Machine Learning

Section ~~8~~ 6: Linear Regression

SLIDES BY Stefan Harmeling

~~3. November 2021~~

TODAY

So far: 1. Guess prob. distr from data

Today: 2. Guess functions from observed input/output pairs (X,Y)

Probabilistically: 1. $P(X,Y)$

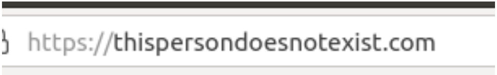
2. $P(Y|X)$ regression/classification

generative model

discriminative model

can draw random samples

e.g.



https://thispersondoesnotexist.com

Overview

- Regression
- Linear regression
- Maximum likelihood estimation
- Ridge regression
- Bayesian linear regression
- Alternatives

This lecture is based on Chapter 7 of Kevin Murphy's textbook "Machine Learning, A Probabilistic Perspective"

Regression:

$X_1, \dots, X_n$ explanatory variables / "independent variables" / regressors / features / ...

Y response variable / dependent var.

$f(x_1, \dots, x_n \mid \theta)$ parametrized family of functions.

Suppose $Y = f(X_1, \dots, X_n \mid \theta) + \boxed{\varepsilon}$ error term

today & often: $\varepsilon \sim$ Gaussian

$E\varepsilon = 0$ ← can be seen as requirement on param. fam.

Find the θ that fulfills this equation "best".

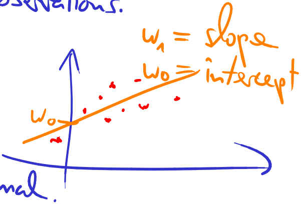
Linear regression: Basic case

$X, Y \in \mathbb{R}$

Given $(x_1, y_1), \dots, (x_N, y_N)$, want to find linear function that best fits to these observations.

$$Y = w_0 + w_1 \cdot X$$

„Best fit": $\underbrace{\sum_{i=1}^{N} (Y_i - (w_0 + w_1 x_i))^2}_{\text{squared error}}$ minimal.



⇝ Find w_0, w_1 minimizing this expression.

Suppose $w_0 = 0$: minimize $\sum_{i=1}^{N} (y_i - w_1 x_i)^2$

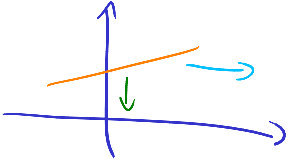
$$0 \overset{!}{=} \frac{\partial}{\partial w_1} (\text{this}) = \sum_{i=1}^{N} 2 (y_i - w_1 x_i)(-x_i)$$

$$= \sum_{i=1}^{N} \left(w_1 x_i^2 - y_i x_i \right) \cdot 2$$

$$= 2\left(\sum w_1 x_i^2 - \sum y_i x_i \right)$$

$$\Longrightarrow \quad \sum_{i=1}^{N} x_i y_i = w_1 \sum_{i=1}^{N} x_i^2$$

$$\Longrightarrow \quad w_1 = \frac{\sum_{i=1}^{N} x_i y_i}{\sum_{i=1}^{N} x_i^2}.$$



In general: $w_0 = \frac{1}{N} \sum_{i=1}^{N} y_i - w_1 \frac{1}{N} \sum_{i=1}^{N} x_i$

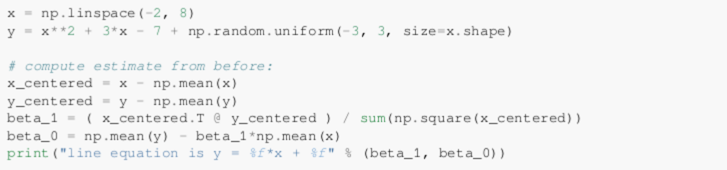
w_1 if $w_0 \neq 0$: Get it by first centering the data: $y_i \longmapsto y_i - \text{mean}(y_i)$

$x_i \longmapsto x_i - \text{mean}(x_i)$

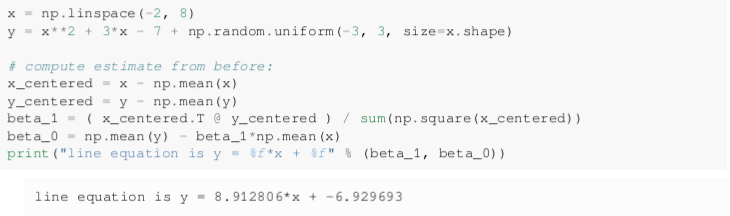
[Shifting doesn't change the slope]

$$W_0 = \mathrm{mean}(y) - W_1 \mathrm{mean}(x).$$

$$W_1 = \frac{\sum_{i=1}^{N} (x_i - \mathrm{mean}(x))(y_i - \mathrm{mean}(y))}{\sum_{i=1}^{N} (x_i - \mathrm{mean}(x))^2}$$

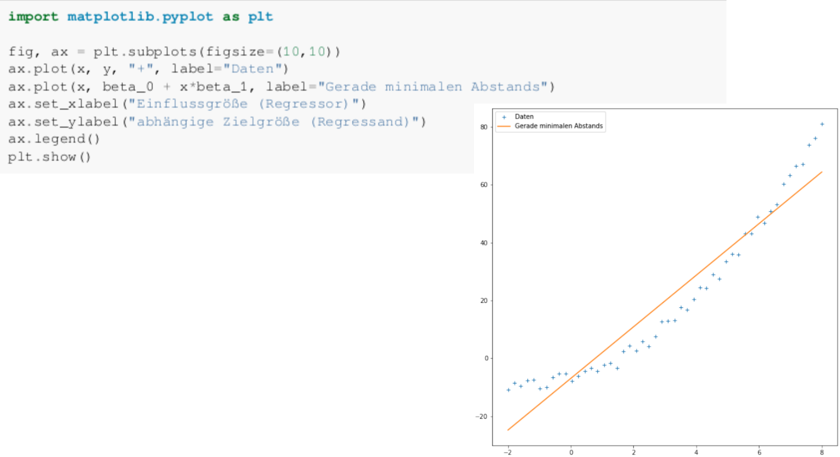


```
x = np.linspace(-2, 8)
y = x**2 + 3*x - 7 + np.random.uniform(-3, 3, size=x.shape)

# compute estimate from before:
x_centered = x - np.mean(x)
y_centered = y - np.mean(y)
beta_1 = ( x_centered.T @ y_centered ) / sum(np.square(x_centered))
beta_0 = np.mean(y) - beta_1*np.mean(x)
print("line equation is y = %f*x + %f" % (beta_1, beta_0))
```

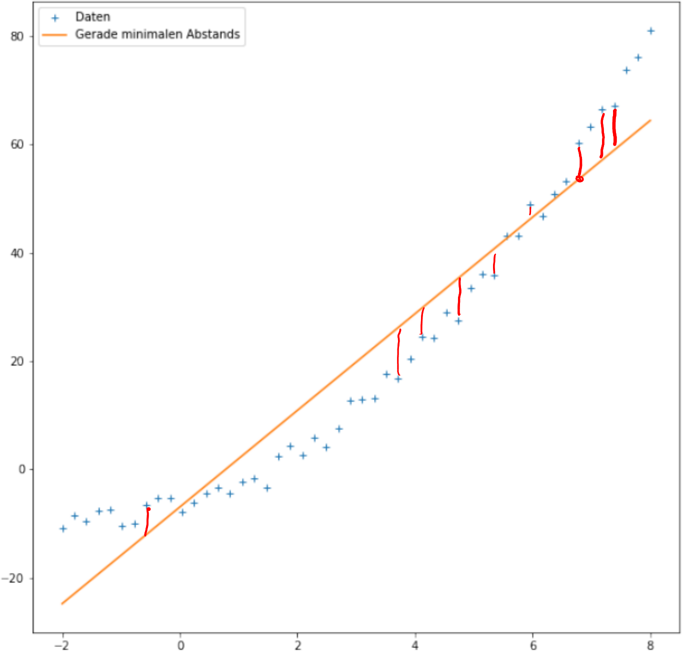


```
x = np.linspace(-2, 8)
y = x**2 + 3*x - 7 + np.random.uniform(-3, 3, size=x.shape)

# compute estimate from before:
x_centered = x - np.mean(x)
y_centered = y - np.mean(y)
beta_1 = ( x_centered.T @ y_centered ) / sum(np.square(x_centered))
beta_0 = np.mean(y) - beta_1*np.mean(x)
print("line equation is y = %f*x + %f" % (beta_1, beta_0))
```

```
line equation is y = 8.912806*x + -6.929693
```



```
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(10,10))
ax.plot(x, y, "+", label="Daten")
ax.plot(x, beta_0 + x*beta_1, label="Gerade minimalen Abstands")
ax.set_xlabel("Einflussgröße (Regressor)")
ax.set_ylabel("abhängige Zielgröße (Regressand)")
ax.legend()
plt.show()
```


Daten
Gerade minimalen Abstands
80
60
40
20
0
−20
−2
0
2
4
6
8

```
from sklearn.linear_model import LinearRegression

# first look at linear, to compare with the above:
linear_regressor = LinearRegression()
linear_regressor.fit(x.reshape(-1, 1), y)
Y_pred = linear_regressor.predict(x.reshape(-1, 1))
```

```
from scipy.optimize import curve_fit

# define function class for regression (linear):
def f(x, beta_0, beta_1):
    return beta_1*x + beta_0

# perform regression analysis:
parameters, covariance_matrix = curve_fit(f, x, y)
assert np.all(np.isclose((beta_0, beta_1), parameters))
```

Many explanatory variables:

$$Y = w_0 + w_1 X_1 + \dots + w_n X_n + \varepsilon$$

$$w = \begin{pmatrix} w_0 \\ \vdots \\ w_n \end{pmatrix}, \quad X^{(0)} = \begin{pmatrix} 1 \\ X_1 \\ \vdots \\ X_n \end{pmatrix} \quad \rightsquigarrow \quad Y = w^T \cdot X^{(0)} + \varepsilon = X^{(0)T} \cdot w + \varepsilon$$

$\rightsquigarrow$ minimize $\sum_{k=1}^{N} \left(Y^{(k)} - w^T X^{(k)}\right)^2$ (least squares)

Matrix notation:

$$Y = \begin{pmatrix} Y^{(1)} \\ \vdots \\ Y^{(N)} \end{pmatrix} \qquad X^T = \begin{pmatrix} 1 & X_1^{(1)} & \dots & X_n^{(1)} \\ \vdots & \vdots & & \vdots \\ 1 & X_1^{(N)} & \dots & X_n^{(N)} \end{pmatrix} \qquad Y = X^T w$$

$$\rightsquigarrow \quad \hat{w} = \underbrace{(X^T)^+}_{\text{pseudoinverse}} \cdot Y$$

if columns of X are lin. indep.

$$(X^T X)^{-1} X^T$$

Probabilistic view:

$$Y = w^T X + \varepsilon \qquad \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

$$\rightsquigarrow \quad Y \mid X \sim \mathcal{N}(w^T X, \sigma^2)$$

Linear regression: model specification (1)

Single data point / linear function

- location x is a vector
- function value at x is modelled as $x^T w$ which is linear in x
- *measured* value y is Gaussian distributed around $x^T w$

$$p(y|x, w) = \mathcal{N}(y|x^T w, \sigma^2) \qquad \text{univariate}$$

- σ^2 is the variance of the measurement noise
- value y is scalar
- parameter w is unknown
- parameter σ^2 is known
- because $x^T w$ is linear in w this is linear regression

Linear regression: model specification (2)

Multiple data points / linear function

- location matrix X contains vectors $x_1, \dots, x_n$ as rows (why rows? see next point)
- function values at X are modelled as Xw which is linear in X (using rows in X makes Xw really simple and minimalistic)
- *measured* values y are Gaussian distributed around Xw

$$p(y|X, w) = \mathcal{N}(y|Xw, \Sigma) \qquad \text{multivariate}$$

- vector y contains for each x_i a scalar value
- because Xw is linear in w this is linear regression

Towards linear regression for nonlinear functions

Basis function expansion

- for scalar x the polynomial basis function

$$\phi(x) = [1, x, x^2, \ldots, x^d]^T$$

or e.g.

$\phi(x) = [\sin(\pi n x)$ for $n = 1 \ldots k]$

leads to polynomials in x

$$y = \phi(x)^T w = \sum_{i=0}^{d} w_i x^i = w_0 + w_1 x + w_2 x^2 + \ldots + w_d x^d$$

- for vector $x = [x_1, x_2]$ the polynomial basis function

$$\phi(x) = [1, x_1, x_2, x_1^2, x_2^2, x_1 x_2, \ldots, x_1^d, x_2^d]^T$$

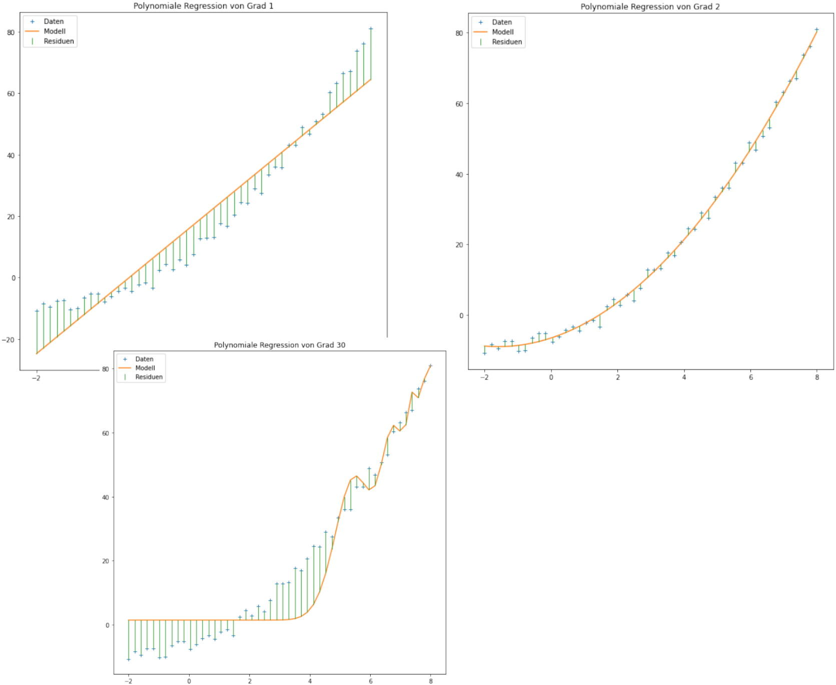
leads to polynomials in x_1 and x_2 (or simply in x):

$$\phi(x)^T w = \sum_{i+j \leq d} w_{ij} x_1^i x_2^j$$

$$= w_{00} + w_{10} x_1 + w_{01} x_2 + w_{20} x_1^2 + w_{02} x_2^2 + w_{11} x_1 x_2 + \ldots + w_{d0} x_1^d + w_{0d} x_2^d$$

- in general ϕ maps vector x nonlinearly onto vector $\phi(x)$
- the entries of vector $\phi(x)$ are also called *features*
- $\phi(x)^T w$ is linear in w and possibly nonlinear in x

```python
# define basis function class
def phi(x, d):
    return np.dstack([np.power(x, i) for i in range(d+1)]).squeeze()

def plot_regression(x, y, Y_pred, ax, x_test=None, y_test=None):
    ax.plot(x, y, "+", label="Daten")
    if(x_test is None):
        x_test = x
    ax.plot(x_test, Y_pred, label="Modell")
    if(y_test is None):
        y_test = y
    yerr = np.stack(( np.maximum(np.zeros_like(y_test), y_test - Y_pred),
                     -np.minimum(np.zeros_like(y_test), y_test - Y_pred)))
    ax.errorbar(x_test, y_test,
                yerr=yerr,
                linewidth=0, elinewidth=1,
                label="Residuen")
    ax.set_xlabel("Einflussgröße (Regressor)")
    ax.set_ylabel("abhängige Zielgröße (Regressand)")
    ax.legend()
    return ax

def poly_regression_plot(x, y, d):
    # perform regression analysis:
    transformed_x = phi(x, d)

    linear_regressor = LinearRegression()
    linear_regressor.fit(transformed_x, y)
    Y_pred = linear_regressor.predict(transformed_x)

    fig, ax = plt.subplots(figsize=(10,10))
    plot_regression(x, y, Y_pred, ax).\
      set_title("Polynomiale Regression von Grad %d" % d)
    plt.show()
```



Polynomiale Regression von Grad 1

Polynomiale Regression von Grad 2

Polynomiale Regression von Grad 30

Linear regression: model specification (3)

Single data point / nonlinear function

- function value at x is modelled as $\phi(x)^T w$ which is nonlinear in x
- *measured* value y is Gaussian distributed around $\phi(x)^T w$

$$p(y|x,w) = \mathcal{N}(y|\phi(x)^T w, \sigma^2) \qquad \text{univariate}$$

- because $\phi(x)^T w$ is linear in w this is linear regression

Linear regression: model specification (4)

Multiple data points / nonlinear function

- $\phi(X)$ is the matrix with rows $\phi(x_1), \ldots, \phi(x_n)$
- function values are modelled as $\phi(X)w$ which is nonlinear in X
- *measured* values y are Gaussian distributed around $\phi(X)w$

$$p(y|X, w) = \mathcal{N}(y|\phi(X)w, \Sigma) \qquad \text{multivariate}$$

- because $\phi(X)w$ is linear in w this is linear regression

Notes

- $\phi(X)$ has just new locations along the rows
- for readability we consider only X instead of $\phi(X)$
- however, all results hold for both X and $\phi(X)$

Goal: estimate parameter vector w

Question

Why is linear regression called linear?

Answers:

A Because it is linear in the features.

B Because it is linear in the parameters.

C Because it honors Francois Philippe Marquis de l'Inéar.

D Because it sounds more scientific than just regression.

Remember:

Linear regression is linear
because it is linear in the parameters.

Maximum likelihood estimation (1)

ML estimator

$$\theta_{\mathrm{ML}} = \arg\max_{\theta} p(\mathcal{D}|\theta)$$

- iid data $\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$
- "iid" means *independent identically distributed*
- iid implies that likelihood factorizes

$$p(\mathcal{D}|\theta) = \prod_{i=1}^{n} p(y_i|x_i, \theta)$$

- this (common) notation is a bit weird, $\mathcal{D}$ was only left of bar, but on RHS x_i is conditioned on
- however, that's ok, the location is always assumed to be known, also for prediction
- so more precisely, $\mathcal{D}$ only contains the values $y_1, \dots, y_n$
- better: $p(y|X, \theta) = \dots$

Maximum likelihood estimation (2)

Instead to look at the likelihood we consider the

Log-likelihood

$$
\begin{aligned}
\ell(w) &= \log p(\mathcal{D}|w) = \sum_{i=1}^{n} \log p(y_i|x_i, w) \\
&= \sum_{i=1}^{n} \log \mathcal{N}(y_i|x_i^T w, \sigma^2) \\
&= -\frac{n}{2\sigma^2} \underbrace{\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i^T w)^2}_{\text{mean squared error}} - \frac{n}{2}\log(2\pi\sigma^2)
\end{aligned}
$$

- mean squared error (MSE) is also called *sum of squared error*, ℓ_2 norm of residual errors, etc.
- ML estimation assuming a Gaussian likelihood leads to the method of least squares

Maximum likelihood estimation (3)

- ML estimation assuming a Gaussian likelihood leads to the method of least squares
- Thus: if we have Gaussian distributed measurements, then least squares is a well-justified method (via ML)
- In Gauss' paper from 1809, he started with the mean which was an established estimator in science (since it is intuitive) and wondered what distribution implies using the mean. By this he invented the normal distriution.

Maximum likelihood estimation (4)

Likelihood

$$p(y|X, w) = \mathcal{N}(y|Xw, \Sigma)$$

Closed-form solution for the ML estimator

$$w_{\text{ML}} = (X^T X)^{-1} X^T y$$

- aka *ordinary least squares* (OLS)
- OLS can be derived by setting the derivative of $\log \mathcal{N}(y|Xw, \Sigma)$ wrt w to zero and solve for w

Maximum likelihood estimation (5)

$d = 1$

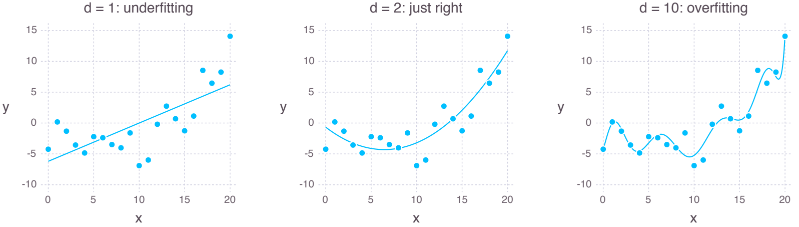
$$f(x) = w_0 + w_1 x$$

$d = 2$

$$f(x) = w_0 + w_1 x + w_2 x^2$$

$d = 10$

$$f(x) = \sum_{i=0}^{10} w_i x^i$$



Notes

- *underfitting* happens if the model is not flexible enough
- *overfitting* happens if the model is too flexible for the amount of data

Ridge regression (1)

Overfitting of ML

- too many parameter, too little data
- usually the weights are very large

Idea

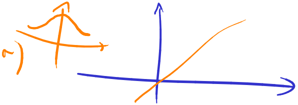
- encourage smoother solutions by putting a zero-mean Gaussian prior on *w* to keep it small

$$p(w) = \mathcal{N}(w|0, \tau^2 I)$$

- the variance τ^2 controls the strength of this prior
- do MAP estimation

Ridge regression (2)

$w \sim \mathcal{N}(0, \tau^2)$



MAP estimation

$$\begin{aligned}
w_{\text{ridge}} &= \operatorname{argmax}_w p(w|X, y) = \operatorname{argmax}_w p(y|X, w)p(w|X)/p(y|X) \\
&= \operatorname{argmax}_w p(y|X, w)p(w) \\
&= \operatorname{argmax}_w \sum_{i=1}^{n} \log \mathcal{N}(y_i|x_i^T w, \sigma^2) + \sum_{j=1}^{d} \log \mathcal{N}(w_j|0, \tau^2) \\
&= \operatorname{argmin}_w \boxed{\underbrace{\boxed{\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i^T w)^2}}_{\text{fit}} + \underbrace{\lambda\|w\|_2^2}_{\text{regularizer}}}
\end{aligned}$$

$e^{-\frac{x^2}{2\tau^2}}$

with $\lambda = \sigma^2/\tau^2$ (just move the σ^2 from the first summand to the second summand and merge with τ^{-2}) and $\|w\|_2^2 = \sum_j w_j^2$.

Solution

$$w_{\text{ridge}} = (\lambda I + X^T X)^{-1} X^T y$$

Ridge regression (3)

Solution

$$
\begin{aligned}
w_{\text{ridge}} &= \operatorname{argmin}_w \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i^T w)^2 + \lambda\|w\|_2^2 \\
&= (\lambda I + X^T X)^{-1} X^T y
\end{aligned}
$$

Notes

- ridge regression is the same as penalized least squares
- $\lambda\|w\|_2^2$ is ℓ_2 regularization (aka weight decay)

Ridge regression (4)

Question

- Why regularize by adjusting λ, why not changing *d*?

Towards an answer

- *d* sets model complexity
- λ measures inverse signal-to-noise ratio (next slides)

Bayesian linear regression (1)

Question

- Can we also derive the posterior distribution over w (instead of point estimates via ML and MAP)?

Prior and likelihood

$$\begin{aligned} p(w) &= \mathcal{N}(w|w_0, V_0) \\ p(y|X, w) &= \mathcal{N}(y|Xw, \Sigma) \end{aligned}$$

Posterior

$$\begin{aligned} p(w|X, y) &= \mathcal{N}(w|w_n, V_n) \\ V_n &= (X^T \Sigma^{-1} X + V_0^{-1})^{-1} && \text{posterior covariance} \\ w_n &= V_n(V_0^{-1} w_0 + X^T \Sigma^{-1} y) && \text{posterior mean} \end{aligned}$$

Bayesian linear regression (2)

Posterior

$$p(w|X,y) = \mathcal{N}(w|w_n, V_n)$$

$$V_n = (X^T\Sigma^{-1}X + V_0^{-1})^{-1} \qquad \text{posterior covariance}$$

$$w_n = V_n(V_0^{-1}w_0 + X^T\Sigma^{-1}y) \qquad \text{posterior mean}$$

Notes

- for $\Sigma = \sigma^2 I$, $V_0 = \tau^2 I$, $w_0 = 0$, the mean of the posterior corresponds to ridge regression

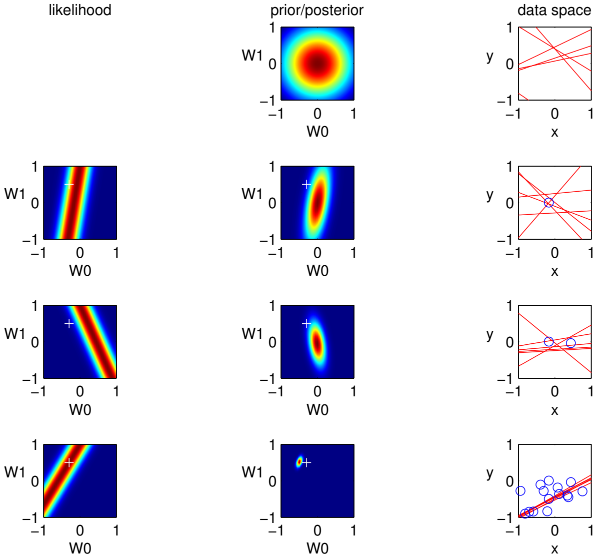
$$w_n = (\lambda I + X^T X)^{-1} X^T y = w_{\text{ridge}}$$

- however, here we have additionally the posterior covariance

$$\begin{aligned} V_n &= \sigma^2(\sigma^2/\tau^2 I + X^T X)^{-1} \\ &= \sigma^2(\lambda I + X^T X)^{-1} \end{aligned}$$

- so λ is the inverse signal-to-noise ratio

Bayesian linear regression (3)



from Murphy's book Figure 7.11

Bayesian linear regression (4)

Posterior predictive distribution

- often we want to do prediction
- thus we integrate the parameter out
- training data $\mathcal{D}$ with n pairs of locations and values
- how is value y at a new data location x distributed?

$$\begin{aligned} p(y|x,\mathcal{D}) &= \int \mathcal{N}(y|x^T w, \sigma^2)\,\mathcal{N}(w|w_n, V_n)\,dw \\ &= N(y|x^T w_n, \sigma_n^2) \\ \sigma_n^2 &= \sigma^2 + x^T V_n x \end{aligned}$$

- note that the variance is location dependent
- again for $\Sigma = \sigma^2 I$, $V_0 = \tau^2 I$, $w_0 = 0$:

$$\sigma_n^2 = \sigma^2(1 + x^T(\lambda I + X^T X)^{-1} x)$$

Alternatives to least squares (1)

Linear regression

$$p(y|x,w) = \mathcal{N}(y|x^T w, \Sigma)$$

Robust linear regression

$$p(y|x,w) = \mathrm{Lap}(y|x^T w, b) = \exp(-\frac{1}{b}\|y - x^T w\|)/Z(b)$$

Alternatives to least squares (2)

Likelihoods and priors for linear regression

Likelihood	Prior	Name
Gaussian	Uniform	Least squares
Gaussian	Gaussian	Ridge regression
Gaussian	Laplace	Lasso
Laplace	Uniform	Robust regression
Student	Uniform	Robust regression

copied from Murphy's book Table 7.1

- note that, uniform prior leads to ML
- however, such a uniform prior can often not be normalized

End of Section 0~~8~~ 6