Data Science and Scientific Computation
Track Core Course

Christoph Lampert

IST AUSTRIA

Institute of Science and Technology

Spring Semester 2016/17

Segment 1, Lecture 4

Overview

Date		no.	Topic
Feb 27	Mon	1	predictive models, least squares regression, model selection, regularization
Mar 1	Wed	2	real data, non-vectorial data, LASSO regression
Mar 6	Mon	3	missing data, nonlinear regression
Mar 8	Wed	4	robust regression, classification,
Mar 13	Mon	5	model evaluation, large-scale model learning
Mar 15	Wed	6	project Q&A
Mar 20	Mon	7	project Q&A
Mar 22	Wed	8	project presentations

Refresher

Summary

General form: regularized least squares regression

$$\min_f \quad \underbrace{\frac{1}{n}\sum_i (f(x_i) - y_i)^2}_{\text{loss}} + \underbrace{\overbrace{\lambda}^{\text{reg.const.}} \; \Omega(f)}_{\text{regularizer}}$$

Models that depends on parameters in a nonlinear way are hard to learn.

Possible model classes that can be learned effectively:

- linear functions: $f(x) = w^\top x + b$
- generalized linear functions: $f(x) = w^\top \phi(x) + b$
- kernelized functions: $f(x) = \sum_i \alpha_i k(x_i, x)$

More about learning linear and nonlinears models

$$f(x) = w^\top x + b \qquad f(x) = w^\top \phi(x) + b$$

other error measures

Least squares regression learns a function $f : \mathcal{X} \to \mathbb{R}$ by minimizing the sum of squared differences (plus a regularizer):

$$\min_{f} \quad \sum_{i=1}^{n} (f(x_i) - y_i)^2 \; + \lambda \, \Omega(w)$$

Least squares regression learns a function $f : \mathcal{X} \to \mathbb{R}$ by minimizing the sum of squared differences (plus a regularizer):

$$\min_f \quad \sum_{i=1}^{n} (f(x_i) - y_i)^2 \; + \lambda \, \Omega(w)$$

Why "the sum of squared differences"?

Least squares regression learns a function $f : \mathcal{X} \to \mathbb{R}$ by minimizing the sum of squared differences (plus a regularizer):

$$\min_f \quad \sum_{i=1}^{n} (f(x_i) - y_i)^2 \; + \lambda \, \Omega(w)$$

Why "the sum of squared differences"?

Looking at "residuals": $r = f(x) - y$:

- it make sense: small r^2 implies $f(x) \approx y$
- r^2 is a nice differentiable function (e.g. $|r|$ is not)
- yields closed-form solutions, if f linear and $\Omega = \lambda \|w\|^2$
- it has a probabilistic interpretation

Probabilistic justification of least squares regression

Assume that the 'real' data generating process is a function with additive **Gaussian** noise, i.e.

$$y = g(x) + \epsilon \qquad \text{for} \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

For fixed x, the probability of seeing any y is

$$p(y|x; g) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(g(x)-y)^2}$$

The likelihood of seeing the values $y_1, \ldots, y_n$ for inputs $x_1, \ldots, x_n$ is

$$\begin{aligned} p(y_1, \ldots, y_n|x_1, \ldots, x_n; g) &= \prod_{i=1}^{n} p(y_i|x_i) \\ &= \frac{1}{(2\pi\sigma^2)^{n/2}} e^{-\frac{1}{2\sigma^2}\sum_i (g(x_i)-y_i)^2} \end{aligned}$$

$$p(y_1, \ldots, y_n | x_1, \ldots, x_n; g) = \text{const} \cdot e^{-\frac{1}{2\sigma^2} \sum_i (g(x_i) - y_i)^2}$$

Maximum (conditional) likelihood principle:

Which function f makes the observed data most likely?

$$\begin{aligned}
&\operatorname*{\mathbf{argmax}}_f \; p(y_1, \ldots, y_n | x_1, \ldots, x_n; f) \\
&\qquad = \operatorname*{\mathbf{argmax}}_f \; \log \; p(y_1, \ldots, y_n | x_1, \ldots, x_n; f) \\
&\qquad = \operatorname*{\mathbf{argmax}}_f \; -\frac{1}{2\sigma^2} \sum_i (f(x_i) - y_i)^2 \\
&\qquad = \operatorname*{\mathbf{argmin}}_f \; \sum_i (f(x_i) - y_i)^2
\end{aligned}$$

Bayesian derivation

$$p(f|x_1,\ldots,x_n,y_1,\ldots,y_n) \overset{\text{Bayes rule}}{=} \frac{p(y_1,\ldots,y_n|x_1,\ldots,x_n;f)p(f)}{p(y_1,\ldots,y_n|x_1,\ldots,x_n)}$$

(assuming that the true f does not depend on $x_1,\ldots,x_n$)

Maximum *a posteriori* (MAP) estimate:

$$\begin{aligned}
&\operatorname*{\mathbf{argmax}}_f p(f|x_1,\ldots,x_n,y_1,\ldots,y_n) \\
&\qquad = \operatorname*{\mathbf{argmax}}_f \log p(y_1,\ldots,y_n|x_1,\ldots,x_n;f) + \log p(f) \\
&\qquad = \operatorname*{\mathbf{argmin}}_f \sum_i (f(x_i)-y_i)^2 + \Omega(f)
\end{aligned}$$

for $\Omega(f) \propto -\log p(f)$

For example: $f(x) = w^\top x + b, \quad p(f) \propto e^{-\|w\|^2}, \quad \Omega(f) = \lambda\|w\|^2$

$$\min \quad \frac{1}{n}\sum_i (f(x_i) - y_i)^2 \; + \lambda\Omega(w)$$

Why shouldn't we minimize the sum of squared differences?

- a) because it's not robust against outliers.

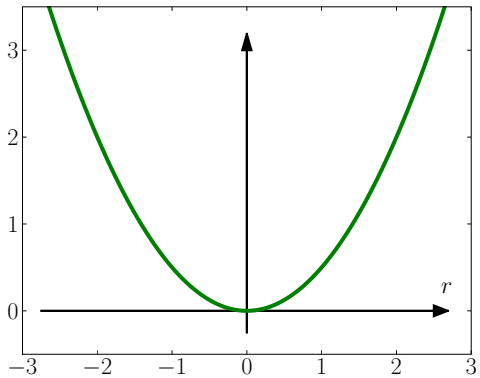
  $\rightarrow$ robust regression

- b) because we want to predict something else than real numbers, e.g. categories $\rightarrow$ classification

Robust Regression

Regression in the presence of outliers

For simplicity of notation,
let's learn just a constant, c



$$\min_c \quad \sum_i (c - y_i)^2$$

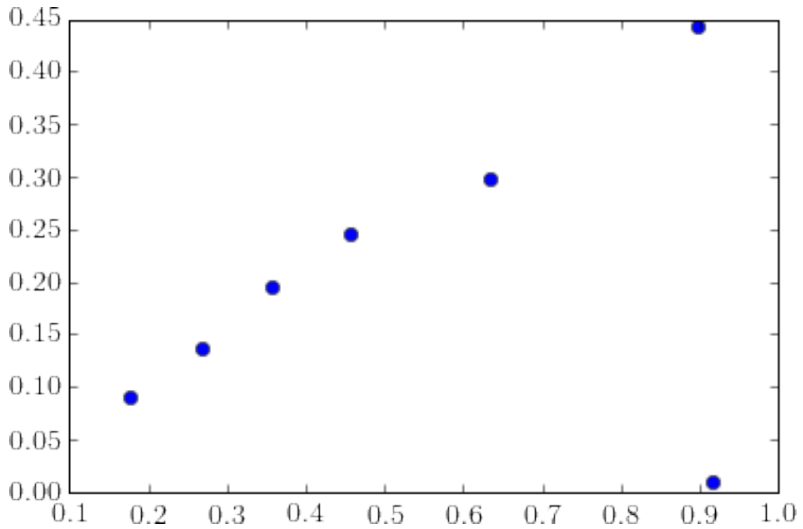
$$= \min_c \quad \|r\|^2 \qquad \text{for} \qquad r = (c - y_1, \ \ldots, \ c - y_n)^\top \in \mathbb{R}^n$$

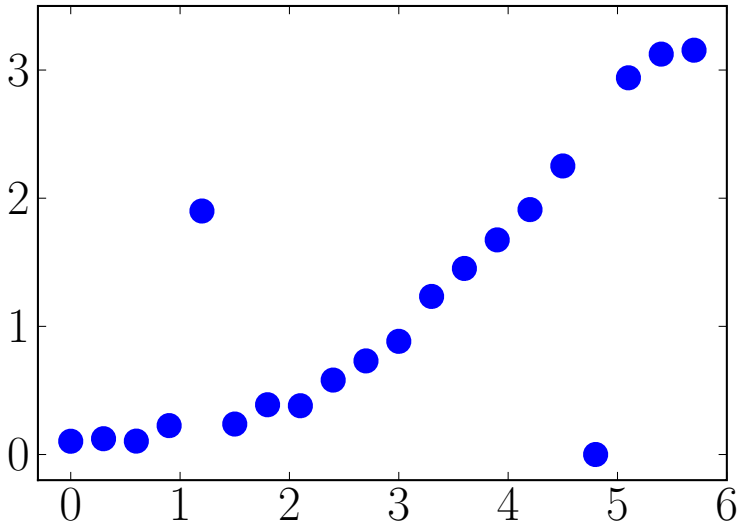
$\|r\|^2$ is smallest if all components of r are approximately the same size:

- e.g. $r = (1, 0, \ldots, 0)$: $\|r\|^2 = 1$
- e.g. $r = (\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n})$: $\|r\|^2 = \frac{1}{n}$

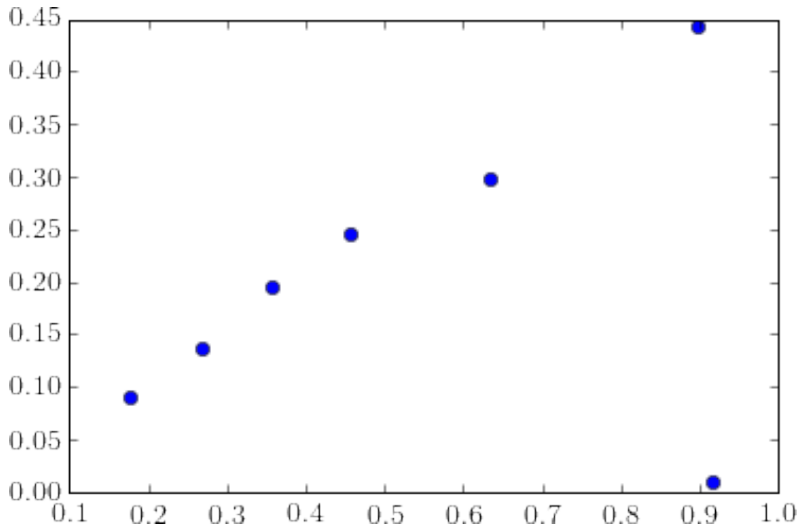
$\rightarrow$ spread the error over all examples

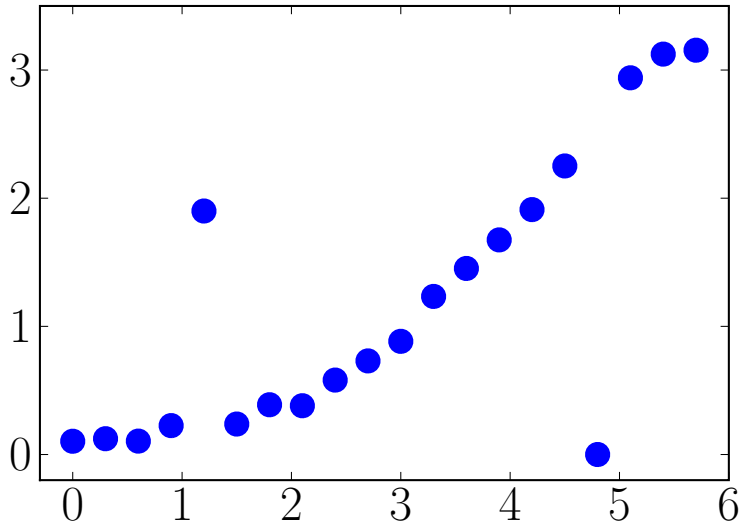
Regression in the presence of outliers



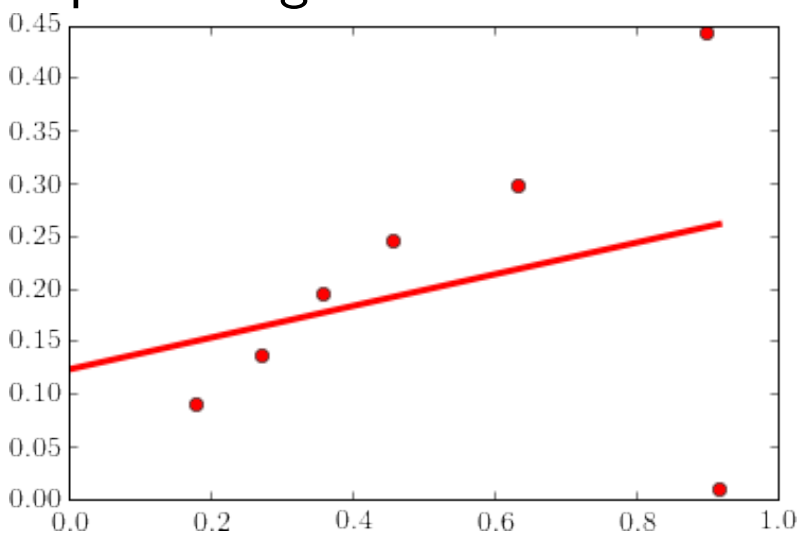


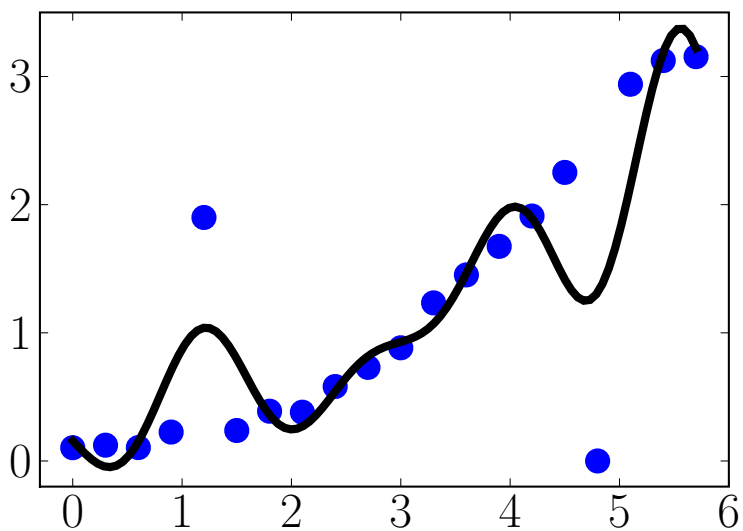
Regression in the presence of outliers





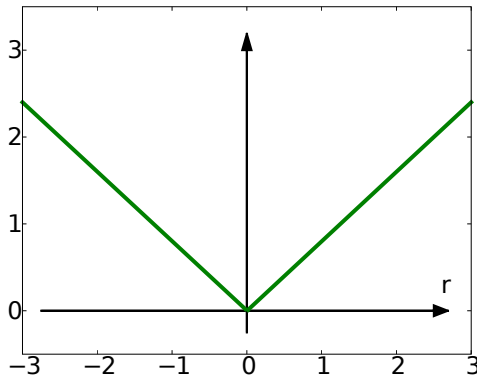
Least squares regression will sacrifice precision elsewhere to fit outliers:





Regression in the presence of outliers

What if we use a different loss, e.g.
sum of absolute values



$$\min_c \quad \sum_i |c - y_i|$$

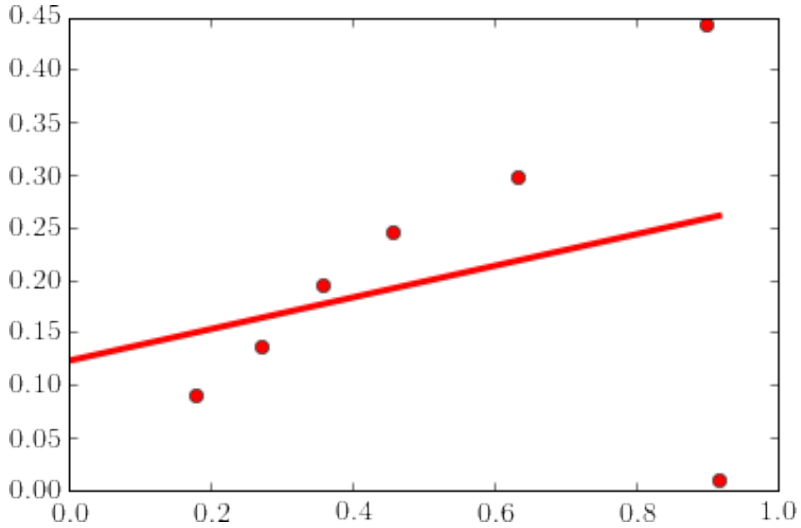
$$= \min_c \quad \|r\|_1 \qquad \text{for} \qquad r = (c - y_1, \ \ldots, \ c - y_n)^\top \in \mathbb{R}^n$$

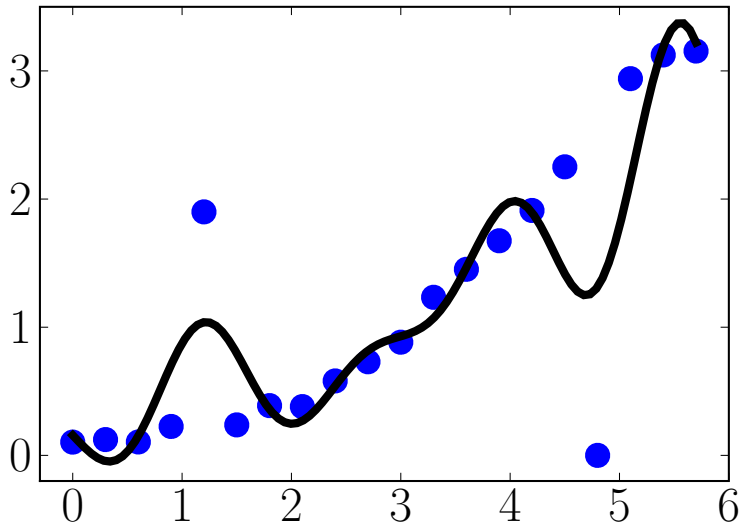
- e.g. $r = (1, 0, \ldots, 0)$: $\|r\|_1 = 1$
- e.g. $r = (\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n})$: $\|r\|_1 = 1$

$\rightarrow \|r\|_1$ does not care if residuals are imbalanced

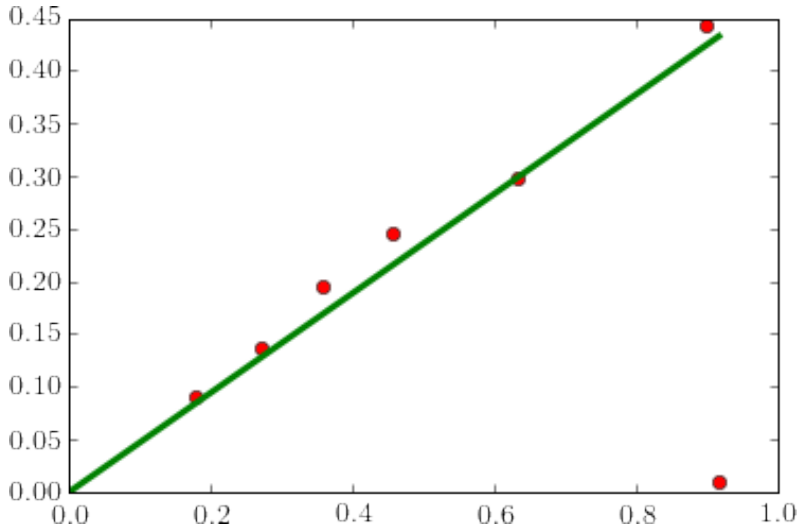
Regression in the presence of outliers

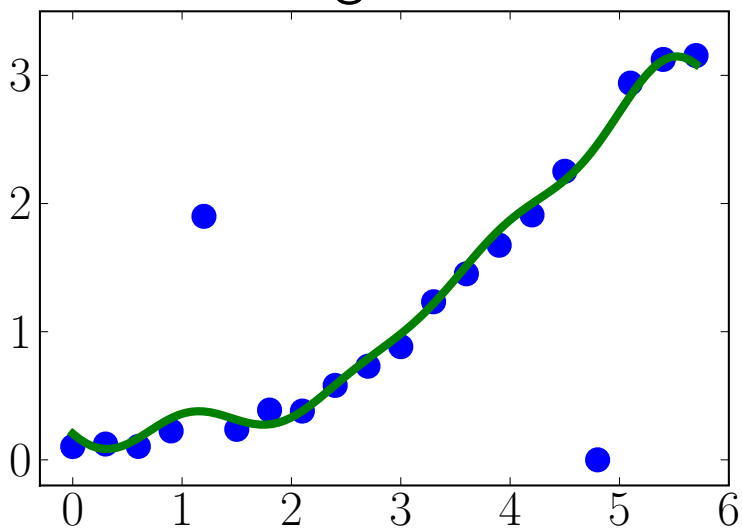
Regression with squared loss:





Regression with absolute value loss is more robust against outliers





Regression in the presence of outliers

General form of regularized regression with ℓ_1 loss:

$$\min_f \sum_i |f(x_i) - y_i| + \lambda\Omega(w)$$

Least Absolute Deviation (LAD) Regression

Regression in the presence of outliers

General form of regularized regression with ℓ_1 loss:

$$\min_f \sum_i |f(x_i) - y_i| + \lambda\Omega(w)$$

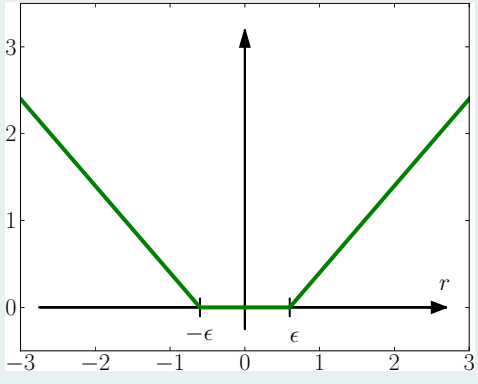
Least Absolute Deviation (LAD) Regression

Variant: Support Vector Regression (SVR) [Vapnik, 1995]

$$\min_w \sum_{i=1}^{n} |w^\top w^\top x_i - y_i|_\epsilon + \lambda\|w\|^2$$

for ϵ-insentive loss (errors below ϵ are free)

$$|t|_\epsilon := \max\{0, |t| - \epsilon\}$$



For $\epsilon = 0$, this becomes the sum of absolute differences.

Can be made nonlinear by explicit preprocessing or kernelization.

Optional: Other robust regressions

1) identify and remove outliers (usually iteratively)

   needs design choices: which/how many of remove

2) least **median** of squares (instead of the mean)

$$\min_f \ \text{median}\left\{ \ (f(x_1) - y_1)^2, \ \ldots, \ (f(x_n) - y_n)^2 \ \right\}$$

   hard to optimize, but truly ignores outliers

3) other robust losses, e.g. truncated squared error

$$\min_f \ \frac{1}{n} \sum_i \min\{ \ (f(x_i) - y_i)^2, C \ \}$$

   hard to optimize, need to choose cut-off point C

Classification

Classification

Each time we have to make a decision, it's a classification problem:

Classification

Each time we have to make a decision, it's a classification problem:

Binary (2-class) classification:

- is an email `spam` or `not spam`?
- is a tumor malicious or benign ?
- is a pixel part of the foreground or of the background?
- ...

Multi-class classification

- car: turn left, turn right, go straight, go back, stop, ...
- ZIP code reading: `'0','1',...,'9'`

Natural quality measure: **prediction accuracy**

$$\text{acc} = \frac{1}{n}\sum_{i=1}^{n}[\![f(x_i) = y_i]\!] \qquad \text{with } [\![P]\!] = \begin{cases} 1 & \text{if } P \text{ is true,} \\ 0 & \text{otherwise.} \end{cases}$$

Binary Classification

Binary classification model: $h : \mathcal{X} \to \{\pm 1\}$ (or: $h : \mathcal{X} \to \{0, 1\}$)

Probabilistic ansatz:

- learn a function $g : \mathcal{X} \to [0, 1]$ that models $p(y = 1|x)$, then decide

$$h(x) = \begin{cases} +1 & \text{if } g(x) \geq 0.5, \\ -1 & \text{otherwise.} \end{cases}$$

Decision theoretic ansatz:

- learn a function $f : \mathcal{X} \to \mathbb{R}$, then threshold

$$h(x) = \begin{cases} +1 & \text{if } f(x) \geq 0, \\ -1 & \text{otherwise.} \end{cases} = \operatorname{sign} f(x)$$

Binary Classification – Logistic Regression

Use a linear model, $f(x) = w^\top x + b$, and make it have range $[0, 1]$ by passing it through a *sigmoid* function, $\sigma(t) = \frac{1}{1+e^{-t}}$

$$g(x) = \frac{1}{1 + e^{-f(x)}}$$

- $g(x) \in [0, 1]$, so we can use it to model $p(y = 1|x)$:
  - $f(x) \to -\infty \quad \Rightarrow \quad g(x) \to 0$
  - $f(x) = 0 \quad \Rightarrow \quad g(x) = 0.5$
  - $f(x) \to +\infty \quad \Rightarrow \quad g(x) \to 1$
- $p(y = 1|x) = g(x) = \frac{1}{1+e^{-f(x)}}$
- $p(y = -1|x) = 1 - g(x) = \frac{e^{-f(x)}}{1+e^{-f(x)}} = \frac{1}{1+e^{f(x)}}$

We can unify both:

$$p(y|x) = \frac{1}{1 + e^{-yf(x)}}$$

Binary Classification – Logistic Regression

Fitting the model: maximum (conditional) likelihood!

$$p(y_1, \ldots, y_n | x_1, \ldots, x_n; f) = \prod_{i=1}^{n} p(y_i | x_i; f) = \prod_{i=1}^{n} \frac{1}{1 + e^{-y_i f(x_i)}}$$

$$-\log p(y_1, \ldots, y_n | x_1, \ldots, x_n; f) = \sum_i \log(1 + e^{-y_i f(x_i)})$$

Which function f makes the observed data most likely?

$$\begin{aligned} &\mathbf{argmax}_f \; p(y_1, \ldots, y_n | x_1, \ldots, x_n; f) \\ &= \mathbf{argmin}_f \; \sum_i \log(1 + e^{-y_i f(x_i)}) \end{aligned}$$

With regularization, minimize

$$\sum_{i=1}^{n} \log(1 + e^{-y_i f(x_i)}) \quad + \quad \lambda \Omega(f)$$

Binary Classification – Logistic Regression

Fitting the model: maximum (conditional) likelihood!

$$p(y_1, \ldots, y_n | x_1, \ldots, x_n; f) = \prod_{i=1}^{n} p(y_i | x_i; f) = \prod_{i=1}^{n} \frac{1}{1 + e^{-y_i f(x_i)}}$$

$$-\log p(y_1, \ldots, y_n | x_1, \ldots, x_n; f) = \sum_i \log(1 + e^{-y_i f(x_i)})$$

Which function f makes the observed data most likely?

$$\begin{aligned} &\mathbf{argmax}_f \; p(y_1, \ldots, y_n | x_1, \ldots, x_n; f) \\ &= \mathbf{argmin}_f \; \sum_i \log(1 + e^{-y_i f(x_i)}) \end{aligned}$$

With regularization, minimize

$$\underbrace{\sum_{i=1}^{n} \log(1 + e^{-y_i f(x_i)})}_{\text{logistic loss (or log-loss)}} + \underbrace{\lambda \Omega(f)}_{\text{regularizer}}$$

(Regularized) Logistic Regression

Solve

$$\min_w \quad \sum_i \log(1 + e^{-y_i w^\top x_i}) \; + \; \lambda \Omega(f)$$

Make predictions using:

$$c(x) = \begin{cases} 1 & \text{if } p(y = 1|x) > 0.5 \\ -1 & \text{otherwise} \end{cases} \qquad = \operatorname{sign} w^\top x.$$

- very efficient to train
- usually very good classification performance
- easy to make multi-class version
- provides probability estimates
- can be kernelized for non-linear classification (but not so common)
- exists in all relevant toolboxes

```
from sklearn.linear_model import LogisticRegression
```

Decision theoretic alternative:

Instead of log-loss, use *hinge* loss: $\ell(t) = \max\{0, 1 - t\}$.

Support Vector Machine

Solve

$$\min_w \quad \sum_{i=1}^{n} \max\{0, 1 - y_i w^\top x_i\} \quad + \quad \lambda\Omega(f)$$

Make predictions using: $c(x) = \operatorname{sign} w^\top x$.

- very efficient to train
- usually very good classification performance
- doesn't provide probabilities
- kernelized variant is more efficient than kernelized logistic regression

```
from sklearn.svm import LinearSVC,SVC
```

Multiclass classification

Multiclass Classification – One-versus-rest reduction

Classification problems with M classes:

- Training samples $\{x_1, \ldots, x_n\} \subset \mathcal{X}$,
- Training labels $\{y_1, \ldots, y_n\} \subset \{1, \ldots, M\}$,
- Task: learn a prediction function $f : \mathcal{X} \to \{1, \ldots, M\}$.

Multiclass Classification – One-versus-rest reduction

Classification problems with M classes:

- Training samples $\{x_1, \ldots, x_n\} \subset \mathcal{X}$,
- Training labels $\{y_1, \ldots, y_n\} \subset \{1, \ldots, M\}$,
- Task: learn a prediction function $f : \mathcal{X} \to \{1, \ldots, M\}$.

One-versus-rest construction:

- train one binary classifier $g_c : \mathcal{X} \to \mathbb{R}$ for each class c:
  - all samples with class label c are positive examples
  - all other samples are negative examples
- classify by finding maximal response

$$f(x) = \operatorname*{\mathbf{argmax}}_{c=1,\ldots,M} g_c(x)$$

Advantage: easy to implement, parallel, works well in practice

Disadvantage: with many classes, training sets become unbalanced.
no explicit *calibration* of scores between different g_c

Multiclass Classification – All-versus-all reduction

Classification problems with M classes:

- Training samples $\{x_1, \ldots, x_n\} \subset \mathcal{X}$,
- Training labels $\{y_1, \ldots, y_n\} \subset \{1, \ldots, M\}$,
- Task: learn a prediction function $f : \mathcal{X} \to \{1, \ldots, M\}$.

Multiclass Classification – All-versus-all reduction

Classification problems with M classes:

- Training samples $\{x_1, \ldots, x_n\} \subset \mathcal{X}$,
- Training labels $\{y_1, \ldots, y_n\} \subset \{1, \ldots, M\}$,
- Task: learn a prediction function $f : \mathcal{X} \to \{1, \ldots, M\}$.

All-versus-all construction:

- train one classifier, $g_{ij} : \mathcal{X} \to \mathbb{R}$, for each pair of classes $1 \leq i < j \leq M$, in total $\frac{m(m-1)}{2}$ prediction functions
- classify by voting

$$f(x) = \operatorname*{\mathbf{argmax}}_{m=1,\ldots,M} \#\{i \in \{1, \ldots, M\} : g_{m,i}(x) > 0\},$$

(writing $g_{j,i} = -g_{i,j}$ for $j > i$ and $g_{j,j} = 0$)

Advantage: small and balanced training problems, parallel, works well.

Disadvantage: number of classifiers grows quadratically in classes.

Multiclass Logistic Regression

- Data: samples $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, labels $\{y_1, \ldots, y_n\} \subset \{1, \ldots, M\}$
- Task: learn a prediction function $f : \mathcal{X} \to \{1, \ldots, M\}$.

Multiclass Logistic Regression

- Data: samples $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, labels $\{y_1, \ldots, y_n\} \subset \{1, \ldots, M\}$
- Task: learn a prediction function $f : \mathcal{X} \to \{1, \ldots, M\}$.

Model of the conditional probability $p(y|x)$:

$$p(y|x) = \frac{e^{w_y^\top x}}{\sum_{k=1}^{M} e^{w_k^\top x}} \qquad \text{for } y = 1, \ldots, M$$

parameterized by M weights vectors, $w_1, \ldots, w_M \in \mathbb{R}^d$.

Multiclass Logistic Regression

Fit model parameters by maximizing conditional likelihood:

$$\min_{w_1,\ldots,w_M} \quad \sum_{i=1}^{n} \log \sum_{k=1}^{M} \exp\left(w_k^\top x_i \; - \; w_{y_i}^\top x_i\right) \quad + \quad \lambda \sum_{k=1}^{M} \Omega(w_k)$$

Make predictions using $f(x) = \mathbf{argmax}_y \, p(y|x) = \mathbf{argmax}_y \;\; w_y^\top x$.

Multiclass Logistic Regression

- Data: samples $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, labels $\{y_1, \ldots, y_n\} \subset \{1, \ldots, M\}$
- Task: learn a prediction function $f : \mathcal{X} \to \{1, \ldots, M\}$.

Model of the conditional probability $p(y|x)$:

$$p(y|x) = \frac{e^{w_y^\top x}}{\sum_{k=1}^{M} e^{w_k^\top x}} \qquad \text{for } y = 1, \ldots, M$$

parameterized by M weights vectors, $w_1, \ldots, w_M \in \mathbb{R}^d$.

Multiclass Logistic Regression

Fit model parameters by maximizing conditional likelihood:

$$\min_{w_1,\ldots,w_M} \quad \sum_{i=1}^{n} \log \sum_{k=1}^{M} \exp\left(w_k^\top x_i - w_{y_i}^\top x_i\right) \quad + \quad \lambda \sum_{k=1}^{M} \Omega(w_k)$$

Make predictions using $f(x) = \mathbf{argmax}_y \, p(y|x) = \mathbf{argmax}_y \; w_y^\top x$.

Existing packages, e.g. `sklearn.linear_model.LogisticRegression`