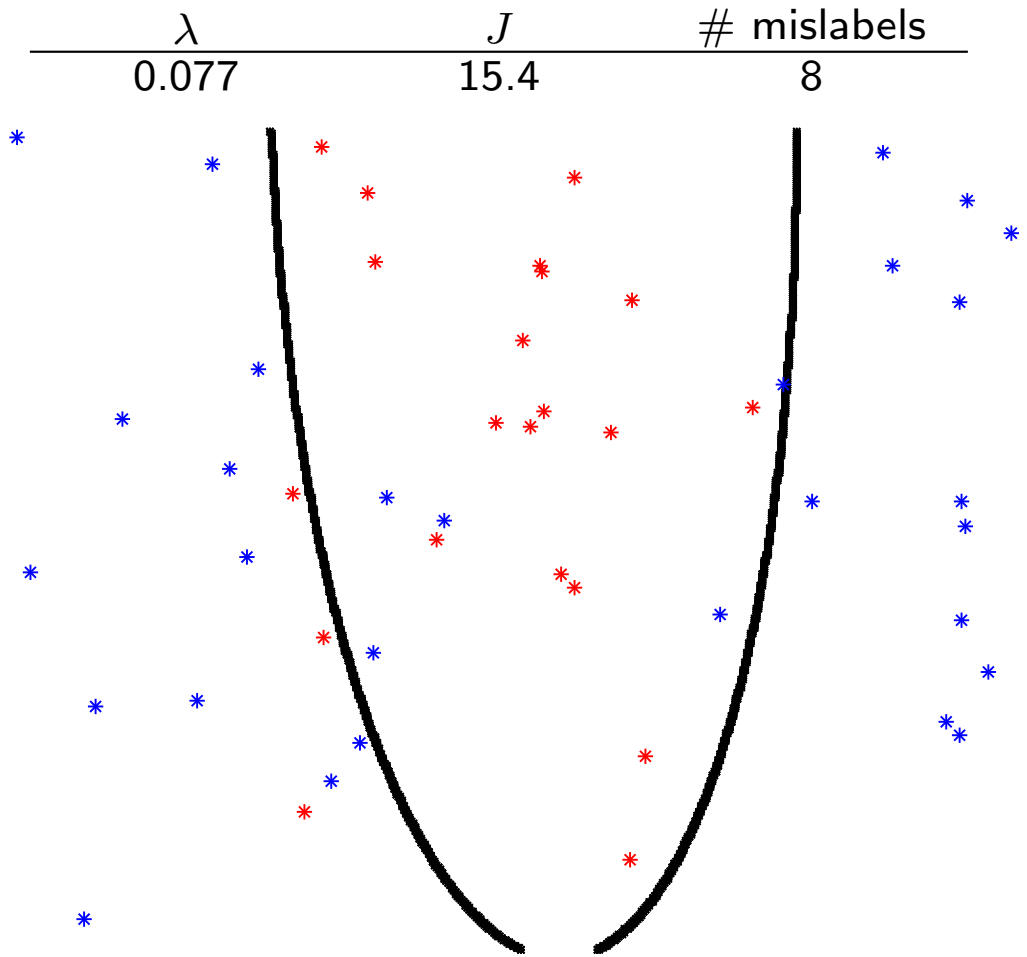
Example – Different regularization

- Regularized logistic regression and polynomial features of degree 6
- Regularization parameter λ, training cost J, # mislabels in training



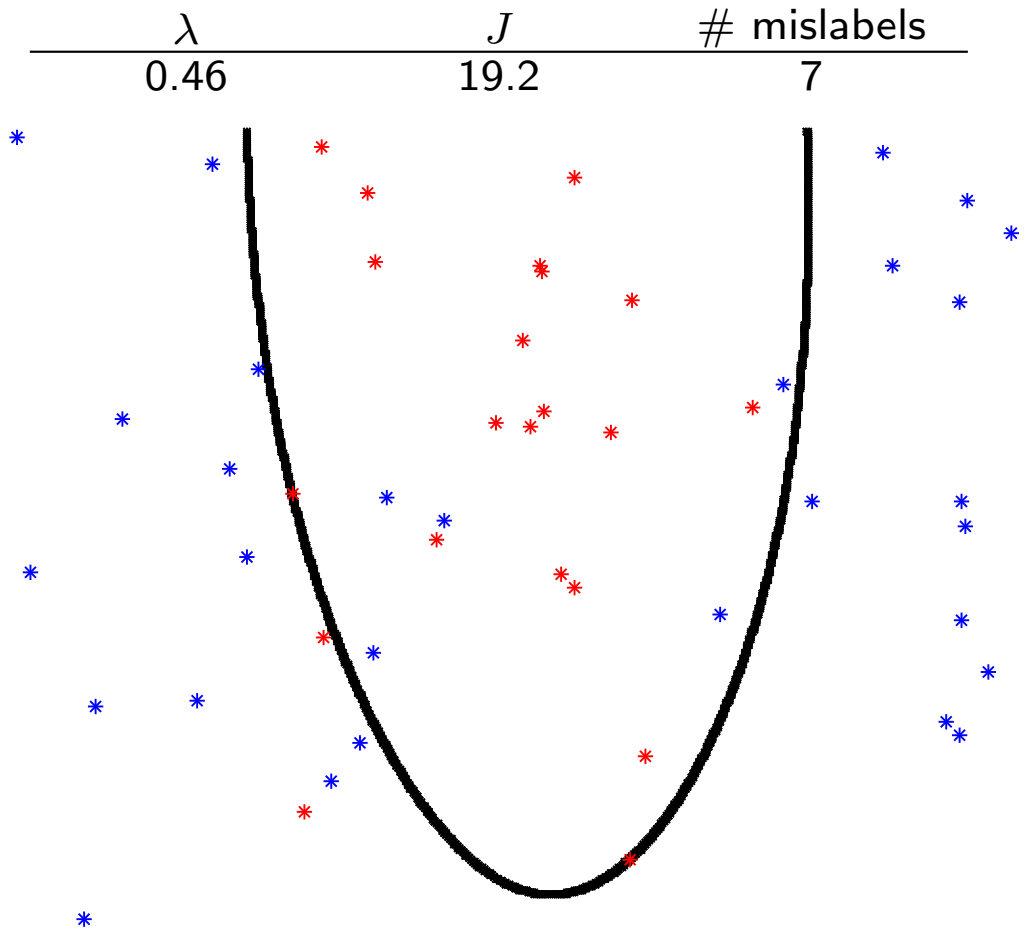
Example – Different regularization

- Regularized logistic regression and polynomial features of degree 6
- Regularization parameter λ, training cost J, # mislabels in training



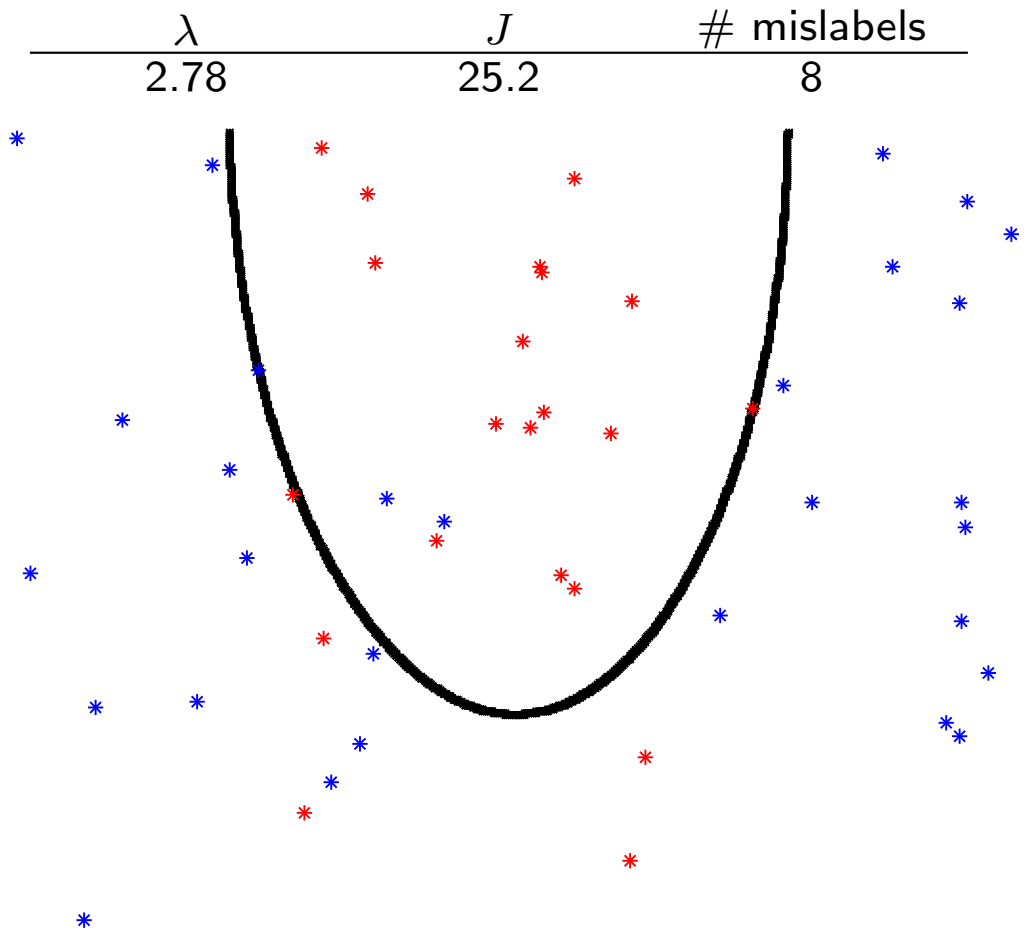
Example – Different regularization

- Regularized logistic regression and polynomial features of degree 6
- Regularization parameter λ, training cost J, # mislabels in training



Example – Different regularization

- Regularized logistic regression and polynomial features of degree 6
- Regularization parameter λ, training cost J, # mislabels in training

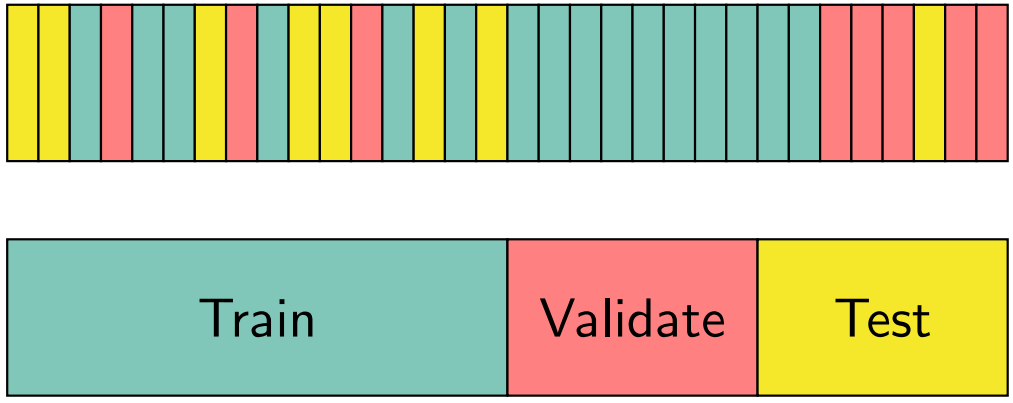


Selecting model

- Model order and regularization parameter are hyperparameters
- Select using holdout or cross validation

Holdout

- Randomize data and assign to train, validate, or test set



Training set:

- Solve training problems with different hyperparameters

Validation set:

- Estimate generalization performance of all trained models
- Use this to select model that seems to generalize best

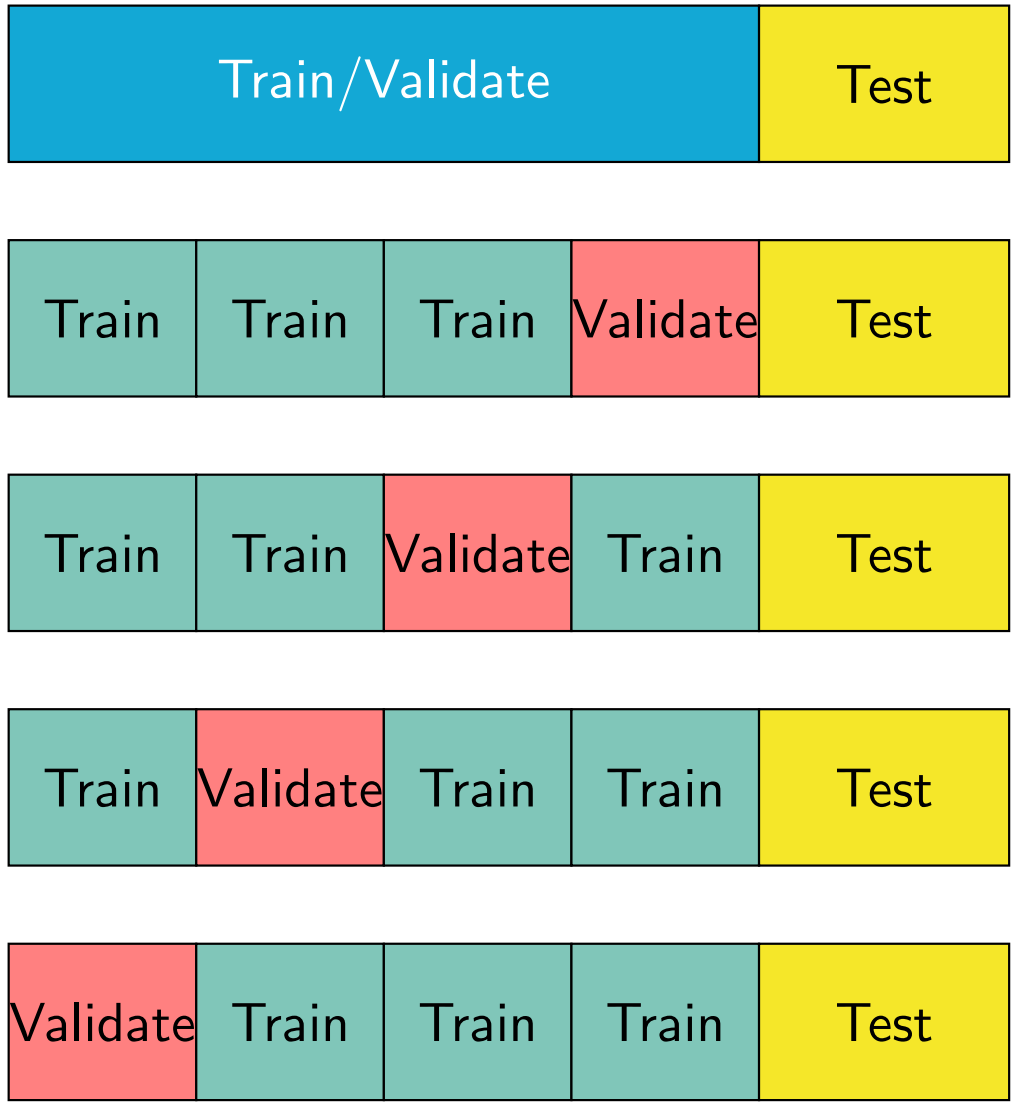
Test set:

- Final assessment on how chosen model generalizes to unseen data
- *Not* for model selection, then final assessment too optimistic

k-fold cross validation

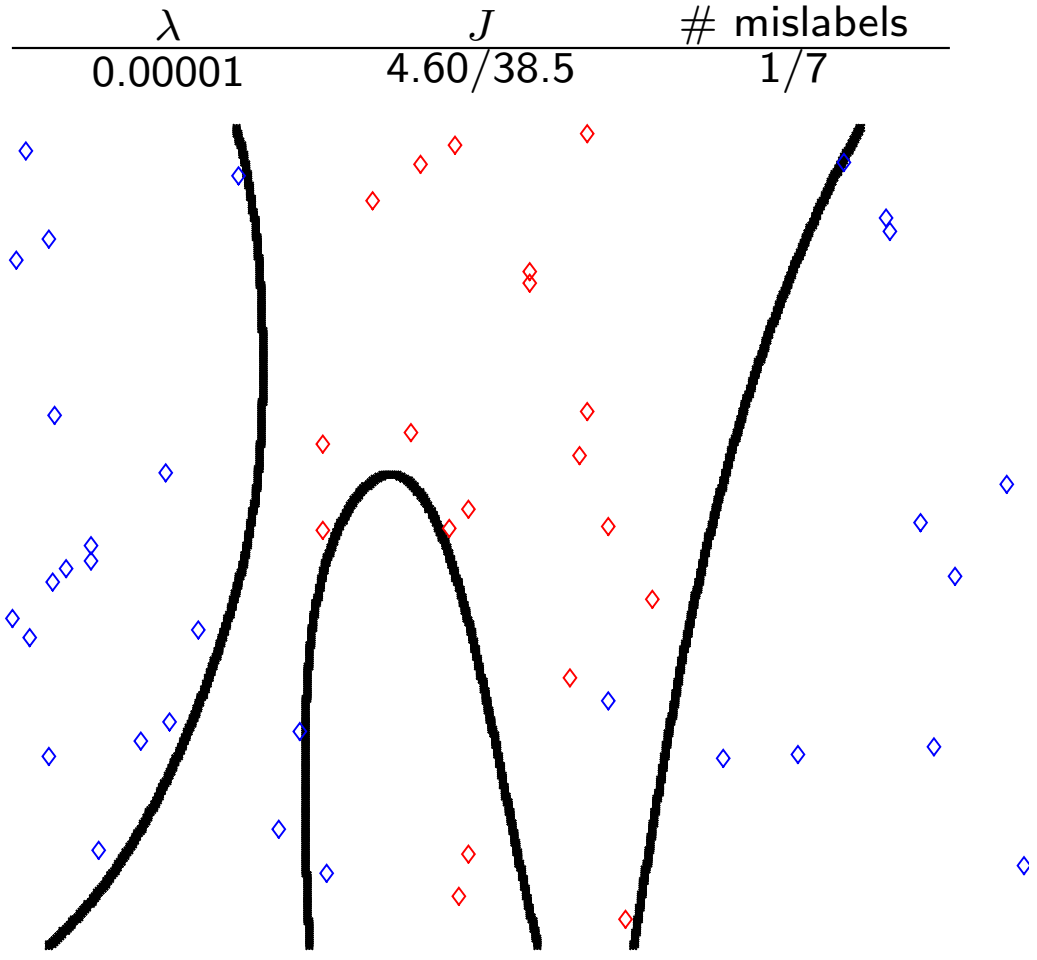
- Similar to hold out – divide first into training/validate and test set
- Divide/validate set into k data chunks
- Train k models with $k-1$ chunks, use k:th chunk for validation
- Loop
  1. Set hyperparameters and train all k models
  2. Evaluate generalization score on its validation data
  3. Sum scores to get model performance
- Select final model hyperparameters based on best score
- Simpler model with slightly worse score may generalize better
- Estimate generalization performance via test set

4-fold cross validation – Graphics



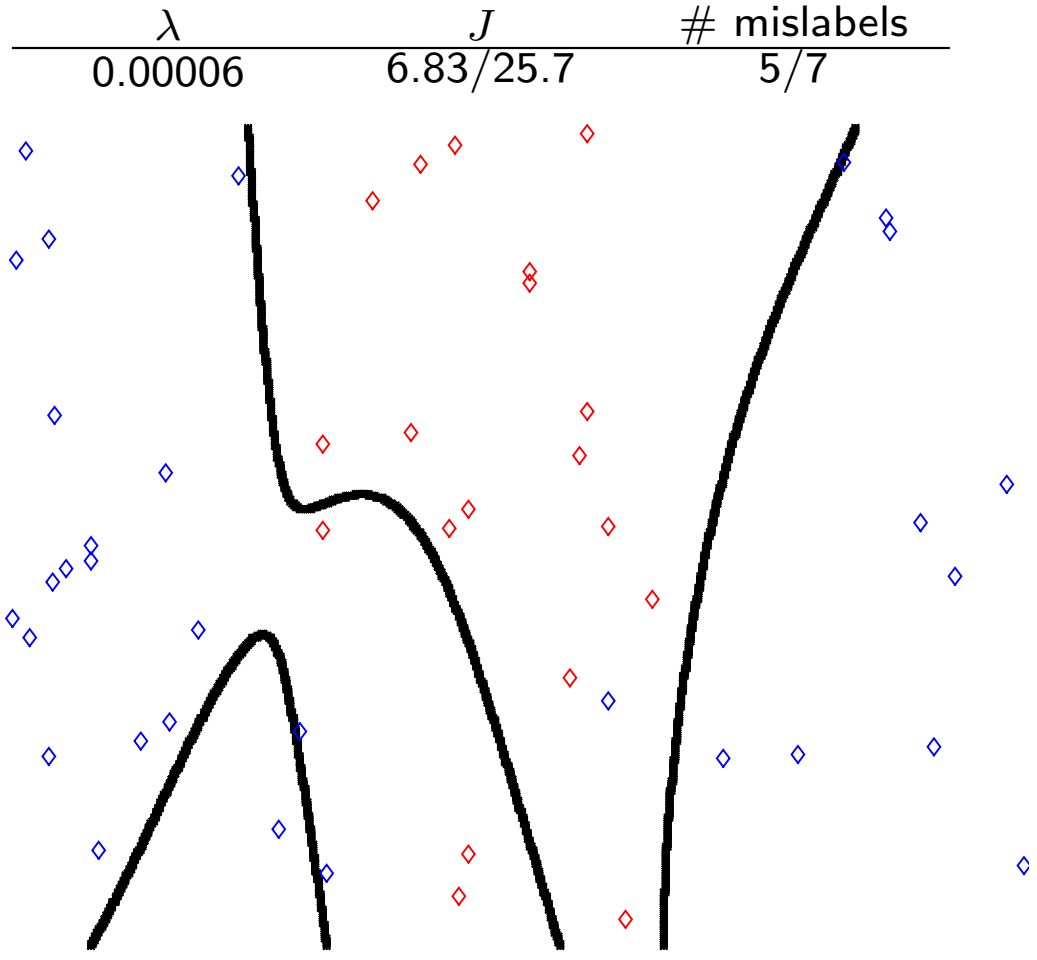
Example – Validation data

- Regularized logistic regression and polynomial features of degree 6
- J and # mislabels specify training/test values



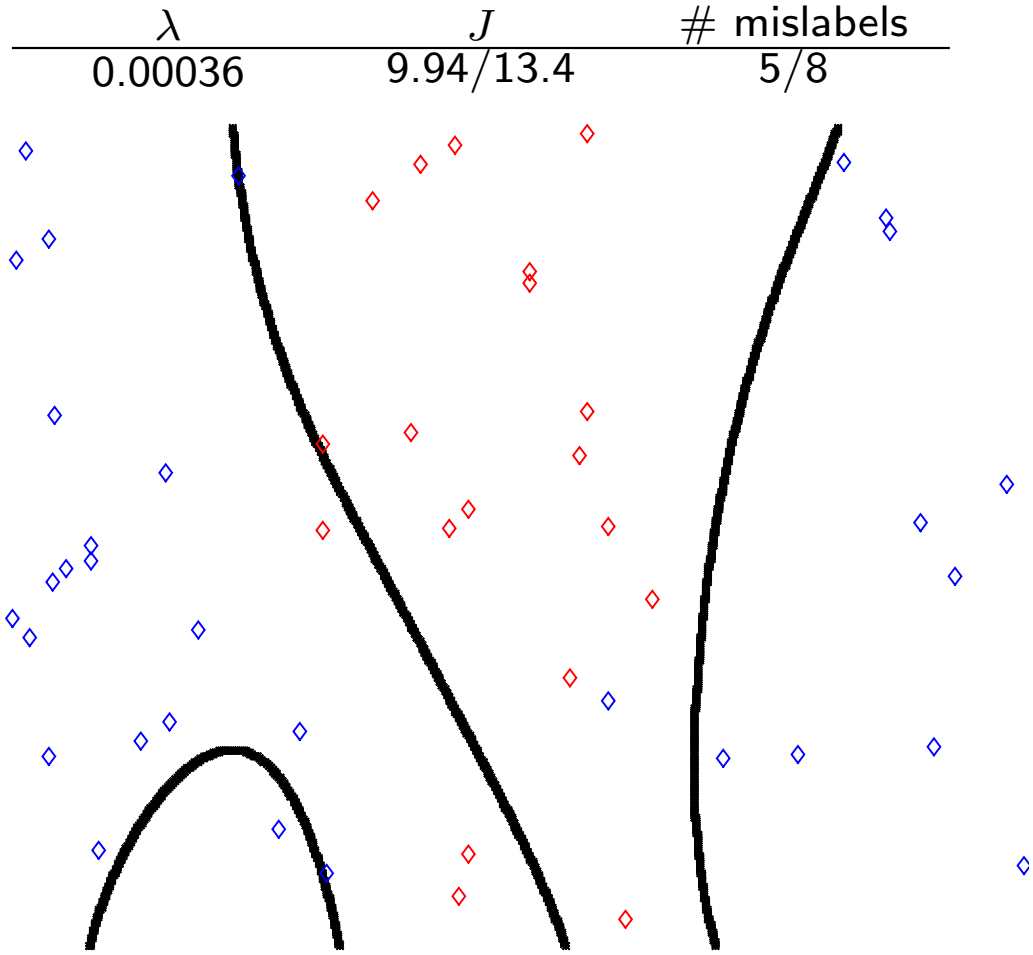
Example – Validation data

- Regularized logistic regression and polynomial features of degree 6
- J and # mislabels specify training/test values



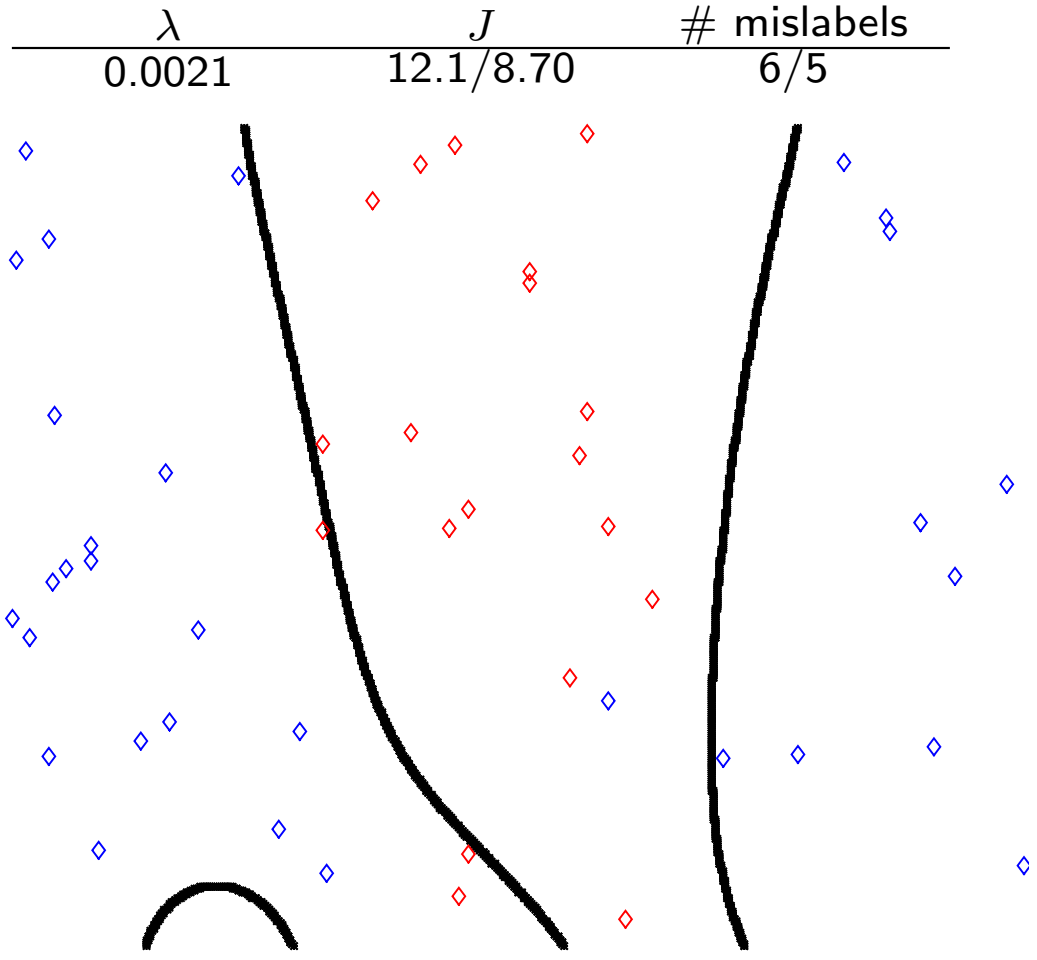
Example – Validation data

- Regularized logistic regression and polynomial features of degree 6
- J and # mislabels specify training/test values



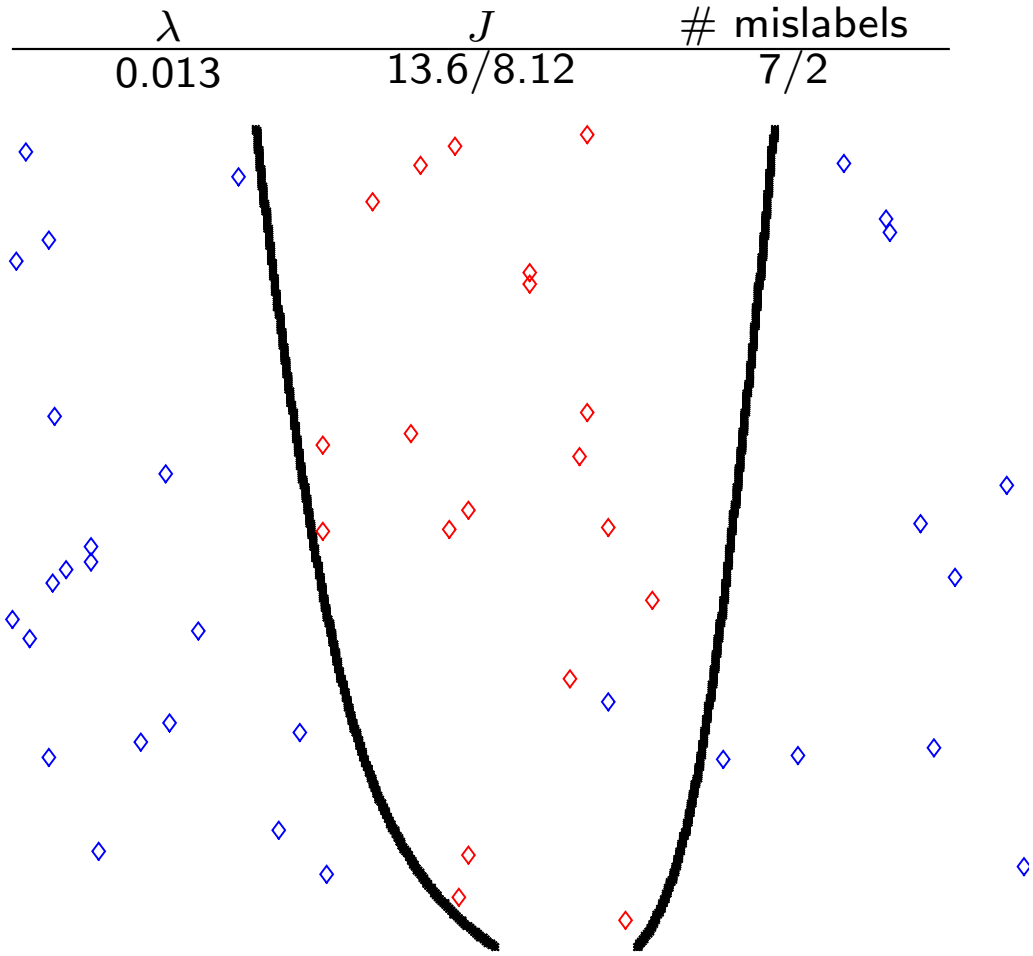
Example – Validation data

- Regularized logistic regression and polynomial features of degree 6
- J and # mislabels specify training/test values



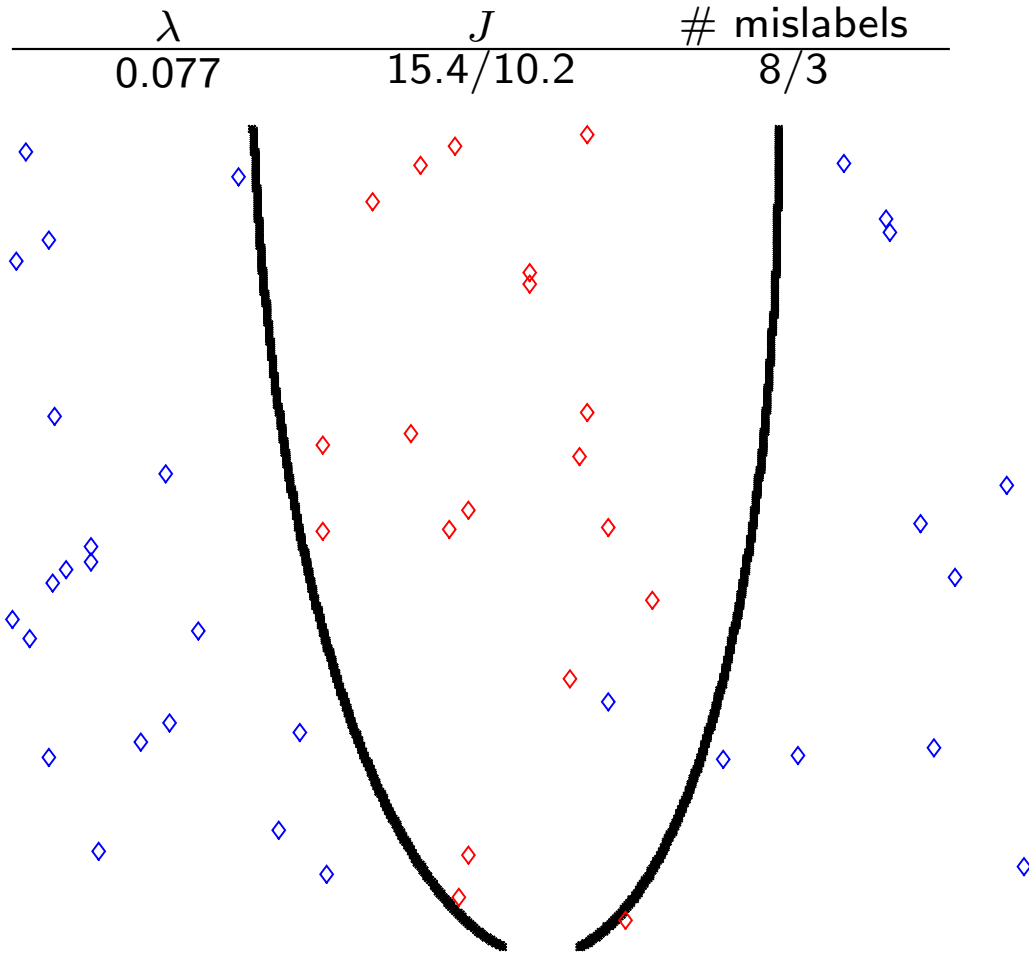
Example – Validation data

- Regularized logistic regression and polynomial features of degree 6
- J and # mislabels specify training/test values



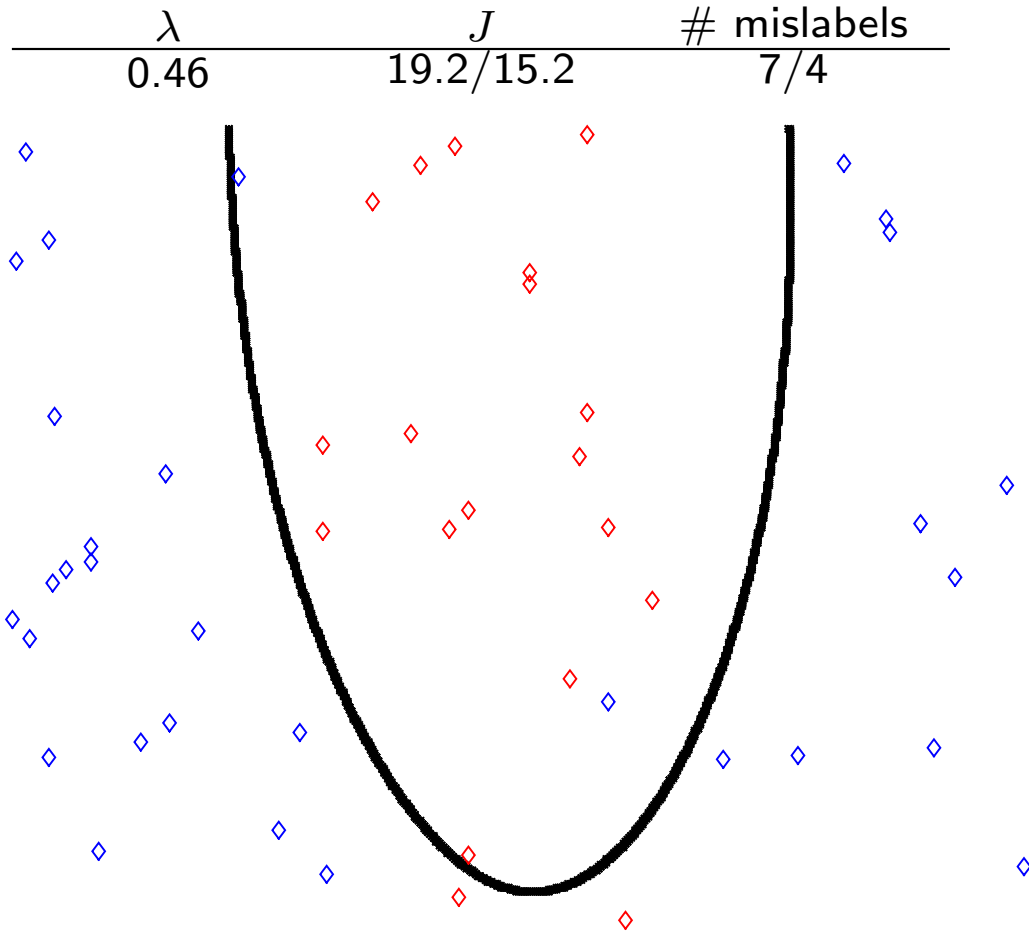
Example – Validation data

- Regularized logistic regression and polynomial features of degree 6
- J and # mislabels specify training/test values



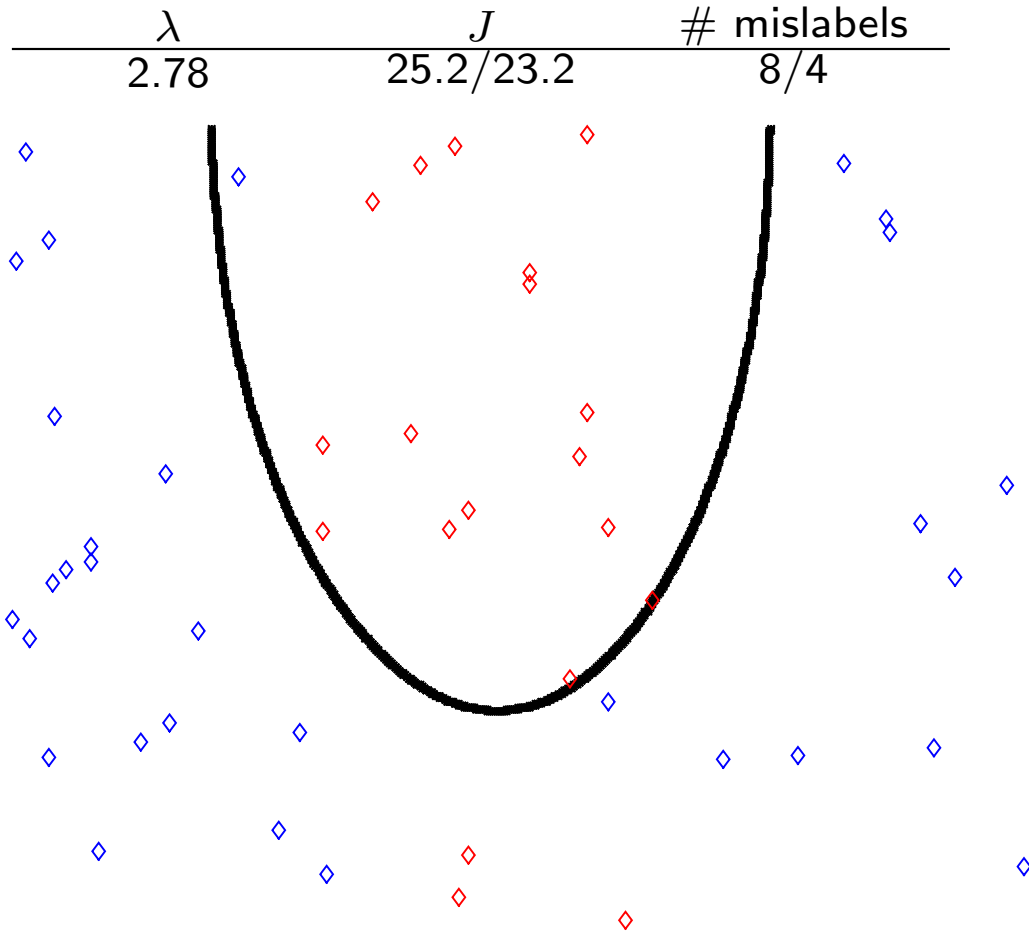
Example – Validation data

- Regularized logistic regression and polynomial features of degree 6
- J and # mislabels specify training/test values



Example – Validation data

- Regularized logistic regression and polynomial features of degree 6
- J and # mislabels specify training/test values

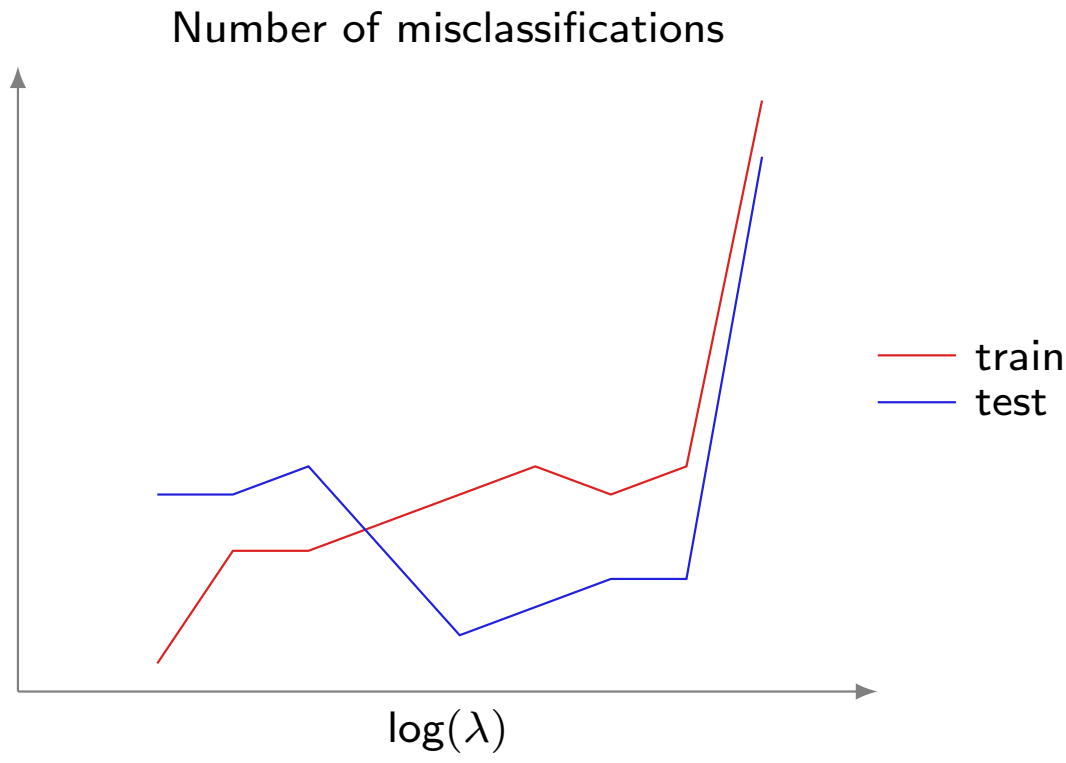


Test vs training error – Cost

- Increasing λ gives lower complexity model
- Overfitting to the left, underfitting to the right
- Select lowest complexity model that gives good generalization

Test vs training error – Classification accuracy

- Increasing λ gives lower complexity model
- Overfitting to the left, underfitting to the right
- Cost often better measure of over/underfitting



Composite optimization

Logistic regression problems are convex problems of the form

$$\underset{\theta}{\text{minimize}}\, f(X\theta) + g(\theta),$$

where

- $f(u) = \sum_{i=1}^{N} (\int \sigma(v_i) dv_i)(u_i) - y_i u_i$ is data misfit term
- $\sigma(u_i) = \frac{1}{1+e^{-u_i}} : \mathbb{R} \to \mathbb{R}$ is a sigmoid function
- X is training data matrix (potentially extended with features)
- g is regularization term (squared 2-norm, 1-norm)

Gradient and function properties

- Gradient of $f \circ X$ satisfies:

$$\begin{aligned}\nabla(f \circ X)(\theta) &= \nabla \sum_{i=1}^{N} \left(\int \sigma(v_i) dv_i \right) (x_i^T \theta) - y_i x_i^T \theta \\ &= \sum_{i=1}^{N} x_i \sigma(x_i^T \theta) - x_i y_i = \sum_{i=1}^{N} x_i (\sigma(x_i^T \theta) - y_i) \\ &= X^T (\sigma(X\theta) - Y)\end{aligned}$$

  where last $\sigma : \mathbb{R}^N \to \mathbb{R}^N$ applies $\frac{1}{1+e^{-u_i}}$ to all elements in $X\theta$

- (Compare to least squares gradient which is same with $\sigma = I$)
- Function and σ properties
  - σ is 0.25 Lipschitz continuous
  - f is convex and 0.25-smooth and $f \circ X$ is $0.25\|X\|^2$-smooth
  - g is convex and possibly nondifferentiable